Drupal for Digital Humanities Projects

Erica Cavanaugh
Alix Shield
Welcome to DHSI 2018!

Thanks for joining the DHSI community!

In this booklet, you will find essential course materials prefaced by some useful information about getting settled initially at UVic, finding your way around, getting logged in to our network (after you’ve registered the day before our courses begin), and so on.

Given our community’s focus on things computational, it will be a surprise to no one that we might expect additional information online for some of the classes - your instructors will let you know - or that the most current version of all DHSI-related information may be found on our website at dhsi.org.

To access the DHSI wifi network, simply go into your wireless settings and connect to the “DHSI” network and enter the password “dhsi2018”.

And please don’t hesitate to be in touch with us at institut@uvic.ca or via Twitter at @AlyssaA_DHSI or @DHInstitute if we can be of any help ....
DHSI Buildings
DHSI, SINM & DLF Buildings
Residences
Buildings Under Construction
General Pay Parking
Reserved Parking
Student Resident Parking
Bus Stop
Parkade
Food Service Outlet
Welcome Centre
Electric Vehicle Charging Station
The 2018 schedule is just about ready! A very few things to confirm, add, etc, but this is the place to be to find out what is happening when / where ...

**Psst: Some Suggested Outings**

If you're here a day or two before we begin, or staying a day or two afterwards, here are a few ideas of things you might consider doing ...

- **Suggested Outing 1, Botanical Beach (self-organised; car needed)**
  A self-guided visit to the wet, wild west coast tidal shelf (and historically-significant former research site) at Botanical Beach; we recommend departing early (around 8.00 am) to catch low tide for a better view of the wonderful undersea life! Consider bringing a packed lunch to nibble-on while looking at the crashing waves when there, and then have an afternoon drink enjoying the view from the deck of the Port Renfrew Hotel.

- **Suggested Outing 2, Butchart Gardens (self-organised)**
  A shorter journey to the resplendently beautiful Butchart Gardens and, if you like, followed by (ahem) a few minutes at the nearby Church and State Winery, in the Saanich Peninsula. About an hour there by public bus from UVic, or 30 minutes by car.

- **Suggested Outing 3, Salt Spring Island (self-organised; a full day, car/bus + ferry combo)**
  Why not take a day to explore and celebrate the funky, laid back, Canadian gulf island lifestyle on Salt Spring Island. Ferry departs regularly from the Schwartz Bay ferry terminal, which is about one hour by bus / 30 minutes by car from UVic. You may decide to stay on forever ....

- **Suggested Outing 4, Paddling Victoria's Inner Harbour (self-organised)**
  A shorter time, seeing Victoria's beautiful city centre from the waterways that initially inspired its foundation. A great choice if the day is sunny and warm. Canoes, kayaks, and paddle boards are readily rented from Ocean River Adventures and conveniently launched from right behind the store. Very chill.

- And more!

  Self-organised High Tea at the Empress Hotel, scooter rentals, visit to the Royal BC Museum, darts at Christies Carriage House, a hangry breakfast at a local diner, whale watching, kayaking, brew pub sampling (at Spinnaker's, Swans, Moon Under Water, and beyond!), paddle-boarding, a tour of used bookstores, and more have also been suggested!

**Sunday, 3 June 2018 [DHSI Registration + Suggested Outings]**

9:00 to 4:00

- **Early Class Meeting: 4. [Foundations] DH For Department Chairs and Deans (Hickman 120, Classroom)**
  Further details are available from instructors in mid May to those registered in the class. Registration materials will be available in the classroom.

3:00 to 5:00

- **DHSI Registration (MacLaurin Building, Room A100)**
  After registration, many will wander to Cadboro Bay and the pub at Smuggler's Cove OR the other direction to Shelbourne Plaza and Maude Hunter's Pub OR even into the city for a nice meal.

**Monday, 4 June 2018**

Your hosts for the week are Alyssa Arbuckle, Ray Siemens, and Dan Sondheim.

7:45 to 8:15

- **Last-minute Registration (MacLaurin Building, Room A100)**

8:30 to 10:00

- **Welcome, Orientation, and Instructor Overview (MacLaurin A144)**
The abstract is about the history of DH, including the medieval history related to Busa, and how DH work that centers social justice is particularly relevant in the current global rise of late fascism. The author also discusses the concept of language as a physical fact, in which substance, not just form, is semantic. They argue that DH has a renewed importance as we head into a moment of feminist historiography.

The talk discusses the interpretive and methodological implications of using 3D printing technologies to prototype the archival diagrams of a proposed but never constructed plastic segmental alphabet letter kit—a game designed by modernist poet Mina Loy for F.A.O. Schwarz. Although intended as a toy for young children, it was also a tool for scholars. The author proposes to share a concept-based approach to interface design for DH.

They also propose to share a concept-based approach to interface design for DH. This approach is based on the idea that DH scholars are working to create new tools for the study of language and culture, and that these tools can be used to explore the connections between DH and other fields, such as social justice and feminism. They argue that DH is not just about creating new tools, but about using those tools to explore and understand the world around us.

The talk also includes a demonstration of the way in which DH can be used to explore and understand the world around us. They present a series of visualizations and diagrams that illustrate the connections between DH and other fields, such as social justice and feminism. The author also discusses the implications of these connections for the future of DH, and how we can use DH to explore and understand the world around us.
Randa El Khatib (U Victoria): "Learning from the Iterative Process."
Abstract: #MyDHis Iterative. In addition to the improvements that come with iterative projects, the iterative process itself is a fruitful area for scholarly inquiry. Within this iterative context, the various teams that I work with and I have been reflecting on and rethinking central DH practices, such as what it means to collaborate, prototype, remix, and implement DH values in our work. In this talk, I will present the various lessons learnt along the way.

Sarah Melton (Boston C): "#MyDHis...People."
Abstract: Taking seriously Miriam Posner’s exhortation to “commit to DH people, not DH projects,” I invite us to reflect on how people are the core of DH. In this brief talk, I will explore the intersections between DH, labor, and infrastructure.

5:00 to 6:00
Opening Reception (University Club)
We are grateful to Gale Cengage for its sponsorship.

Tuesday, 5 June 2018

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch break / Unconference
"Mystery" Lunches
• DHSI Lunchtime Workshop Session (click for workshop details and free registration for DHSI participants)
  - 73. Introduction to ORCID (Digital Scholarship Commons, Classroom).

1:30 to 4:00
Classes in Session

• DHSI Colloquium Lightning Talk Session 1 (MacLaurin A144)
  Chair: James O’Sullivan
  • New Modes of DH and Archival Skills Acquisition in a Graduate Public History Course. Paulina Rousseau (Ryerson U)
  • Walking a Transect: Exploring a Soundscape. John Barber (Washington State U)
  • Centering the Edge Case: Designing Services for Humanities Data Research. Grace Afsari-Mamagani (New York U)
  • Orwellian Vocabulary and the 21st-Century Politics. Ilgin Kizilgunesler (U Manitoba)
  • Making Open Data from a Gray Archive. Sara Palmer (Emory U)

6:00 to 8:00
DHSI Newcomer’s Beer-B-Q (Felicitas, Student Union Building)

Wednesday, 6 June 2018

9:00 to Noon
Classes in Session

Lunch break / Unconference
"Mystery" Lunches
• Brown Bag Lecture: Alexandra Branzan Albu (U Victoria): “Visual Recognition of Symbolic and Natural Patterns”
  (Digital Scholarship Commons, 3rd Floor McPherson Library)

Abstract: Image-based object recognition is a visual pattern recognition problem; one may characterize visual patterns as either symbolic or natural. Symbolic patterns evolved for human communication; they include but are not limited to text, forms, tables, graphics, engineering drawings etc. Symbolic patterns vary widely in terms of size, style, language, alphabet and fonts; however, literate humans can easily compensate for this variability and instantly recognize most symbolic patterns. On the other hand, natural patterns characterize images of physical structures; they often lack the intrinsic discriminability and structure of symbolic patterns, and vary widely in terms of pose, perspective, and lighting.

This lecture will explore similarities and differences in approaches designed for recognizing visual and symbolic patterns, and will address the following questions via examples.
- What are the distinctive characteristics of natural patterns? What dimensions of variability can we infer?
- What are the distinctive characteristics of symbolic patterns? What dimensions of variability can we infer?

Alexandra Branzan Albu is an Associate Professor with the Department of Electrical and Computer Engineering and cross-listed with Computer Science. Her research interests are related to image analysis, computer vision, and visual computing. She is actively pursuing outreach activities dedicated to increasing the women's presence in electrical engineering and computer science.

1:30 to 4:00
Classes in Session
Thursday, 7 June 2018

9:00 to Noon
Classes in Session

12:15 to 1:15
UVIC Library/ETCL lunchtime talk: “A Humanities Application of 3D printing and Machine Translation in the ChessBard and Loss Sets” by Dr. Aaron Tucker
Digital Scholarship Commons, 3rd floor, Mearns Centre for Learning / McPherson Library
Bring your lunch and come on up!.

1:30 to 4:00
Classes in Session

4:15 to 5:15
DHSI Colloquium Lightning Talk Session 3 (MacLaurin A144)
Chair: James O’Sullivan
- Documenting Deportation: A Collaborative Digital Collection. Paulina Rousseau (Ryerson U)
- Unleashing the Power of Texts as Networks: Visualizing the Scholastic Commentaries and Texts Archive. Jeffrey Witt (Loyola U Maryland) and Drew Winget (Stanford U)
- #haunteDH: Punching holes in the International Busa Machine Narrative. Arun Jacob (McMaster U)
- Text in World: Computational Analysis of Trauma in Genocide Narratives. Nanditha Narayanamoorthy (U York) and Krish Perumal (U Toronto)

7:30 to 9:30
(Groovy?) Movie Night (MacLaurin A144)

Friday, 8 June 2018 [DHSI; DLFxDHSI Opening]

9:00 to Noon
DHSI Classes in Session

12:15 to 1:15
DHSI Lunch Reception / Course E-Exhibits (MacLaurin A100)

1:00 to 2:00
DLFxDHSI Registration (MacLaurin A100)

1:30 to 1:50
[DHSI] Remarks, A Week in Review (MacLaurin A144)

2:00 to 3:00
Joint Institute Lecture (DHSI and DLFxDHSI):
Bethany Nowviskie (CLIR DLF and U Virginia): “Reconstitute the World: Machine-reading Archives of Mass Extinction”
Chair: Lisa Goddard (U Victoria)
(MacLaurin A144)

Abstract: The basic constitution of our digital collections becomes vastly more important in the face of two understandings: first, that archives of modernity are archives of the sixth great mass extinction of life on our planet; and next, that we no longer steward cultural heritage for human readers alone. In the same way that we people are shaped by what we read, hear, and see, the machine readers that follow us into and perhaps beyond the Anthropocene have begun to learn from “unsupervised” encounters with our digital libraries. What will we preserve for the living generations and artificial intelligences that will come? What do we neglect, or even choose to extinguish? And from an elegiac archive, a library of endings, can we create forward-looking, speculative collections—collections from which to deep-dream new futures? The most extra/ordinary power we possess is the power to make poetry from records of the past. Could it be called on, one day, to reconstitute the world?
Saturday, 9 June 2018 [DLFxDHSI + DHSI Conference and Colloquium]

8:30 to 9:00
DLFxDHSI Registration (MacLaurin A100)

9:00 to 5:30
DLFxDHSI UnConference Sessions

9:00 to 4:00
• 53. Building Your Academic Digital Identity (MacLaurin D105, Classroom)

• DHSI Colloquium Day Conference (MacLaurin A144)

Welcome
People I: Documenting Online Lives. Chair: Molly Nebiolo (University of New York)
- Examining Gendered Harassment Online and in Silicon Valley. Andrea Flores (Utica College)
- This is Just to Say I Have <X> the <Y> in your <Z>: Modernist Memes in an Era of Public Apology. Shawna Ross (Texas A&M University)

Break

People II: Documenting Lives Online. Chair: Dheepa Sundaram (College of Wooster)
- Youtube Yoga and Ritual on Demand: The Virtual Economics of Hindu Soteriology. Dheepa Sundaram (College of Wooster)
- The Resemblage Project: Creativity and Digital Health Humanities in Canada. Andrea Charise (University of Toronto) and Stefan Krecsy (University of Toronto)

Lunch

Projects I: Building and Analyzing. Chair: Yannis Rammos (New York University)
- Building the ARTECHNE Database: New directions in Digital Art History. Marieke Hendriksen (Old Dominion University)
- The Ineffective Inquisition: The Holy Office’s Sphere of Influence in Early Modern New Spain. Kira Homo (Pennsylvania State University)

Break

Projects II: Mapping and Visualizing. Chair: Innocent Opara (Qumet Institute)
- Mapping Sarah Sophia Bank’s Numismatic Collection. Erica Hayes (North Carolina State University) and Kacie Wills (University of California, Riverside)
- Text Mining and Visualizing 18th Century American Correspondence. Ashley Sanders Garcia (University of California, Los Angeles)

Break

Practices: Digital Scholarship on Campus and in the Classroom. Chair: Alyssa Arhuckle (University of Victoria)
Sunday, 10 June 2018 [SINM + DHSI Registration, Workshops]

8:30 to 9:00  Symposium on Indigenous New Media Registration (MacLaurin A100)

9:00 to 5:00  DHSI Registration (MacLaurin A100)

9:00 to 4:00  SINM Sessions
- 63. Symposium on Indigenous New Media: Reading Group (Hickman 105, Classroom)
- 72. Symposium on Indigenous New Media: Indigitization (Hickman 120, Classroom)
  Full details here

9:00 to 4:00  DHSI All Day Workshop Sessions (click for workshop details and free registration for DHSI participants)
- 53. Building Your Academic Digital Identity (MacLaurin D105, Classroom)
- 54. An Introduction to the Archaeology of 1980s Computing (MacLaurin D114, Classroom)

9:00 to Noon  DHSI AM Workshop Sessions (click for workshop details and free registration for DHSI participants)
- 55. Regular Expressions (MacLaurin D111, Classroom)
- 56. 3D Visualization for the Humanities (MacLaurin D010, Classroom)
- 58. DH Fieldwork Methods (MacLaurin D016, Classroom)
- 60. Pedagogy of the Digitally Oppressed: Inculcating De-/Anti-/Post-Colonial Digital Humanities (MacLaurin D107, Classroom)
- 61. Introduction to #GraphPoem. Digital Tools for Poetry Computational Analysis and Graph Theory Apps in Poetry (MacLaurin D101, Classroom)
- 62. Creating a CV for Digital Humanities Makers (MacLaurin D115, Classroom)

1:00 to 4:00  DHSI PM Workshop Sessions (click for workshop details and free registration for DHSI participants)
- 64. Agent-Based Modelling in the Humanities (MacLaurin D111, Classroom)
- 65. Unleash Linux on MacOS (MacLaurin D010, Classroom)
- 66. DHSI Knits: History of Textiles and Technology (MacLaurin D016, Classroom)
- 67. Crowdsourcing as a Tool for Research and Public Engagement (MacLaurin D109, Classroom)
- 69. Web Annotation as Critical Humanities Practice (MacLaurin D103, Classroom)
- 70. Dynamic Ontologies for the Humanities (MacLaurin D107, Classroom)
- 71. Social Media Research in the Humanities (MacLaurin D101, Classroom)

4:10 to 5:00  Joint Institute Lecture (DHSI and SINM):
  David Gaertner (U British Columbia): "A Landless Territory?: CyberPowWow and the Politics of Indigenous New Media."
  Chair: Deanna Reder (Simon Fraser U)
  (MacLaurin A144)
  Abstract: Following the 1997 launch of Skawennati’s (Mohawk) CyberPowWow, digital space has become a vital new territory for the resurgence of Indigenous storytelling and cultural practice: “We have signed a new treaty,” Cree artist Archer Pechawis wrote of this period, “and it is good. We have the right to hunt, fish, dance and make art at www.CyberPowWow.net, .org and .com for as long as the grass grows and the rivers flow.” This talk will critically explore the theoretical, cultural, political-economic, and gendered dynamics underwriting the histories and futures of Indigenous new media. Particular attention will be given in examining the ways in which new media and digital storytelling connect to and support key issues in the field of Indigenous studies, such as sovereignty, self-determination, decolonization, and land rights.

After the day, many will wander to Cadboro Bay and the pub at Smuggler’s Cove OR the other direction to Shelbourne Plaza and Maude Hunter’s Pub OR even into the city for a bite to eat.

Monday, 11 June 2018 [DHSI + SINM]
Your hosts for the week are Ray Siemens and Dan Sondheim.

| 7:45 to 8:15 | DHSI Last-minute Registration ([MacLaurin A100](#)) |
| 8:30 to 10:00 | DHSI Welcome, Orientation, and Instructor Overview ([MacLaurin A144](#)) |
| 9:00 to 4:00 | SINM Sessions |

- DHSI Classes in Session (click for details and locations)
  - 28. [Foundations] Developing a Digital Project (With Omeka) ([Clearihue D132, Classroom](#))
  - 29. [Foundations] Models for DH at Liberal Arts Colleges (& 4 yr Institutions) ([MacLaurin D109, Classroom](#))
  - 32. Stylometry with R: Computer-Assisted Analysis of Literary Texts ([Clearihue A102, Lab](#))
  - 33. Digital Storytelling ([MacLaurin D111, Classroom](#))
  - 34. Text Mapping as Modelling ([Clearihue D131, Classroom](#))
  - 35. Geographical Information Systems in the Digital Humanities ([Clearihue A105, Lab](#))
  - 36. Open Access and Open Social Scholarship ([MacLaurin D114, Classroom](#))
  - 37. Introduction to Machine Learning in the Digital Humanities ([Cornett A229, Classroom](#))
  - 38. Queer Digital Humanities: Intersections, Interrogations, Iterations ([MacLaurin D110, Classroom](#))
  - 41. Using Fedora Commons / Islandora ([Human and Social Development A160, Lab](#))
  - 42. Documenting Born Digital Creative and Scholarly Works for Access and Preservation ([MacLaurin D115, Classroom](#))
  - 43. Games for Digital Humanists ([MacLaurin D016, Classroom & Human and Social Development A170, Lab](#))
  - 44. XPath for Document Archeology and Project Management ([Cornett A128, Classroom](#))
  - 46. Surveillance and the Digital Humanities ([MacLaurin D103, Classroom](#))
  - 47. Text Analysis with Python and the Natural Language TooKit ([Clearihue A103, Lab](#))
  - 48. Information Security for Digital Researchers ([Clearihue D130, Classroom](#))
  - 49. Wrangling Big Data for DH ([Human and Social Development A150, Lab](#))
  - 50. Accessibility & Digital Environments ([MacLaurin D101, Classroom](#))
  - 51. Critical Pedagogy and Digital Praxis in the Humanities ([MacLaurin D105, Classroom](#))
  - 52. Drupal for Digital Humanities Projects ([MacLaurin D107, Classroom](#))

| 10:15 to Noon | Lunch break / Unconference Coordination Session ([MacLaurin A144](#)) (Grab a sandwich and come on down!)

DHSI Lunchtime Workshop Session (click for details and locations)

- 73. Introduction to ORCID ([Digital Scholarship Commons, Classroom](#))

| 12:15 to 1:15 | Joint Institute Lecture (DHSI and SINM): Jordan Abel (Simon Fraser U): "Indigeneity, Conceptualism, and the Borders of DH." Chair: Michelle Brown (U Hawaii) ([MacLaurin A144](#))

Abstract: This talk brings together digital humanities discourses in computational textual analysis and Indigenous Literary Studies to analyze a corpus comprised of every book of Indigenous poetry published in Canada, extending from Pauline Johnson's 1895 book The White Wampum to Marilyn Dumont's 2015 book The Pemmican Eaters. While the main goal of this research project initially centered on the topic modeling of a corpus of Indigenous poetry, the project also addresses the systemic barriers that have prevented such work gaining traction, and likewise attempts to address the specific challenges that Indigenous writing (and in particular Indigenous poetry) present to current Digital Humanities methodologies.

| 5:00 to 6:00 | Joint Reception: DHSI and SINM ([University Club](#))

| 9:00 to Noon | Classes in Session |

| 12:15 to 1:15 | Lunch break / Unconference "Mystery" Lunches

DHSI Lunchtime Workshop Session (click for workshop details and free registration for DHSI participants)

- 73. Introduction to ORCID ([Digital Scholarship Commons, Classroom](#)).
Wednesday, 13 June 2018

1:30 to 4:00 Class in Session
4:15 to 5:15 DHSI Colloquium Lightning Talk Session 4 (MacLaurin A144) 
Chair: Lindsey Seatter
- Mapping Indigenous and Chicanx/o Environmental Imaginaries using GIS. Stevie Ruiz (California State U, Northridge), Quetzalli Enrique (California State U, Northridge), Enrique Ramirez (California State U, Northridge), and Tomas Figueroa (California State U, Northridge)
- Doing DH with Graphic Narratives. John Barber (Washington State U)
- "But is it any good?: A quantitative approach to the popularity of digital fanfiction. Suzanne Black (U Edinburgh)
- The American Prison Writing Archive (APWA). Doran Larson (Hamilton C), Janet Simons (Digital Humanities Initiative, Hamilton C), and William Rasenberger (Hamilton C)
6:00 to 8:00 DHSI Newcomer's Beer-B-Q (Felicitas, Student Union Building)

Thursday, 14 June 2018

9:00 to Noon Classes in Session
12:15 to 1:15 Lunch break / Unconference
- "Mystery" Lunches
1:30 to 4:00 Classes in Session
4:15 to 5:15 DHSI Colloquium Lightning Talk Session 5 (MacLaurin A144) 
Chair: Lindsey Seatter
- Faraway, so close: Has the political environment really changed in Ecuador? Luis Meneses (Electronic Textual Cultures Lab, U Victoria)
- Re-mixing Melville's Reading: Text Analysis of Marginalia with R and XSLT. Christopher Ohge (U London, School of Advanced Study) and Steven Olsen-Smith (Boise State U)
- Developing Interactive and Open-Source OER: Inquiry-Based Music Theory. Evan Williamson (U Idaho)
- Spatial Humanities and the Web of Everywhere. Ken Cooper (SUNY Geneseo)
6:00 to 7:00 "Half Way There (yet again)!"][An Informal, Self-Organized Birds of a Feather Get-Together] (Felicitas, Student Union Building)
Bring your DHSI nametag and enjoy your first tipple on us!

Friday, 15 June 2018

9:00 to Noon Classes in Session
12:15 to 1:15 Lunch Reception / Course E-Exhibits (MacLaurin A100)
Institute Lecture: William Bowen (U Toronto Scarborough): “Discovery, Collaboration and Dissemination: Lessons Learned and Plans for the Future” (MacLaurin A144)

Abstract: Much has changed and continues to change in digital humanities since the formal establishment of Iter in the Fall of 1997. However, the mandate of the not-for-profit partnership to support “the advancement of learning in the study and teaching of Middle Ages and Renaissance (400–1700) through the development and distribution of online resources” continues to have relevance. This presentation explores the striking challenges faced by Iter and presents our current thinking on the realization of this mandate for the future through a platform with a focus on facilitating the discovery of the academic resources necessary to our work; creating an environment for collaboration, sharing and developing projects; and on enabling the distribution and publication of our scholarship.

2:40 to 3:00

Awards and Bursaries Recognition
Closing, DHSI in Review (MacLaurin A144)

Contact info:
institut@uvic.ca P: 250-472-5401 F: 250-472-5681
Monday, June 11
10:15 -- 10:45  Introductions and overview
  ● DfH 1.2 - Overview and audience
  ● DfH 1.3 - Drupal compared to other CMSes
  ● Review pre-DHSI assignment: “Evaluate a Digital Platform”
10:45 -- 11:30  Installing Drupal (Pantheon) & overview of demo site
  ● Walk through steps on Pantheon; also see DfH 3.4
  ● DfH 1.5 - overview of example site
11:30 -- 12:00  Essential Drupal jargon
  ● DfH 2.4 - understanding Drupal’s components, core & database
  ● DfH 2.5 - Drupal components
12:15 -- 1:15   Lunch
1:30 -- 2:30   Navigating a Drupal site (basic configuration, installing modules, etc.)
  ● DfH 2.6 - overview of site building
  ● DfH 4.4 - essential modules (used as examples for practicing installing modules in 4.5)
  ● DfH 4.5 - installing & enabling modules
  ● DfH SFTP appendix
2:30 -- 4:00   Data modeling and content types
  ● DfH 5.2 - introducing content types and other entities
  ● DfH 5.4 - data modeling for content types
  ● DfH 5.5 - preliminary data modeling for example site

Tuesday, June 12
9:00 -- 10:30  Creating and configuring content types
  ● DfH 6.2 - creating new content types
  ● DfH 6.3 - adding and configuring fields
  ● (Rest of DfH 6 is for reference, but may make reference to individual sections like automatic nodetitles and pathauto)
  ● DfH 7.2 - configuring content types for the example site
  ● DfH 8.5 - adding data (just to create a couple nodes for use in Views)
10:30 -- 11:15  Drupal configuration checklist (user profiles, paths, permissions, etc.)
  ● DfH 9.2 - configuring node display for example content type
  ● DfH 10 - users and permissions, focusing on basic account creation, orientation to the permissions page
  ● DfH 6.9 - Pathauto
  ● DfH 11 - Menus & blocks
11:15 -- 12:00  Intro to Views
12:15 -- 1:15   Lunch
1:30 -- 3:00   Basic View creation
  ● DfH 12 - Views (creation of simple example view)
3:00 -- 4:00     Free time for project work; will meet with people individually to discuss data modeling for their own projects

**Wednesday, June 13**

9:00 -- 10:30     Importing data
   - DfH 14.4 - Importing from CSV
10:30 -- 12:00     Advanced Views: relationships and contextual filters
   - DfH 13.4 - Site timeline (uses Simple timeline, which doesn't require a library); has a relationship
   - DfH 13.8 - Table with exposed filters
12:15 -- 1:15      Lunch
1:30 -- 2:15       Theming and site design
   - DfH 18 - Theming
2:15 -- 4:00       Break-out groups based on shared interests (e.g. mapping, timelines, multimedia)

**Thursday, June 14**

9:00 -- 10:15     Exporting data
   - DfH 15.2 - Views Data export
10:15 -- 11:00     Troubleshooting Drupal
   - DfH 4.3 - Assessing modules
   - DfH 20.9 - Debugging Drupal
11:00 -- 12:00     Updating Drupal (modules, core) and Drupal 8
   - DfH 20.2 - Maintenance activities
   - DfH 20.3 - Core, module and theme updates
   - DfH 20.4 - Database backup
   - DfH 20.5 - File backup
   - DfH 20.6 - Whole-site backup
   - DfH 2.3 - What about Drupal 8?
12:15 -- 1:15      Lunch
1:30 -- 2:30       Demo of Drupal site migration to generic shared hosting
2:30 -- 4:00       Tour of useful modules (Biblio, Taxonomy Manager, etc.) and Q&A

**Friday, June 15**

9:30 -- 12:00     Q&A and wrap-up
When Not to Use Drupal

Submitted by Quinn on Tue, 07/03/2012 - 20:41

WordPress is better sometimes

Of all the CMSes available to humanists, WordPress is both the most convenient to use and the most clearly aligned with traditional scholarship. After a relatively painless installation, WordPress provides the capacity to display long narrative text with embedded images and videos. Without any modification, a WordPress site can be built to contain both a running blog and a set of static sub-pages, open for comment or not, and available to the outside world.

Many universities offer WordPress installations through their Academic Computing or Information Technology departments, making this process even easier. With a wide variety of available themes, the look and feel of a WordPress installation can be dramatically changed to emphasize text or image content.

The scholarly community has shown much love for WordPress. For many academics, a blog is the first foray into digital content creation and WordPress is often the first blog that’s utilized as an application rather than subscribed to as a service from sites like Blogger and WordPress.com. The blog has been dealt with as a form, theoretically and productively, for quite some time. Even back in 2006, projects like the Institute for the Future of the Book (http://www.futureofthebook.org/) was trying to develop methods to make the blog form more nuanced through the creation of WordPress extensions like CommentPress (http://www.futureofthebook.org/commentpress/).

WordPress is particularly well-suited as a mechanism to support individual scholarship and commentary using traditional linear narratives. If an individual or project is only interacting with the outside world via reporting their findings in the format of traditional articles or short posts, then WordPress is a perfectly acceptable solution.

That said, once you try to move beyond the blog form to integrate the kind of collaborative workspaces and community-oriented functionality typically needed for a moderately collaborative digital project, WordPress is lacking. And if you want to deal with data within that site, and taxonomies and data models and visual representation of such data, then you’re better off in Drupal, which can provide you with all the functionality necessary to present long narratives with embedded images but also provide you the capability to integrate datasets and sophisticated collaborative tools.
Omeka is better sometimes

Omeka (http://omeka.org/) is, like Drupal, a CMS built on the MySQL-PHP technology stack. Omeka is designed for exhibiting narratives and collections to audiences, and with the recent release of Neatline (http://neatline.org), which extends Omeka to include sophisticated geospatial and temporal functionality, these narratives can include rich geographic and chronological data. Omeka, especially enabled with Neatline, is also well-supported by two key institutions in digital humanities research: CHNM (http://chnm.gmu.edu/) and the Scholars' Lab (http://www2.lib.virginia.edu/scholarslab/) at UVa. Another benefit of Omeka is that it's focused on metadata standards for public humanities work like that found in museums and libraries, and handles much of the taxonomic effort that you may find unwieldy in Drupal. As such, if your project is focused on the presentation of a narrative with visual, textual and spatial data, then you should seriously consider using Omeka.

For any digital humanities project where the aim is to present time and space-enabled data, you should try out Omeka with the Neatline extensions. If you're just getting started in creating a collection of objects, places, and so on, you should also first take a look at Omeka. The value of Drupal over Omeka comes when your use case becomes too specific or complex and you need to create specific data models, views into the data, and services to connect to your work.

TEI is much better for certain things

When a humanist ventures into the world of digital textual scholarship and data archiving, they quickly encounter TEI (the Text Encoding Initiative (http://tei-c.org)). Many scholars are first exposed to the idea of separating style from content-- indicating what something means (e.g. book title, ship name, emphatically spoken word), rather than how it should be displayed (italics)-- through a workshop on TEI. The TEI Consortium offers a set of extremely detailed and extensible guidelines for how to add metadata to text, with a major use case being the development of digital critical editions. Projects involving the preparation of textual materials are all but expected to use TEI, particularly if they are applying for grants, and using TEI is widely seen as a way to increase the likelihood that digital textual scholarship will be accessible in the long term.

Scholars who are only used to working with word processing software face a learning curve with TEI. There is something initially counterintuitive about taking a human-readable text and rendering it nearly unreadable with angled brackets full of additional information. The fact that rendering a TEI encoded document readable again involves writing XSLT exacerbates this issue. There have been various efforts aimed at addressing these challenges, with the TEI-ann and TEILiteEditor plugins providing an alternative to manually typing angled brackets, and TEICHI (http://www.teichi.org) enabling basic search and display of TEI texts in Drupal, to name a few.

The impulse to use a durable format for the sake of preservation can push a scholar towards TEI, even when it doesn’t align well with his overall goals for the textual material. For a project on Thomas Jefferson, constructing a TEI bibliography of his library might be the best choice, particularly if the project also involves marking up his personal papers in TEI. But a continually evolving bibliography of secondary literature about Vladimir Nabokov-- even if its compilation is a research project that spans a scholar's entire career-- is less of a good fit for TEI, and even less so if the scholar wants to solicit additional entries from Nabokov scholars worldwide. A TEI-based


approach to such a project might involve the following steps:

1. Taking the Word document where the data has been collected (probably without any markup whatsoever) and copying the text into an XML document
2. One-by-one, marking up the data using the <biblStruct> element, and its numerous sub-elements (<analytic>, <monogr>, <imprint>, <pubPlace>, and <biblScope>, to name a few typical examples)
3. Writing XSLT to output various kinds of listings (all entries sorted by author, all entries sorted by date, only monographs, etc.)
4. Realizing that some of the shortcuts and errors in your TEI “break” your listings (e.g. listing all authors of a book together makes it difficult to group together works by author)
5. Cleaning up the TEI XML to address the issues in #4
6. Generating static HTML from your XML and XSLT, and posting them on your personal university web space
7. Emailing a list of Nabokov scholars and soliciting additional bibliography entries, which you manually add to your TEI document when they email them to you
8. Realizing that it’s a pain to have to re-generate all your HTML files every time you add a new entry to your TEI document, and resigning yourself to the drudgery, or...
9. Trying to get your local IT support to configure and support a server with the rather specialized Apache Cocoon web framework, which can run the XSLT “on the fly”, so you only need to upload the updated XML and the server does the rest of the work

This workflow requires knowledge of how to use an XML editor, how to mark up texts using TEI, and how to write XSLT, as well as an IT department with the resources to support niche projects and the knowledge to set up and maintain a Java-based framework they’re probably not already using.

Achieving the same goals with Drupal might involve:

1. Having your local IT support install Drupal; its requirements are much simpler than Apache Cocoon, and your institution might already be running it
2. Install the Bibliography module (http://drupal.org/project/biblio)
3. Enter the bibliography entries from the Word document into web forms that are easy enough to understand that you might feel comfortable delegating some or all of the data entry to student assistants
4. Go to yoursit.yourinstitution.edu/biblio at any point during or after the data entry process to browse your bibliography and sort and filter by a variety of metadata fields
5. Install the Views module and use the interface to generate any listings not provided out-of-the-box by the Bibliography module
6. Email a list of Nabokov scholars and invite them to create an account on your site, and provide them with simple instructions for how to add new listings
7. Sit back and watch your bibliography grow

All of these steps can be done by a humanist without any specialized training or a background in technology, following the instructions laid out in this book.

A Drupal-based bibliography facilitates collaboration and display of the bibliography, but what about preservation?
It’s unrealistic to expect your Drupal site will run on a server forever, or that your library will archive your data with the full Drupal environment in tow. At some point(s) in the project’s lifecycle, the data will need to migrate to a more archivable form, and there are Drupal modules that can help you do that; see [section about import/export].

When deciding between TEI and Drupal, consider the following factors:

- Is there a module that aligns well with the kind of markup I’d be doing in TEI (e.g. the Bibliography module)?
- If not, is my markup simple enough that I can achieve the same kind of granularity in my data storage by creating a custom content type (e.g. if you’d only be marking up your text for poem, stanza, and line)?
- What are potential funders’ expectations for the use of TEI in projects similar to mine?
- How important is it that my data interoperate with other TEI texts as part of a larger initiative? This may not preclude the use of Drupal for data entry, if you have a strategy worked out for exporting and converting the data into TEI.
- How much collaboration is required for this project?

If your project uses detailed markup that covers more than just structural elements of the text, chances are that TEI is the right choice. For simpler text encoding projects, it’s important to weigh the pros and cons of TEI and Drupal as they relate to your goals for your data, amount of technical support available, and workflow. Drupal might be a better choice, but it may also require that you explain to colleagues why you did not use TEI; be thorough in your exploration of both options and clear about the rationale behind your decision if you choose the Drupal path.

**Hand-crafted .js for maps/visualizations is better sometimes**

Most humanities research using digital tools will produce some form of representation of relationships, demographics or other phenomena related to research. This often takes the form of sophisticated plots and diagrams as is produced in software like Tableau, or cartographic format as results from spatial analysis using ArcGIS, or network representation like that produced by network analysis toolkits like Gephi. Each of these software packages provides the ability to not only represent the rich content of a dataset but also to interrogate the individual components and interact with the data on its own. When a scholar or project needs to share the results of these explorations between members or with the community at large, these tools provide the capacity to output static images with legends for just such a purpose. Posting these images in a web-accessible manner is trivial, and can be accomplished in any CMS.

However, there is often a need to provide researchers or community members with the capacity to explore the dataset with some of the same interactive functionality provided by the desktop tools but in web form. This is typically accomplished by bringing to bear one of the rich Internet scripting platforms, like Flash, Silverlight or JavaScript in the browser. These platforms have various libraries to provide data visualization and interactivity that can work off the same data the scholars are examining with their desktop applications.

Typically, you would not think of using a CMS to deal with rich data visualization and interactivity, but that’s beginning to change, especially with geographic data in Drupal, as will be explained in the Spatial Drupal chapter.
Integration of data as content types into Drupal comes with significant overhead and typically provides less functionality and slower performance than hand-crafted scripts. If all one wants to do is show a flat dataset in a well-known visualization pattern, then the costs of building it in Drupal far outweigh any benefit.

But, if a project is utilizing some form of visual representation directly in their research, such as geographic representation of the research matter, then Drupal allows that scholar or project to represent a timeline or map-based view of the ongoing, collaborative research.

Other CMS (Django/Flask/ZoobieX, etc.)

To be written...

Tags:  WordPress Django TEI Omeka Javascript maps

(http://drupal.forhumanists.org) (http://creativecommons.org/licenses/by-sa/3.0/)
Assignment: Evaluate a Digital Platform

A content management system is a computer application that makes it easier to digitally publish by offering tools like a back-end, with buttons and text editors. The site is easy to format and reformat because the content is stored in a database. Often times Drupal is the perfect solution for a digital humanities project due to its flexibility. However, as Quinn Dombrowski’s article “When Not to use Drupal” illustrates, this is not always the case. It is important to evaluate content management systems and publication platforms before using them. Use the following questions to evaluate 1 of the applications listed to get an idea of what other systems offer.

WordPress
Omeka
Scalar
FromThePage
Mukurtu

- How does this platform distinguish itself from others?
- Who built (builds) the platform?
- For whom is the platform intended (the audience)?
- What language(s) is (are) it written in?
- What kinds of sites or projects currently use this platform?
- How large and active is the user community?
- What possibilities does it offer for display? For example, how easy is it to reconfigure the form of a project? How many options are configurable?
- Does the platform seem to assume that you want to display content in a certain configuration? If so, what is it?
- How easy is it for someone to install this platform and use it?
- Who can modify the platform’s source code?
- What kind of database does the CMS use?
- Can content be exported/imported via various file formats?
- What standards, if any, does the platform follow? Can it be mapped to different standards?
- Can you attach metadata to the content you enter? If so, what kind?
- What is the cost associated with installing and maintaining this platform?
Table of Contents

1. Overview

2. Introducing Drupal
   2.1 Overview
   2.2 The Drupal community
   2.3 What about Drupal 8?
   2.4 Understanding Drupal's components: code and database
      2.4.1 Drupal core code
      2.4.2 Module code
      2.4.3 Theme code
      2.4.4 Database
      2.4.5 Summary of code and database
   2.5 Drupal components
      2.5.1 Content types
      2.5.2 Nodes
      2.5.3 Taxonomies
      2.5.4 File types
      2.5.5 Views
      2.5.6 Blocks
      2.5.7 Menus
      2.5.8 Users
      2.5.9 Roles
      2.5.10 Summary of Drupal components
   2.6 Overview of site building
      2.6.1 Deciding on Drupal
      2.6.2 Basic configuration
      2.6.3 Data modeling
      2.6.4 Implementing the data model
      2.6.5 Improving the display
      2.6.6 Theming
      2.6.7 Pre-launch configuration

3. Installing Drupal
   3.1 Pantheon
   3.2 Installing Drupal
      3.2.1 Choose a profile
      3.2.2 Choose language
      3.2.3 Verify requirements, set up database, install profile
      3.2.4 Configure site
         3.2.4.1 Site information
         3.2.4.2 Site maintenance account
         3.2.4.3 Server settings
      3.2.5 Update notifications

4. Modules
   4.1 Overview
4.2 Finding modules
4.3 Assessing modules
  4.3.1 Modules published on drupal.org vs. those published elsewhere
  4.3.2 Module versions
  4.3.3 Number of users
  4.3.4 Maintenance and development status
  4.3.5 Release notes
4.4 Essential modules
4.5 Installing and enabling modules
  4.5.1 Installing modules using the Drupal interface
  4.5.2 Installing modules by accessing the filesystem
    4.5.2.1 Why sites/all/modules?
  4.5.3 Enabling modules
  4.5.4 Configuring modules
4.6 How many modules do you need?
4.7 Disabling and uninstalling modules
  4.7.1 Disabling
  4.7.2 Uninstalling
  4.7.3 Removing the code from your site
5. Content types and introduction to data modeling
  5.1 Overview
  5.2 Introducing Content types
  5.3 Drupal’s default content types: Basic Page and Article
  5.4 Data modeling for content types
    5.4.1 Data vs metadata
    5.4.2 Content type vs. taxonomy
    5.4.3 When to merge similar content types
    5.4.4 File types vs. content types
    5.4.5 User profiles vs. content types
    5.4.6 Drupal-based considerations for determining content types
    5.4.7 Drupal’s “title” field vs real-world titles
  5.5 Preliminary data modeling for the example site
    5.5.1 The data
    5.5.2 The goal
    5.5.3 Breaking down the data
    5.5.4 Enriching the content types
      5.5.4.1 Person
      5.5.4.2 Event
      5.5.4.3 Image
6. Configuring content types and fields
  6.1 Overview
  6.2 Creating new content types
    6.2.1 Submission form settings
    6.2.2 Publishing options
6.2.3 Display settings
6.2.4 Comment settings
6.2.5 Menu settings

6.3 Adding and configuring fields
6.3.1 "Manage fields" interface
6.3.2 Adding fields

6.4 Core Drupal fields
6.4.1 Text
6.4.2 Long text
6.4.3 List (text)
6.4.4 Term reference
6.4.5 File and the media browser widget
   6.4.5.1 Configuration for uploaded files
6.4.6 Image

6.5 Fields provided by modules
6.5.1 References
6.5.2 Date
6.5.3 Partial date
   6.5.3.1 Partial date formats
   6.5.3.2 Base estimate values
   6.5.3.3 Minimum components
   6.5.3.4 Field configuration
6.5.4 Link
6.5.5 Geofield
6.5.6 Field group

6.6 Media
6.6.1 Configuring the file system
6.6.2 Configuring file settings
6.6.3 Configuring media browser settings
6.6.4 Configuring file type profiles

6.7 Conditional fields

6.8 Configuring Automatic Nodetitles
6.8.1 Advanced Automatic Nodetitles configuration with Entity Tokens
6.8.2 Automatic Nodetitles and Node Reference Create

6.9 Improving node URLs with Pathauto

7. Further data modeling and applied content type creation

7.1 Overview
7.2 Configuring content types for the example site
   7.2.1 Person
   7.2.1.1. Content type settings
   7.2.1.2 Fields
   7.2.1.3 Automatic nodetitles
   7.2.1.4 Pathauto
   7.2.2 Event
7.2.2.1. Content type settings
7.2.2.2 Fields
7.2.2.3 Pathauto

7.2.3 Image
7.2.3.1 Content type settings
7.2.3.2 Fields
7.2.4.3 Automatic Nodetitles
7.2.4.4 Pathauto

7.4 Conclusion

8. Configuring input forms and adding content
8.1 Overview
8.2 Text formats
  8.2.1 Field configuration
  8.2.2 Text format configuration
  8.2.3 Configuring a text format
  8.2.4 Additional filters
8.3 WYSIWYG configuration
  8.3.1 Installing modules and CKEditor library
  8.3.2 Selecting CKEditor in the WYSIWYG module
  8.3.3 Configuring CKEditor for the Filtered HTML text format
  8.3.4 Facilitating internal linking
8.4 Organizing content editing forms
8.5 Adding content
  8.5.1 Configuration options
  8.5.2 Previewing content
8.6 Viewing and editing content
8.7 Revisions

9. Node display
9.1 Overview
9.2 Configuring node display
  9.2.1 Label and field visibility
  9.2.2 Labels and visibility for Person content type
  9.2.3 Labels and field visibility for Event content type
  9.2.4 Labels and field visibility for Image content type
9.3 Configuring date fields
  9.3.1 Date fields in the example site
  9.3.2 Configuring partial date fields
  9.3.3 Partial date field in the example site
9.4 Configuring geospatial fields
  9.4.1 Geospatial fields in the example site
9.5 Configuring image fields
  9.5.1 Configuring image styles
  9.5.2 Advanced image style configuration
  9.5.3 Colorbox
9.5.4 Image fields on the example site
9.6 Configuring file fields and multimedia
9.7 CCK Blocks

10. Users and permissions
10.1 Overview
10.2 User configuration settings
10.3 Account settings
   10.3.1 Registration and cancellation
      10.3.1.1 Account registration
      10.3.1.2 Account cancellation
   10.3.2 Personalization (user avatars)
   10.3.3 System emails
10.4 Creating user profiles
   10.4.1 Real Name
10.5 Roles and permissions
   10.5.1 Content permissions
   10.5.2 Field permissions
   10.5.3 Permissions for unpublished nodes
   10.5.4 Permissions for multi-author nodes
   10.5.5 Taxonomy access control
10.6 Spam prevention
   10.6.1 Honeypot
   10.6.2 CAPTCHA

11. Menus and blocks
11.1 Overview
11.2 Blocks
   11.2.1 “Demonstrate block regions” and positioning tricks
   11.2.2 Assigning blocks to regions
   11.2.3 Block configuration
   11.2.4 Region settings
   11.2.5 Visibility settings
   11.2.6 Custom blocks
11.3 Menus
   11.2.1 Default menus
   11.3.2 Adding a menu
   11.2.3 Adding items to menus
      11.2.3.1 Editing a node
      11.3.3.2 Add link
   11.3.4 Arranging menu items
   11.3.5 Making a menu available
   11.3.6 Dropdown menus

12. Views
12.1 Overview
12.2 A very simple view using “add new view”
12.2.1 Choosing what to display
12.2.2 Displays
12.2.3 Next steps for new views
12.3 Displays
12.3.1 Titles, and overriding and reverting display settings
12.4 Format
  12.4.1 Grouping fields
  12.4.2 Show
12.5 Fields
  12.5.1 Configuring date fields
  12.5.3 Configuring image fields
  12.5.4 Rearranging fields
  12.5.5 Style settings
  12.5.6 Inline fields and rewriting fields
    12.5.6.1 Using “Format” to display inline fields
    12.5.6.2 Excluding and rewriting fields
12.6 Filter criteria
  12.6.1 And/or options
12.7 Sort criteria
12.3.7 Display-specific settings
  12.3.7.1 Page settings
  12.3.7.2 Feed settings
  12.3.7.3 Block settings
12.3.8 Header & Footer
12.3.9 Pager

13. Advanced Views

14. Importing data
  14.1 Overview
  14.2 Essential modules for data import
  14.3 Overview of Feeds settings
    14.3.1 Basic settings
    14.3.2 Fetcher
      14.3.2.1 HTTP Fetcher
      14.3.2.2 File upload
    4.3.3 Parser
      14.3.3.1 Common syndication parser
      14.3.3.2 CSV parser
      14.3.3.3 HTML Xpath parser and XML Xpath parser
    14.3.4 Processor
      14.3.4.1 Node and taxonomy term processor
      14.3.4.2 User processor
    14.3.5 Mapping
      14.3.5.1 Unique targets
      14.3.5.2 Term reference fields
14.3.5.3 Node reference fields
14.3.5.4 Image and file targets
14.3.5.5 Date targets
14.3.6 Feeds Tamper
   14.3.6.1 Adding a plugin
   14.3.6.2 Explode
   14.3.6.3 Trim
   14.3.6.4 Rewrite
   14.3.6.5 Find replace / find replace REGEX
   14.3.6.6 Copy source value
   14.3.6.7 HTML entity decode
   14.3.6.8 Convert case
   14.3.6.9 Strip tags
   14.3.6.10 Keyword filter
14.3.7 Importing data
14.3.8 Import page options
14.4 Example: importing from a CSV
   14.4.1 Source data
   14.4.2 Creating the importer and basic configuration
   14.4.3 Mapping
   14.4.4 Feeds tamper
   14.4.5 Importing
14.5 Importing from RSS feeds
   14.5.1 Source data
   14.5.3 Mapping
   14.5.4 Feeds tamper
   14.5.5 Importing
14.6 Importing from XML
   14.6.1 Source data
   14.6.2 Creating the importer and basic configuration
   14.6.3 Settings for XML Xpath parser
   14.6.4 Processor settings, mapping and debugging
   14.6.5 Feeds tamper
   14.6.6 Importing

15. Search
16. Taxonomy modules
17. Themes
18. Other useful modules
19. Running and maintaining a Drupal site
20. Debugging Drupal
21. Writing documentation
22. Appendix A: SFTP
   22.1 Installing software
   22.2 Connection information
1. Overview
This course pack is based on a draft of the forthcoming *Drupal for Humanists* book by Quinn Dombrowski, to be published by Texas A&M Press. It reflects the state of the book as of March 2015. Students will have access to a more recent (and hopefully complete) draft on Google Docs. Feedback is very much welcome! Please talk to Quinn, or email quinnd@berkeley.edu with any comments.

2. Introducing Drupal

2.1 Overview
This chapter provides an overview of Drupal as a software platform and as an open source community.

2.2 The Drupal community
Drupal is open source software, supported by one of the largest communities of developers. Over 25,000 people have contributed code to Drupal. These developers include hobbyists who develop or contribute to modules for fun, people who work at companies that build Drupal websites professionally, and people who work for organizations that use Drupal (including universities, businesses, and the governmental and non-profit sectors). While you may think that only the code contributed by people affiliated with a university is likely to be relevant for your project, digital scholarly projects have much more in common with product catalogs and directories of government offices than it may initially seem. All these sites need a way to create user accounts and assigning different levels of permissions to different users. They all need to inhibit spammers, display information in menus, and provide intuitive interfaces for data entry and data display. Mapping can be an important feature on a website, whether it’s the shipping origin of a package, the reported location of a pothole, or the source of an archaeological artifact.

A vast amount of time has been spent on developing and refining Drupal’s code, and the maintenance, support, and development of new and old modules and themes is ongoing. The result of all this work is available to anyone to freely use and modify. All of the modules and themes distributed on the Drupal website are free to download; only a tiny handful of niche modules distributed through other channels require payment. While there is a vibrant marketplace for paid themes (site designs), there are many options available at no cost.

In one sense, it is as if projects that use Drupal have a development staff of thousands of people, even if the project itself includes no programmers. In another sense, because there are no developers dedicated specifically to your project beyond ones you hire yourself, if you encounter obscure bugs or issues with a module (or, worse, a particular combination of modules), they may linger unresolved for a long period of time. Module developers have
different levels of personal commitment to maintaining the modules they create. Some modules are widely used by people who have the skills to create “patches” that can fix problems that arise, and these users sometimes offer those solutions to others who are encountering the same problem, even before the module maintainer has addressed it. Choosing widely-used modules (see section 4.3) reduces the risk of encountering problems that remain unaddressed indefinitely. Sometimes, though, you may need to hire a developer to fix a bug, or write a module to provide specific functionality for your site. While this may cost a few thousand dollars, by making the fix or the module available to the rest of the community you can give back to the Drupal ecosystem and perhaps save another project some time and money in the future. This approach stands in contrast to the custom development of a project or database from scratch, where little or none of that work is reusable by others.

2.3 What about Drupal 8?

This book covers the steps necessary to build a site using Drupal 7. This choice may seem odd, since Drupal 8 is almost available. The Drupal philosophy is to provide backward compatibility for data (to make it possible to migrate your site to a new version), but not for code\(^1\), meaning that every module has to be rewritten between versions to reflect significant changes to the underlying architecture of Drupal. Because of this, there has historically been a lag time of about a year between when a new major version of Drupal is released and when there’s sufficient module support for the new version to truly be usable for new projects.

The architectural changes in Drupal 8 are at a much bigger scale than changes between previous versions. Drupal has historically had its own, somewhat quirky requirements for how certain programming tasks are implemented. In an effort to attract professional PHP programmers, it was decided that Drupal 8 would adopt conventions and frameworks that are widely used in the broader PHP development community. These conventions are very different from the “Drupal way” of doing things, and some Drupal developers feel like these changes make Drupal less accessible, particularly for small organizations and projects. A group of these developers have started a Drupal 7 “fork” (a new content management system split off from Drupal, but based on the Drupal 7 core and module code at the point of the split) called Backdrop\(^2\).

As an analogy, imagine a small, vibrant community of philosophers in 17th century Barcelona, where participants only write treatises in Catalan. The use of Catalan within this community is treated as such a given that participants may never have learned Latin, even though Latin is the language of scholarship elsewhere in Europe. In order to increase the influence of this school of philosophy and contribute to conversations happening elsewhere in Europe, a number of influential figures within this group decide that all treatises must henceforth be written in Latin. Other members of the community are disgruntled at the fact that their treatises in progress must be rewritten in Latin. They feel that learning Latin would take too much time and would distract them from their actual work. Latin’s case system is seen as too difficult to learn fluently, and philosophers are concerned they won’t be able to express themselves with the same nimbleness as they could before. The philosophers with the greatest concerns break away from the original community and continue to develop the same philosophical ideas they worked with before, while maintaining the use of Catalan (perhaps with the adoption of a few more Latinisms as a gesture towards the larger community).

\(^1\) https://www.drupal.org/node/65922
\(^2\) https://groups.drupal.org/node/325403
It remains to be seen how many people will adopt Backdrop, and whether the concerns that fueled the origins of the project will in fact be realized once Drupal 8 has been released for some time, and module development does (or does not) flourish for the new version. The fact remains that there are thousands of scholarly sites built on Drupal 7 (and Drupal 6), and they all will face migration decisions: to Drupal 8, to Backdrop, or to something else. Drupal 7 will be supported until Drupal 9 is released. This will likely be a few years off; Drupal 8 took more than four years from the time development started until it was released. By building a site in Drupal 7, you can take advantage of the benefits Drupal provides now, while leaving your options open for future migration, depending on how things evolve within the community.

2.4 Understanding Drupal's components: code and database

On a technical level, a Drupal site consists of code, mostly written in the language PHP, with some written in JavaScript, and some CSS (Cascading Stylesheets) to make it look attractive. The code “talks to” a database -- usually MySQL -- that stores Drupal- and module-specific configuration information, as well as the content you enter into the site (names and birth dates of people, project profiles, transcriptions of documents, etc.)

If you have previous experience with SQL and databases, you may be taken aback if you look directly at Drupal’s database. It is not configured in a straightforward, intuitive manner; no one creating a database from scratch would create something that remotely resembles a Drupal database. While this can be frustrating for developers who are trying to build new modules and understand how Drupal stores information, for “site builders” who are creating sites by combining and configuring different modules, what the database looks like “under the hood” is irrelevant. Drupal’s code knows how to interact with the database, and it’s strongly recommended that you only make changes to the database using Drupal's code (or, for developers, Drupal’s APIs) as a translator and intermediary. Making changes to the database directly (for instance, by sending SQL commands to the database and/or using an interface to the database itself like PHPMyAdmin) is dangerous, and often has unpredictable side effects, like data disappearing or getting corrupted.

Most of the work of building a Drupal site involves using the administrative interface (which is largely determined by the code) to customize settings for the site and add content, which makes changes to the database.

While Drupal’s code can easily remain unseen by the person creating the site (referred to in Drupal jargon as the “site builder”), it is important to have a general understanding of the different sources of your site’s code, so you can make better guesses about where in the site configuration to go to make different kind of changes, and effectively troubleshoot when things go wrong.

At the same time, keep in mind that the code that makes up your site is not what makes the site unique. If all the code for your site were accidentally deleted, as long as you know what versions of which modules your site uses, you could re-generate your site exactly as it was within a couple hours by re-downloading Drupal core and modules. If
your database and/or “files” directory vanished, you would be in dire straits without a backup.

2.4.1 Drupal core code

The most basic Drupal site consists of the core Drupal code, plus a database. The Drupal core is the software available at http://drupal.org/project/drupal, and includes all the essential infrastructure and plumbing for a basic site (menus, user account management, file uploading capabilities, in addition to the Drupal administrative interface itself), as well as an extremely limited number of modules that provide specific site functionality. Modules that are part of Drupal 7 core include those that enable content types, taxonomies for tagging and classifying content, comments, a basic discussion forum, and contact forms.

Because Drupal-powered sites can have such diverse needs, but all are required to use Drupal core, the Drupal development team has deliberately kept the number of core modules to a minimum. Users are expected to identify, download, and install the additional modules that meet the needs of their individual project.

2.4.2 Module code

Modules are packaged-up code that provide some new piece of site functionality. This can range from tweaking the label of some field on the administrative interface to completely overhauling the way content access permissions are handled, to providing a word cloud visualization of tags on your site. There are over 10,000 modules available for Drupal 7, ranging from those that meet nearly-ubiquitous needs (like Views, which can display your content in a variety of useful ways, which is installed on over 800,000 sites), to those that solve specific problems (like Shibboleth User Provisioning, which adds features to a module that enables integration with some university authentication systems, and is used on less than 1,000 sites.)

With so many modules, how do you know what to use? Chapter 4 provides recommendations of modules that you should install immediately on almost any site, and subsequent chapters provide recommendations for modules that are useful for different kinds of scholarly projects. In addition, there is a Drupal developer adage that--whatever your site is lacking, or whatever frustration you're encountering-- "there's a module for that". While this not always true, particularly when you encounter difficulties related to a discipline-specific problem (such as grappling with different transliteration conventions for ancient alphabets), searching for keywords related to your need or issue, plus "Drupal module" will often surface helpful additions to your site's module list.
2.4.3 Theme code

The design of your site is largely determined by the site theme. Like modules, themes are packages of code that you add to your site to achieve a certain effect (in this case, to give your site a particular look). Unlike with modules, it’s okay to modify the theme code that you’ve downloaded. Theme code consists of PHP files that specify "regions" (i.e., things like how many columns your site can have, whether you can put text in a footer area, etc.) While all themes include one or more CSS files that determine the site’s visuals (like color, fonts, and text spacing), some themes provide configuration options you can access through the Drupal interface that allow you to change the site’s colors without having to write any CSS yourself.

2.4.4 Database

The core Drupal code and the code for the modules you have installed define what is possible to do on your site. The site’s database stores the information about how you’ve specifically chosen to configure the site, based on the possibilities and constraints provided by the code. For example, the Pathauto module gives you the ability to define the path (URL) that will be used for your content. When you go into the settings for Pathauto and specify that you want your blog posts to appear at yoursit.org/blog/[year]/[month]/[date]/[title], that information is saved into the database.

In addition to storing information about the site configuration, the database stores your site content. This lack of separation between content and configuration can be problematic. For example, if you have a development version of your site and a live version, and you’ve made configuration changes on the development site that you want to move to the live site, you can’t just import the development site’s database if there’s been any content changes to the live site (new pages, new comments on blog posts, new users, etc.) since the last time the two were synced.

If you have only used HTML to develop websites, switching to a database-powered content management system like Drupal opens up a wide range of new options for managing and displaying your data, but it also makes the storage of that data less intuitively accessible. Where you could previously point to a specific .html file that contained the data displayed on a single webpage, the content that appears on an equivalent webpage in a Drupal site is actually stored in many different places throughout the database. But whether you want to backup your data, share it with someone else, or export it for use with other software, there are many options for getting your data out of the Drupal database in a variety of formats, including PDF, Excel-compatible CSV files and XML.

2.4.5 Summary of code and database

The Drupal core code provides the essential plumbing for your site, and a small group of modules that provide basic website functionality. The additional modules you install
provide much of the functionality of your site, and the choice of which modules to install largely depends on what your site needs to do. Your site’s theme and its associated configuration options determine what the site looks like. All the information about your site’s configuration, as well as your site’s content, are stored in the database.

2.5 Drupal components

The Drupal core code provides an administrative interface where you will be doing most of the work involved in configuring your site. The following major components of Drupal can be configured using this interface. Collectively, they make up the structure of the site.

2.5.1 Content types

One of Drupal’s defining features as a content management system is the ability to create content types—essentially, templates for storing different kinds of data. In contrast to WordPress, where a user can choose between creating a blog post and a page—both of which are by default limited to a title, text area, and some tags—Drupal allows you to create any number of content types, each of which can have as many fields as you need for storing different kinds of content (URLs, structured dates, pointers to other pieces of content or users, videos from external hosting providers like YouTube, in addition to multiple types of fields for text and images). This makes Drupal well-adapted for storing and organizing data beyond simple webpage content.

As an example, consider a departmental website. Such a site could make use of Drupal’s default "Basic page" content type—which contains a title field and a body field—for general informational pages about the department and its programs. Departmental news items could be added through the site using the "Article" content type. This content type contains a title and body, but also includes an image field, tags, and has comments enabled by default.

The department could also use the "Basic page" content type to publish course descriptions, but Drupal content types enable the site builder to store that information in individual fields—rather than a large blob of text—and then display the information in a variety of useful ways.

Imagine the site builder creates a content type called "Course". By default, Drupal includes a title field and a body field with every new content type, though the body field can be renamed or removed entirely. The developer might then add the following fields:

- A "topic" field, for selecting from a standardized list of course topics (e.g. 19th century, sexuality)
- An "instructor" field, for selecting one or more faculty members or grad students affiliated with the department
• A “semester” field, for choosing which semester the course has been and/or will be taught
• A "prerequisites" field, for typing in the names of prerequisite courses, where Drupal will autocomplete based on the courses that have already been entered
• A "syllabus" field, for uploading a PDF of the course syllabus

When the site webmaster goes in to add information about a departmental course, they will be presented with an easy-to-use form containing these fields.

While this is a fairly simple content type, using only Drupal core modules and the References module\(^3\), there are modules that enable fields tailored for data ranging from email addresses, links and dates to geographic coordinates and temperature data. Content types can be made even more elaborate using Field groups (see section 6.5.6) - for instance, you might want to group the time a class is scheduled with the room number the class is held. The Field Collection module allows for repeating field groups. This could be useful in the example of a language class that holds drills at a different time and place than the main class session, so the course content type has to accommodate multiple sets of time/location information.

2.5.2 Nodes

In Drupal jargon, a "node" refers to an instantiation of a content type. Whenever you add new content to your site using any content type, you have created a node. For most sites, almost all the data on the site is stored as nodes; an exception might be if your site is primarily a directory of people, in which case your content would be mostly stored in user profiles (see chapter 10).

2.5.3 Taxonomies

Drupal's "taxonomy" system provides an easy way of associating categories, tags, or other controlled or uncontrolled vocabularies with your content and users. You can create multiple vocabularies, which each contain terms. To continue with the departmental website example, the site builder would need to create a vocabulary for course topics, and populate it with the list of terms that the department uses to topically categorize its courses. When adding terms, there are options for adding a description for the term, as well as specifying its parent terms within that vocabulary in order to form hierarchies of terms (e.g. putting “Folklore” and “Children’s literature” under a parent term “Genre studies”).

Furthermore, within a particular vocabulary, site builders can configure fields that will be available when creating or editing any term within that vocabulary. For example, a site builder might create a vocabulary called "University", where users can specify the university they are affiliated with as part of their user profile. A site builder could add a field, “Address” to the “University” vocabulary. Every university entered by a user would

\(^3\) https://www.drupal.org/project/references
be stored as a taxonomy term, and users with the right set of administrative privileges could edit those terms to add the address for the university. This would make it possible for the site to display a map showing the universities that its users are affiliated with, using the Views module.

When a node or user is tagged with a term from a vocabulary, by default that term appears on the node or user page as a link to a term page that displays all other nodes or users tagged with that same term, making it easy to browse related content.

2.5.4 File types

The Media module allows site builders to create “File types”, profiles that are associated with one or more kinds of uploaded files, and/or files pulled in from a third-party hosting provider like YouTube. By default, Drupal provides file types for audio, documents, images, and video. You can add fields to any of these file types, just like you would with a content type, so that users who upload a file need to fill in some metadata. You can also create multiple file types for the same kind of file, and give them different fields -- for instance, in order to differentiate painting and photography.

2.5.5 Views

By default, Drupal provides two options for displaying content: 1) viewing a full node, and 2) seeing the default "content feed" (much like the WordPress "posts" page) that displays the shortened “teaser” version of the most recent nodes you've created. This may be suitable for a simple blog, but any site that has custom content types will need the Views module to make the best use of those content types. Chapters 11 and 12 cover Views in depth, but at its essence, Views allows you to query your database in simple or complex ways without writing any code. A very small subset of the possible displays you can generate with Views (often with the help of additional modules) includes:

- A table of all site users and their university affiliation, with dropdown criteria (drawn from fields in the user profile, such as discipline) that can be used to filter the list
- Rotating slideshows
- A gallery of image thumbnails
- A list of blog posts written by the person whose profile you’re currently viewing
- A map displaying all the locations stored as part of nodes, where clicking on a location pushpin pulls up the title and a link to the node where those coordinates can be found
- A CSV export of some or all the content on the site
- An RSS feed
- A list of the most active site users
- Information stored in the user profile of a user referenced in the node you’re currently viewing -- in the departmental website example, Views could display on a course page whether the instructor is a professor, lecturer, or grad
student, without the webmaster having to enter that information as part of the course profile.
The results of these queries can be displayed on your site in many ways, including as a standalone page with its own URL, as a feed, as a plain-text file for exporting data as a spreadsheet, or as a block (see below).

2.5.6 Blocks

Blocks are generally small containers of content or site functionality (such as a user login box, or a menu) that can appear in different places on the site (e.g. footer, right sidebar on blog post pages, top right corner of the front page). Blocks can be created by modules or views, or you can manually create a custom block, which is convenient for things like a copyright notice for a site footer.

On the block administration page, you can drag and drop blocks into the different regions (header, footer, sidebar, etc.) that have been defined by your site's theme. In addition to determining where on a page the block appears, you can configure the block to appear only on pages that meet certain criteria. For example, if you have a block that's only relevant within a particular area of the site, like "Related blog posts", you can specify that it should only appear on pages from the "Blog" content type.

2.5.7 Menus

Drupal has a fairly straightforward menu system. You can create any number of menus, and pages within a menu can be nested arbitrarily deep. When you create a node or a view, you can add easily add it to a menu. Module-generated pages (e.g. site contact forms) can be added to a menu manually.

Every menus is automatically available as a block. While some themes automatically display the "Primary links" menu, in most cases you need to go to the block administration page and place the block corresponding to the primary menu in the appropriate "menu" region for your theme. How the menu block is displayed varies by theme, but modules like Superfish can provide a dropdown display for menus in order to easily handle nested menu items.

2.5.8 Users

Even small sites that do not accept user-contributed material will likely have multiple users. For security reasons, each user should have their own account, even short-term research assistants.

By default, Drupal creates a page for each user that includes their username and how long their account has existed. By default, users can upload an avatar (small picture) that will display along with their username on content they've created using the default
"Article" content type, though this is entirely configurable and removable if desired. As with nodes and taxonomy terms, you can add any number of fields to user profiles, including fields that only users with administrative privileges-- and perhaps not even the user himself-- can see and edit.

In a university context, users may expect to be able to log in using their pre-existing campus username and password. Modules like LDAP and Shibboleth Authentication (which requires additional configuration at the server level) make this possible, and they can also be configured to pull in data from your campus directory, such as a person’s full name (good for populating a "Full name" field, if you’ve added that to your user profiles) and information about whether the person is faculty, student or staff, which may be useful for managing their level of access to the site using roles (see below).

2.5.9 Roles

Roles are a key component of Drupal’s permission system, providing differentiated levels of access to your site (including, but not limited to, administrative functionality.)

The user account that you create during the Drupal installation process is referred to as "user 1". This account automatically receives irrevocable full access and permissions to absolutely everything on the site. Therefore, the login information for that account should be closely guarded, and it should be rarely used. If there are users who should have full administrative access to the site, it’s better to give them their own user account with the “administrator” role.

When you install Drupal using the "standard" install option, Drupal will create a role called "administrator", that automatically gets full permissions to core functionality, and any new module that's installed. The administrator role also has permissions to modify permissions for its own role, and all other roles. Only skilled and trusted Drupal site builders should have the administrator role assigned to their user account. In a situation where multiple people with different skill sets are collaborating on a project using Drupal, it’s best to set up different roles where the permissions are tailored to what kind of work the person is doing, in order to reduce the risk that a user could "break something". An undergraduate assistant, for instance, might have permission to create and save blog posts, but not publish them publicly until a graduate student assistant reviews the content. There might be a role corresponding to the work of a particular research assistant who has been exclusively tasked with uploading and adding metadata to images. User accounts can have multiple roles, and the permissions the user has are cumulative. The Drupal permissions system is remarkably granular; almost every module provides its own permission configuration options (so you could allow a research assistant to administer the configuration of one module and not another), and

---

4 https://www.drupal.org/project/ldap
5 https://www.drupal.org/project/shib_auth
modules like Field Permissions⁶ can restrict what user roles can edit which fields in nodes, taxonomies, file types, and user profiles.

2.5.10 Summary of Drupal components

Drupal's content type system allows you to create highly customizable templates for storing and managing your content: content types, file types, and user profiles. These templates can include the option of using taxonomy terms to categorize your content, and taxonomy terms can have their own customizable templates for storing related metadata. Each piece of content you create using a content type is called a node. Views is a module that provides a user interface for querying your data and displaying the results in a wide variety of ways, including blocks—small boxes of data that can be placed any of the regions (header, sidebar, etc.) of your webpage defined by your theme. The menu system provides users with a way of navigating site content. Users can optionally have customized profiles, and a user's roles define their level of access within Drupal's granular permission system.

2.6 Overview of site building

The process of developing a project using Drupal can be broken down into a set of phases, generally executed sequentially, but usually with some degree of revisiting previous stages as the project progresses. An analogy with a face may be helpful here.

2.6.1 Deciding on Drupal

In the analogy, decide whether you want to create a face.

Decide if Drupal is a good candidate platform (see chapter 1). Create a list of all the things you want your site to be able to do (e.g. search content; filter search results by X, Y, and Z; display maps of all painting locations; allow users to sign up for accounts without project staff intervention; restrict access to data types A and B). Nothing is too small or obvious to exclude from this list. It may be helpful to organize these by priority. What are the features your site absolutely must have? Which would be good to have, and which are optional, depending on time and resources?

Do some research on modules and techniques (e.g. Views configuration) that can help you accomplish each feature on your checklist. You don't have to have the details of a solution worked out for each feature, but you should get a sense for whether the different features are doable given the modules that currently exist, or whether you may need to budget for some custom development. Alternately, your investigation may reveal that Drupal is not the right platform for the project, and you should explore other options.

Finally, decide whether or not to move ahead with Drupal. Can Drupal do most or all of what you want on your list of site features? How does it compare to other platforms? How do the costs (of time, if not money) of building the project in Drupal compare to alternatives? If you've decided to move ahead with Drupal, continue.

---

⁶ https://www.drupal.org/project/field_permissions
2.6.2 Basic configuration
This phase involves the creation of blood vessels, skin, bones, muscle. Even though some of them will not be visible in the end result, these structures are necessary though not unique to faces.

Install Drupal and the modules essential for any site (see chapter 4). Move on to the next step.

2.6.3 Data modeling
In this phase, you decide on what facial structures your face will include. Will it be a human face, or that of a cat? Or will you combine a dragon’s snout with a koala’s eyes and a chinchilla’s fur? Depending on your choices, you will probably need to create some additional fur, eye lenses, or skin, bones or muscle.

Develop a data model for the content you want to store on the site. Depending on your data model, you will probably need to install more modules (e.g. to be able to store URLs, pointers to other content, etc.) Move on to the next step.

2.6.4 Implementing the data model
Create the constituent parts of the face: eyes, nose, mouth, fur, etc. Refine the sculptural details on each as needed. At this point, you have all the components of the face, but they’re just jumbled on the underlying skull -- not arranged in any particular order.

Create the content types, vocabularies, user profiles, and file types for your site. Configure the data input form and add content.

2.6.5 Improving the display
Arrange the parts of the face in their correct positions relative to one another.

Configure the display of your content types, user profiles, and file types. Create the necessary Views for your site. Create menus.

2.6.6 Theming
Apply color to the the structurally complete but gray face you have created.

Install and configure a theme for your site. Arrange the blocks you have created within that theme.

2.6.7 Pre-launch configuration
Make sure that all the moving parts of the face (eyes, mouth, etc.) move smoothly and correctly.

Configure site search. Install anti-spam modules. Double-check permissions for anonymous users and each of the roles on the site. Write documentation.
3. Installing Drupal

3.1 Pantheon
We will be using the Pantheon hosting platform (http://pantheon.io), which provides free development hosting for Drupal and WordPress sites. Your site will have an ugly URL, and you can’t map a real URL to it without a paid plan with Pantheon, which is rather expensive (starting at $25/month). It’s better to develop a site on Pantheon, then move it somewhere else (your institution, Reclaim Hosting, Bluehost, etc.) when you’re ready to launch it.

Go to Pantheon and create an account, then add a new site. Choose Drupal 7. Once your site has been created, follow the steps below.

3.2 Installing Drupal

3.2.1 Choose a profile
When installing Drupal outside of Pantheon, there will be two configuration profiles: “Standard” and “Minimal”. Pantheon has its own configuration profile that’s based on “Standard”, but includes some additional configuration to integrate the site with Pantheon’s hosting infrastructure. If you move your site off Pantheon, there are some Pantheon-specific modules you’ll need to disable (if you’ve enabled them), but otherwise you shouldn’t have to make major changes to your site, even if it was originally set up using the Pantheon configuration profile.

Choose the Pantheon profile (the only one provided), and hit the “Save and continue” button.

3.2.2 Choose language
Drupal installs with an English-language administrative user interface by default. If you want the core administrative interface to be in another language, click on the “Learn how to install Drupal in other languages” link and follow the instructions there. The completion status for the translation of the core Drupal interface varies by language; an overview is available at http://localize.drupal.org.

Hit the “Save and continue” button.

3.2.3 Verify requirements, set up database, install profile
Because Pantheon hosting is pre-configured to support Drupal sites, the process for installing Drupal on Pantheon immediately skips through these three steps.

3.2.4 Configure site
The last step in the Drupal configuration process allows you to specify the name of the site, the username and password for the “user1” account (which will always have full permissions to all administrative functionality), the email address from which the site will send mail (e.g. new passwords, notifications, etc.), and how the site will handle update notification. All these things can be changed later through Drupal’s administrative interface.
3.2.4.1 Site information
This section includes the site title and the site email address.

Site name
The site name will appear as part of the HTML title for every page on the site. For example, if someone were to bookmark an “About” page on your site, the bookmark would, by default, be saved as “About | Your Site Title”, with whatever value you put in for your site title. Most themes (site designs) display the site title in large lettering towards the top of every page, though you can turn off (for instance, if you want to have a graphic banner image with your site name, instead of just text).

To change the site name later, go to /admin/config/system/site-information, or Configuration > System > Site information using the administration menu module.

Site email address
The site email address is the email address that the site uses to send out emails such as password changes, notifications (e.g. of new content), etc. To minimize the risk of those emails being caught in a spam filter, it’s best to use an email address that has the same domain name as your site. If you’ve registered your own domain name (e.g. myproject.org), check the documentation provided by your domain registrar or hosting provider for creating an email address that uses the same domain name. While you could put in an email address that doesn’t actually exist (i.e. that doesn’t forward to an email address that someone checks, and isn’t connected to its own inbox), it’s best to use an email address where you’ll be able to receive replies from someone who, for instance, responds to a user account creation email with a question about the site.

To change the site email address later, go to /admin/config/system/site-information, or Configuration > System > Site information using the administration menu module.

3.2.4.2 Site maintenance account
This section allows you to create the account often known in Drupal forums as “user 1”. This account, the first user account created on a site, has complete, unrevocable access to all administrative functionality on the site. You should create the “user 1” account with generic information using “admin” (or something similar) as the user name, and an email address affiliated with the site itself rather than any individual person. Each user’s email address must be unique, so don’t use an email address you’ll want to use again for the account belonging to a specific person.

For the security of your site, and for better tracking who’s doing what, it’s best for every user of your site to have their own individual account. It’s much easier to remember to deactivate or remove permissions from a research assistant’s own account at the end of their time with the project than to remember to change the password for your “user 1” account.

Create your “user 1” account, and use that username and password to log in just long enough to create an account for yourself with administrative privileges. Record the information about the “user 1” account somewhere safe (and not accessible by anyone other than your most trusted partners on the project), log out, and log in with your own account from then on.
You can change the information you set up here for your “user 1” account in the future by going to /user/1/edit.

3.2.4.3 Server settings
If your primary audience will be from a specific country, choose that country from the drop-down; otherwise, you can leave it as “none”. If you choose a default country, Drupal will set the correct first day of the week (i.e. whether weeks should begin on Sunday or Monday when displayed on calendars).

Drupal requires you to set a default time zone, though you can configure the site so that individual users can choose their own time zones.

Both of these settings can be changed later by going to /admin/config/regional/settings, or Configuration > Regional and language > Regional settings using the administration menu module.

3.2.5 Update notifications
You should have the “Check for notifications automatically” checkbox enabled. This will display a notice when a module you’ve installed (or Drupal core itself) needs to be updated. It will also add an administrative option that will allow you to install new modules using the Drupal administrative interface. If you don’t enable this checkbox, you can have the same effect later by enabling the “Update manager” module on the modules page (/admin/modules).

You might want to have “Receive email notifications” enabled. This will send a notification to the email address used by the “user 1” account when there are modules that need updates. If you’re working on the site frequently, you’ll catch these updates through the notifications on the site. Once you’re working on the site less frequently, you might want to enable these email notifications. You can change this setting, as well as configure the frequency for checking updates and the list of emails that should receive the notification, afterwards by going to /admin/reports/updates/settings, or Reports > Available updates > Settings.

4. Modules
4.1 Overview

Once you’ve successfully installed Drupal, you’ll be shown a blue screen with some suggestions about what to do next, with a black and gray administration bar at the top of the page. Configuring Drupal will be much easier if you immediately install and enable a number of essential modules (code packages that provide functionality beyond Drupal's core features, similar to WordPress plugins) that all or almost all sites need. Once you've installed these modules, you can proceed with developing content types, highly customizable templates for storing your data (as discussed in Chapter 5 and beyond).

Sections 4.1 and 4.2 cover how to go about looking for modules that fit your project’s needs, and how to evaluate whether the module is sufficiently reliable to be worth incorporating into your project. These are important considerations as you develop your own site. The following chapters, which discuss in depth how to build the example site, provide a list of recommended
modules, which may make sections 4.1 and 4.2 less urgent. To move directly to the list of essential modules and how to install and enable them, begin with section 4.3.

4.2 Finding modules

A visit to the drupal.org Modules page\textsuperscript{7} will show that there are over 15,000 modules that you could download and install. A variety of filters (maintenance and development status, category, core compatibility, and others) offer some options for narrowing down this list. Choosing only core compatibility with Drupal 7.x-- thereby limiting the list to only those modules that you can install using the version of Drupal described in this book-- still results in a list of nearly 10,000 modules. With so many options, how can you decide which to install?

While drupal.org should be considered the authoritative place for downloading modules, it’s far from the best place to find modules. Even if you know the name of a module, it may not turn up in the first page of results if you search for it on drupal.org; it is much more efficient to search for Drupal module Your-Module-Name on Google or some other search engine.

The modules described in this book are a good place to start when looking for modules that address common needs for scholarly Drupal sites.

If you encounter a situation not addressed in this book, a general-purpose search engine will again be a better resource than drupal.org for trying to identify useful modules. For instance, a Google search for drupal module tag cloud will pull up links to multiple modules on drupal.org that you can then assess and compare. It will also pull up a few tutorials and forum discussions, for instance, about how to use the Views module to accomplish this task. Even vaguer queries, such as drupal better taxonomy select, generally turn up useful results.

4.3 Assessing modules

While it may not seem like it, adding a module to your site is in many cases a serious commitment. If you use a module that provides a type of field that you’re going to use, the data that you store in the corresponding fields is going to depend on the module maintainers. Bugs in the module, or in future updates to the module, may lead to data loss. Also, if a module for a field doesn’t provide integration with Views (for displaying the data) and/or Feeds (for importing data in bulk), but the developers promise that integration will be available soon, your project development may have to be put on hold until it’s ready -- and it may never materialize.

Modules which provide additional display options for Views (for data entry, export or visualization) or data-entry widgets for fields may seem less important than modules that provide field types. If you have to uninstall such a module, you wouldn’t lose any data. What you might lose, however, is a good deal of time spent training student assistants in how to do data entry. You would have to figure out an alternative approach to accomplishing the tasks that those modules were addressing, which may not be easily done. You would have to rewrite your site documentation.

These worst-case scenarios should not deter you from installing modules; Drupal sites are supposed to have modules, and usually, many of them. They do, however, highlight the

\textsuperscript{7} https://www.drupal.org/project/project_module
importance of conducting some assessment of modules, as described below, before you install them, and certainly before you come to depend on them.

Even as a non-programmer, you shouldn’t see Drupal as software that gets written far away, and you have to live with the consequences of whatever changes are, or aren’t, made. Drupal is an open source project, and many module developers are people with entirely separate jobs, who volunteer to write modules in their spare time. If progress isn’t being made on an issue as fast as you would like, you can often get in touch with the developer to inquire about it. Maybe the developer would be interested in working with someone at your university as a co-maintainer, or perhaps you can write the developer into your next grant proposal to fund some of the work you need done. See (future chapter) for more on getting involved with the Drupal community.

4.3.1 Modules published on drupal.org vs. those published elsewhere

As a general rule, you should only install modules that you have downloaded from drupal.org. For a module to be listed on drupal.org, it has to meet the Drupal coding standards (which require, among other things, that the code be well-commented to make it easier for other programmers to understand), pass a series of automated tests, and be free of identified security holes (which could put your site in jeopardy of being hacked). While the degree to which modules are actively maintained does vary, if a security hole is found in a module, the module will be removed from drupal.org if it is not updated to address the problem in a timely manner.

Modules generated by the Features module (which allows you to package up site configurations-- such as content types and Views-- and move them between sites as a module) are generally not listed on drupal.org, though they may be sub-modules of module packages that provide some other functionality. For instance, the Feeds module includes a Feature-based module that provides a sample importer. Stand-alone modules generated by Features can be shared using a “Features Server”, but in many cases, people use Github to publish their Features. Installing a Feature module from Github comes with some risk (there’s no easy way to ensure that the reliable code exported from the source site hasn’t been tampered with) but it may not be as risky as downloading a module that purports to add some new functionality to the site.

There are many modules that are published on Github, or on project websites. In some cases, the developers don’t feel like it’s worth publishing the module on drupal.org because it’s designed for a niche audience. In other cases, publishing the module on Github is a way to make a module available while it is waiting to be reviewed on drupal.org. Many well-known Drupal modules that have been developed for digital humanities are not distributed on drupal.org. For modules that aren’t distributed on drupal.org, it’s harder to get information on how many people have the module installed, and it may be more difficult to find out how well the module works, depending on whether there’s a pointer to an active issue queue. There’s also no guarantee that the code meets Drupal coding standards. Nonetheless, at least for modules developed within the digital humanities community, you may be able to alleviate your concerns

8 https://www.drupal.org/coding-standards
9 https://www.drupal.org/project/features
10 https://www.drupal.org/project/feeds
11 See, for instance, from Stanford: https://drupalfeatures.stanford.edu/
12 These include TEICHI for displaying TEI (http://www.teichi.org/downloads), Scripto for crowdsourced transcription (http://scripto.org/download/), and Islandora, which integrates Drupal with Fedora (https://github.com/Islandora).
through typical communication channels (e.g. inquiring on Twitter or DH Answers\(^\text{13}\)), or even asking the developers directly. Particularly since these modules are designed specifically to address digital humanities use cases, they should not be considered off-limits due to the way they are distributed, but a little extra caution is beneficial.

4.3.2 Module versions
Once you’ve found a module that looks like it can address a need on your site, and you’re looking at the module’s page on drupal.org, the first thing you should do is check to see if there’s a version of the module compatible with your version of Drupal. In this book, we are using Drupal 7, so you need to look for a module version that starts with “7.x”. “7.x” is shorthand for “any version of Drupal 7”. You should be running the most recent version of Drupal 7, which will include all the latest security patches. For most modules, it doesn’t matter exactly what version of Drupal 7 you’re running, but modules aren’t backwards compatible, and there’s no way you can run a 6.x-version module on a Drupal 7 site.

To see the versions of the module that are currently available, scroll down to the bottom of the module’s page on drupal.org. The last section in the main column is labeled “Downloads”. Most modules have at least a green table (“Recommended releases”) and a red table (“Development releases”):

![Figure 4.1 The download section of a module page](image)

Section 4.5 will cover the steps to take when you actually want to download and install a module; here, it’s important just to note that there is a version 7.x-1.4 available as a recommended release. Other modules might only have a development release; others might only have an “other release” (which appears in a yellow table), for instance, for a beta version of the module that is more stable than a development version, but not necessarily ready to be released as a recommended version.

For modules where no 7.x version is listed under “Downloads” at all, it’s possible that users other than the module’s developer may have put together a Drupal 7 version. Look in the issue queue for an issue called something like “D7 port of [module name]”, and there may be code in that comment thread that you can use on your site. A word of warning, however: while such unofficial ports may be usable, they tend to rate poorly on the other sustainability factors

\(^{13}\) [http://digitalhumanities.org/answers/]
described below, especially if they’re not taken up by the module’s developers and provided as a supported version.

4.3.3 Number of users

With open source software like Drupal, the number of users of the software is an important metric. If many sites have a module installed, that indicates that the module works, and does what it claims to do. If a module has a large user base, more people are keeping an eye on the module and odds are better that bugs will be reported in a timely manner. Some of those users may themselves be Drupal developers, and might volunteer to fix bugs as they arise, if the original module developer can’t keep up with the bug reports. When it comes time for a major version upgrade (e.g. Drupal 7 to Drupal 8), modules with a lot of users are more likely to be ported to the new version of Drupal in a timely manner.

The number of sites that report using a module is listed under “Project information” (directly above the “Downloads” section on the module page), under the heading “reported installs”. If you click “View usage statistics”, you can see a chart for how those numbers have changed over time, for each release of the module.

![Weekly project usage chart](image)

Figure 4.2 Usage statistics for Date module.

How many users counts as “many” varies by module. Views (which allows you to display your data in a variety of ways) has over 850,000 sites reporting its usage. Link (which provides a field for storing URL data) has over 300,000 sites reporting its usage. Both of these are important for a wide variety of Drupal sites, across all sectors. The CAS module[^14], by comparison, is only used by 10,000 sites; this module provides a way for users to log into a Drupal site that’s compatible with many campus authentication systems. It’s still important for university-developed Drupal sites of various kinds (departmental websites, organizational websites, and research websites), but is not generally relevant for businesses that use Drupal. The Partial Date module[^15] provides a field for storing date information of different granularities. For example, if you have a precise date for one document you want to include in your site, and you only know a month and year for another, the Partial Date module can accommodate that

[^14]: https://www.drupal.org/project/cas
[^15]: https://www.drupal.org/project/partial_date
better than the Date module. Only around 900 sites report using Partial Date, but since it solves a fairly specialized problem, that counts as a solid number of users.

In general, if a module has fewer than 500 users, you should look carefully at the other factors for module selection before incorporating it into your project, because the small user community can be a liability.

4.3.4 Maintenance and development status
The maintenance and development status (directly above the “Downloads” section on the project page) is self-reported by module developers and may not be current, though the “last modified” date gives you some sense of its currency. Nonetheless, this information can give you some sense of their perception of the module. The full explanation of all the possible taxonomy terms for both maintenance and development status is available on drupal.org\(^\text{16}\), but “Actively maintained” is the most promising maintenance status for the longevity of a module. As long as there’s a recommended, non-beta release available for the module, “minimally maintained” is generally fine. “Seeking new maintainer” indicates that the original developer no longer wishes to be responsible for the module, but a new developer has not stepped into the role of module maintainer— a warning sign for module sustainability.

---

**Project Information**

- Maintenance status: Actively maintained
- Development status: Under active development
- Reported installs: 451,065 sites currently report using this module. View usage statistics.
- Downloads: 2,738,075
- Automated tests: Enabled
- Last modified: October 24, 2014

*Figure 4.3 Project information for Date module*

Drupal project pages include information that allow you to check whether the self-reported information for maintenance and development status is accurate. If a module is said to be under active development, what was the last date a version of the module (including development releases) was released? You can check this by looking at the dates listed next to each version under “Downloads”. If a module is said to be actively maintained, how quickly are bug reports and issues addressed? You can check on bug reports and issues in the right sidebar of the module’s page, towards the top of the page. The “statistics” section of this sidebar has some information (including a small graph with the number of open bugs over time), but for modules that aren’t particularly active, there may be no data under “statistics” at all. In those cases, it’s better to click on “open bugs” and read through the posts yourself, and see whether the issues are being addressed by the developer or others in a timely manner.

\(^{16}\) See https://www.drupal.org/node/1066982
As shown in figure 4.4, a very widely used and well supported module like Date can have thousands of issues, including over a thousand open issues. The number of issues by itself shouldn’t necessarily lead you to question the sustainability of a module, if other factors are positive. More important than the number of issues is the way that open issues are handled. Does anyone respond? Are confirmed issues (those that others can recreate) addressed? These are things that require you to read through the issue queue yourself to assess.

In some cases, users who are themselves developers write patches for bugs and post them in the issue thread; it’s up to the developers to incorporate those patches into new releases. Seeing threads with a patch that multiple users confirm fixes the problem, but where no developer has stepped in to announce that the patch will be included in a new release, is both a good and a bad sign: good, insofar as there’s an active community of users, but bad for the official maintenance of a module.

4.3.5 Release notes
If you click on the “Notes” link next to any release, you’ll see some information about it. For non-development releases (as well as many development releases), there’ll be a list of bugs that the release fixes, as well as links to the page for each bug. There are cases when it may be best to install a development version of a module, and it’s important to read the release notes first to make sure there aren’t any warnings against using that version of the code.

4.4 Essential modules

The trend in Drupal development is to include more and more universal or near-universal features as part of Drupal core, in order to reduce the number of modules that must be
maintained (and updated) separately. Nonetheless, the following modules are worth installing immediately, on any kind of site, before you move on to other configuration.

- Administration menu\(^{17}\) - replaces the default menu bar of administration options with a drop-down version that gives you access to any administration page, from anywhere on the site. Instructions in this book for how to access administrative pages are given with navigating the administration menu in mind, though this largely mirrors the internal structure of how administration pages are organized. To enable this module after you’ve installed it (see section 4.4), be sure to enable both "Administration menu" and "Administration menu toolbar style", and disable "Toolbar", which is listed under the "Core" section.

- Advanced help\(^{18}\) - allows you to see help documents provided by other modules that take advantage of this module’s advanced way of handling help files. This module gives developers a way to write more extensive help documents than are normally possible in Drupal, but to use these documents the module has to be enabled. Help documents often make it easier for you to figure out how to configure a module.

- Backup and Migrate\(^{19}\) - a good module to install, even if you’re not facing any impending migrations. This module allows you to easily download your entire database, and restore from a previously downloaded version if your configuration efforts go awry. It also lets you configure automatic, periodic backups.

- Chaos Tool Suite (ctools)\(^{20}\) - required to install Views.

- Views\(^{21}\) - allows you to create highly custom displays of your data, including lists, galleries and table, as well as slideshows, timelines and maps (with support from additional modules).

- Module Filter\(^{22}\) - modifies the Module page interface to make it easy to find modules by searching for module names and filtering them by whether they are enabled, disabled, required, or unavailable. It also replaces the module-enabling checkbox with an on/off toggle. Without this module, there is no way to search for a module by using your browser’s search function (e.g. Ctrl-F) since the module name appears everywhere the module is required by another module. Once your list of modules gets beyond a certain size, this module becomes essential.

- Token\(^{23}\) - required to install Pathauto; provides access to small bits of text (e.g. the value of a particular field) in other contexts.

- Pathauto\(^{24}\) - lets you define patterns for nicer-looking URLs.

### 4.5 Installing and enabling modules

Once you’ve identified a module you’d like to add to your site, you need to install it (by adding the code for the module to your Drupal installation) and enable it (by checking the box or boxes corresponding to the module on the Drupal Modules page.) There are two ways to install modules: using the Drupal interface, or by accessing the filesystem directly. Using the Drupal

\(^{17}\) https://www.drupal.org/project/admin_menu

\(^{18}\) https://www.drupal.org/project/advanced_help

\(^{19}\) https://www.drupal.org/project/backup_migrate

\(^{20}\) https://www.drupal.org/project/ctools

\(^{21}\) https://www.drupal.org/project/views

\(^{22}\) https://www.drupal.org/project/module_filter

\(^{23}\) https://www.drupal.org/project/token

\(^{24}\) https://www.drupal.org/project/pathauto
interface is easiest, but may not be compatible with your hosting environment. Even if you primarily use the Drupal interface to install modules, you should be comfortable accessing the filesystem because some modules require that you upload additional files into specific directories, which can’t be done through the interface. Also, while you can update most modules through the Drupal interface, you have to do Drupal core updates via the filesystem.

If you're using university hosting for your Drupal site, particularly if you’re using some sort of "Drupal hosting service", you may not have permission to install new modules. In this sort of arrangement, the Drupal support team has usually chosen a handful of common modules to make available to users, and they'll do the work of updating those modules for you. Hopefully there is significant overlap between the available modules and at least those listed below as essential; if not, the hosting service is probably not be suitable for developing a digital humanities project site, and you may need to explore other options.

4.5.1 Installing modules using the Drupal interface

Click on "Modules", in the black menu bar at the top of the screen. Right above the list of modules with checkboxes, there should be an option for "Install new module"; click on it. If you don’t see this option, scroll down through the “Core” modules, check “Update manager”, and save; the “Install new module” option should appear.

In another tab, go to the page of the module you want to install on drupal.org, and copy the URL of one of the download links (either .zip or .tar.gz). You do not need to download the module, just copy the download link.
Back on your site, paste the download link into the “Install from a URL” field, and click on the “Install” button.

You can find modules and themes on drupal.org. The following file extensions are supported: zip tar tgz gz bz2.  

Install from a URL
For example: http://ftp.drupal.org/files/projects/name.tar.gz

Or
Upload a module or theme archive to install
Choose File No file chosen
For example: name.tar.gz from your local computer

Install

Figure 4.6 Pasting a URL into the “Install from a URL” field

A progress bar will appear, and once it’s complete, you should see a message indicating that the module was installed successfully.

Installation was completed successfully.

date
- Installed date successfully

Next steps
- Install another module
- Enable newly added modules
- Administration pages

Figure 4.7 Successful module installation

A number of options will appear under this message:

- Install another module: this will return you to the screen where you can paste a URL for another module
- Enable newly added modules: this does not, in fact, enable the modules. All it does is take you back to the modules page, where you can enable the modules yourself by checking their boxes. It may seem counterintuitive, but each module package you install
may have multiple sub-modules, and you might not need to enable all of them. This gives you the choice of which sub-modules to enable.

- **Administration pages:** this takes you to a page with a list of top-level administrative functions. After you’ve installed the Administration Menu module, you shouldn’t need to access this page, because all administrative functions will always be available through the Administration Menu toolbar.

If you have more modules to install, choose “Install another module”; otherwise, choose “Enable newly added modules”.

### 4.5.2 Installing modules by accessing the filesystem

Go to the module's page and download the recommend release that corresponds to your version of Drupal (should be highlighted in green, and start with “7.x”). There's no fundamental difference between the .zip and the .tar.gz version, but one might be easier for your computer to unzip than the other. On Windows, choose .zip; on Mac and Linux, either works. (*Note:* sometimes there are good reasons to download versions other than the recommended ones; we will discuss these in class.)

Once you've downloaded the module file, unzip it. You'll get a folder with the same name as the file you downloaded, without the version number (e.g. token-7.x-1.5.tar.gz will give you a folder named "token").

Navigate to your Drupal filesystem; in most cases, this will involve launching an (S)FTP client {see appendix A for instructions on how to install, configure and use an SFTP client}, or if you've installed Drupal on your own computer, you simply need to find the right folder on your hard drive. The Drupal filesystem directory has folders in it including "includes", "misc", and "modules", as well as other files like index.php, install.php and update.php.

*Do not* move the module folder into the folder called "modules". This is an easy mistake to make, and it will significantly complicate the process of updating Drupal's core files. Instead, go to the "sites" folder, then go into "all" and create a new folder called "modules". Move the module folder into the sites/all/modules folder you just created.

Once the module files are in place, load (or reload) the Modules page (click on it in the default toolbar, or using the Administration Menu toolbar) and the module and any sub-modules should appear on the page.

#### 4.5.2.1 Why sites/all/modules?

Putting "contrib" modules (modules that have been developed by the Drupal community, rather than those that are part of Drupal's core) in sites/all/modules is considered best practice. There's also a practical reason: if you later decide to do a multi-site setup, where you're running multiple Drupal sites using the same Drupal core code, the modules you've added will automatically be made available to every additional site, too. Not only does this save you time when building new Drupal sites (since you won't have to re-download every module you want to use), it also saves you time every time you update the modules-- you can update the code in sites/all/modules, and the latest improvements, security patches, etc. will be present on all sites.
Are there other options? When you're running a multi-site setup, sometimes you want only one site to have access to a particular module, or a particular module version (e.g. you may need to run an older version of a common module to ensure compatibility with a niche module that isn't updated very often, but you don't want all your sites to lag behind.) In those cases, you’d put the older version of the common module in sites/site-name-here/modules. When a Drupal site loads its modules, it checks the modules folder specific to that site first, and any modules there that are the same as modules in sites/all/modules override the versions in the shared folder.

4.5.3 Enabling modules

In the default interface, you can simply enable a module by clicking on “Modules” in either the default toolbar or the Administration Menu, checking the checkbox next to that module, and hitting the “Save configuration” button at the bottom of the screen.

If you’ve installed and enabled the Module Filter module, the interface is somewhat different. Checkboxes are replaced with on/off toggles. You can also search for a module using the search bar at the top, which is more effective than using the standard browser search. (The standard browser search will pull up all references to the module on the page, which may include numerous instances of the module as a dependency for other modules.) You can also use the vertical tabs on the left to filter modules by category, or by whether they have recently been disabled/enabled, or are new.

A word of caution: filters that you set, either through the checkboxes at the top or in the sidebar, stay enabled until you disable them. If you limited the module results to a particular category at some point in the past, and then later search for a module using Module Filter, the only results that will appear are those in the category you previously selected, potentially making it appear that the module you’re searching for doesn’t exist. If you’re surprised to be not getting any results, make sure that you’ve selected “All” at the top of the left menu, and all the horizontal checkboxes under the search bar are checked.
4.8 Module Filter Modules page interface

The Module Filter module makes the “Save configuration” button “sticky” as part of the left sidebar, so you don’t have to scroll to the bottom to use it. With Module Filter, when you enable a module using the toggle, its new status will appear in yellow and the module will be highlighted in green. This indicates that your changes have not yet been saved. Hitting the “Save configuration” button on the left will finalize the change.

4.9 Enabling the Statistics module using the Module Filter interface for the Modules page, before clicking the “Save configuration” button.

A considerable number of modules have dependencies: modules that must be enabled before that particular module can be enabled. If the checkbox (or toggle, using Module Filter) next to the module you want to enable is grayed out, check the list of dependencies (listed in in small print under the module description) for ones that are accompanied by bright red “missing” text in parentheses, and search for and install those modules. Sometimes, dependencies have maroon “disabled” text in parentheses next to their name. This means that the module has been installed, but not enabled yet. If there are no missing dependencies, but there are one or more dependencies that are listed as disabled, don’t worry about enabling the disabled dependencies manually. You should be able to check the checkbox to enable the module that has dependencies, and once you hit the “Save configuration” button, Drupal will offer to enable the dependencies for you.

A note of caution: the dependencies list on the module page is not necessarily a comprehensive list of all the components necessary for the module to work properly. The README.txt file contained in the module directory will provide all the information you need to get the module
working. As a rule of thumb, any module that has the Libraries API module\textsuperscript{25} as a dependency will require you to upload additional code via the file system.

4.5.4 Configuring modules

Many modules need to be configured after they are installed, in order to accomplish what you need. The location of a given module’s settings within the Administration Menu is often not intuitive; the fastest way to get to a module’s settings the first time is to find the module on the Modules page (click on “Modules” in the Administration Menu), and click the “Configure” link next to its name.

![Figure 4.8 Using the Module Filter module, a list of modules with the arrow indicating the “Configure” link](image)

When you click on the “Configure” link for a module, look at the breadcrumb at the top of the configuration page. This will give you a clue as to where you can find that page again using the Administration Menu. Figure 4.9 shows the configuration page for the Update Manager module. Disregard the first two parts of the breadcrumb path (“Home” and “Administration”, and the rest of the components will get you most of the way towards finding the page within the Administration Menu (in this case, it’s located under Reports > Available updates > Settings).

![Figure 4.9 The Update Manager configuration page. Components of the breadcrumb path relevant for finding it again using the Administration Menu are indicated.](image)

Every module has its own set of configuration settings; this book will cover how to configure individual modules as needed for site development in subsequent chapters. The Advanced Help module, which provides a new top-level item in the Administration Menu, may be of use when configuring modules, or figuring out why a module doesn’t work as expected (in some cases, you may have to upload an additional library via the filesystem). If you’re not sure how to configure a module, click on “Advanced Help”, click on the name of the module, and you should be able to access, at a minimum, the README.txt file that is included with the code. Some modules, such as Views, provide extensive documentation through the Advanced Help interface.

\textsuperscript{25} https://www.drupal.org/project/libraries
4.6 How many modules do you need?

Drupal sites vary significantly in the number of modules installed, and which modules those are. Newcomers to Drupal often face the temptation to install and enable any module they come across that they can imagine possibly needing, for the convenience of having that functionality available if a situation arises where it would be useful. For the sake of project sustainability (both with regard to the issues described in section 4.2, and with regard to the time necessary to maintain your site), it is important to resist this temptation and be more deliberate in your module selection.

Every time you enable a module, you are responsible for updating it -- if not with every release (some releases provide new features rather than important bug or security fixes), probably around 1-3 times per year. The more modules you have enabled, the more updates you will need to perform on a regular basis.

By default, Drupal checks for module updates periodically, and will notify you (with a yellow or red bar, depending on the importance of the updates) at the top of the module administration page when updates are available. By default, Drupal checks for updates daily, but you can configure this by going to Reports > Available updates > Settings using the administration menu. Updating modules isn't hard, but it can be a hassle if you have many modules enabled. The trade-off is worth it if the modules are providing useful functionality for your site, but not if they're enabled "just in case".

Furthermore, modules can sometimes interact with one another in unexpected and problematic ways. If you're trying to figure out why your site isn't working correctly, the fewer modules you have enabled, the easier it may be to troubleshoot. Problems are not always obvious immediately: sometimes a routine module update will cause a conflict with another module, where there hadn't been one before. Many modules create new tables in your database, and if you're using inexpensive commercial hosting for your site (particularly if you have a multi-site setup with multiple Drupal sites on the same account), there may be restrictions about the total number of database tables you can have as part of the account. Having many modules installed may also slow down your site's performance.

Don't hesitate to add modules to your site when it sounds like they might fill a need (especially if you've backed up your database before doing so, just in case enabling the module causes catastrophic problems-- which can sometimes happen). However, it's important to be clear about why you've enabled every module on your site. At least twice a year, look through the list of enabled modules and consider disabling-- or even uninstalling-- modules that you don't need anymore (e.g. modules to support theme development once you're done building your theme, modules that didn't end up doing what you'd hoped they would do, etc.) Module clutter can build up once a site has been running for some time and has undergone iterative improvements, and it's worth periodically keeping in check.

Finally, every module you install exacts a certain toll on the performance of your site. The precise impact of a module varies, but each module may add to the amount of processing your Drupal site will have to perform every time a page is loaded. There are ways to mitigate this impact, through performance optimization techniques, but the best policy is to run as lean a site as possible to begin with.
4.7 Disabling and uninstalling modules

4.7.1 Disabling
If there’s an enabled module on your site that you’re not using and don’t plan to use, you should at least disable it, by unchecking the box(es) corresponding to the module on the Modules page. Disabling a module means that you are no longer impacted by security vulnerabilities in its code, and you no longer need to update it. Disabling a module is a prerequisite for uninstalling it.

You can't disable a module if other modules that are still enabled depend on it. Also, if the module provides a field type (e.g. date, link, etc.) you can't disable it if fields of that type exist in your database. You must either delete those fields if you aren’t using them anymore (this should be the case, if you want to disable a module necessary for their continued existence), or convert the field type to something else (e.g. from a structured date field to text). There is currently no module that allows you to convert field types; if you need to do this, you might be able to find tips and tutorials by Googling for drupal convert [old field type] field to [new field type] (e.g. drupal convert link field to text). To see an overview of all the fields you have on your site, go to Reports > Field list using the Administration Menu; a link to this page is also available as part of the text for any field-providing module: “Required by: Drupal (Field type(s) in use - see Field list”).

4.7.2 Uninstalling
Disabling a module makes it no longer active on your site, but changes you’ve made to the module’s default configuration settings are still stored in the database. If you choose to re-enable the module later, those settings will generally return to the state you left them in. If you want to completely remove the module (and tidy up your database in the process), or if you want to start completely from scratch with the configuration for a particular module, you should uninstall the module after you've disabled it.

Using the Administration Menu, go to Modules > Uninstall, and check the box next to the module(s) you want to uninstall.

4.7.3 Removing the code from your site
Uninstalling a module removes all trace of the module from your database, but it does not remove the module code from your sites/all/modules directory. As a result, the module will still appear on the list of all modules (albeit disabled), unless you remove the code by accessing the filesystem directly; there is no way to remove the code through the Drupal UI.

Removing the module code is not necessary, but if you go through the process of disabling and uninstalling modules, it’s generally worth taking the extra step to clean up your sites/all/modules directory by deleting the module code. All you need to do is connect to the server, navigate to the correct directory, click on the module’s folder, hit “delete”, and confirm that you want to delete it.

Keep in mind that removing the module code should always be the last step; never remove module code before you have disabled and, ideally, uninstalled the module. When a module is enabled, Drupal will look for the module code, and the site will likely display errors if the module isn’t where Drupal expects it. If you accidentally remove the code for the wrong module, you can
fix the errors by re-installing the module via the filesystem (i.e. by downloading the code from Drupal.org, unzipping it, and uploading it to sites/all/directory.)

5. Content types and introduction to data modeling

5.1 Overview
This chapter will introduce the concept of "content types" in greater depth, and will provide a general discussion of factors to consider when doing data modeling for your project.

5.2 Introducing Content types
On a Drupal site, "content types" are how you store most of the site content, from small pieces of data that will never be shown to the end user in isolation (e.g. the name and geographic coordinates of a location that you want to refer to from another content type) to more complex entities with a lot of different properties (e.g. a project profile, including information about the developers, the funding agency, a description, website URL, links to relevant publications, etc.) to simple content that stands alone for visitors to the site (e.g. basic pages like "about the project", or blog posts). Creating content types in Drupal is a straightforward process; making good choices about how to break down the material you want to present on the site into content types, and how to structure those content types takes some careful deliberation.

One consequence of Drupal's extreme flexibility and customizability is that a Drupal site can easily become a convoluted mess—where it's not clear which modules, blocks, or views are generating any particular snippet of text on the screen, and it's not clear how one would add new data to the site, or why content is showing up in a particular order—in a way that's simply not possible with WordPress or Omeka. The choices made around content types are generally the primary culprit when a Drupal site descends into chaos, as the site developer adds more and more ad-hoc fixes to get the data to show up the ways it needs to.

If a project team includes both technically-oriented members and scholarly-oriented members who generally work separately, planning the content types is one occasion that necessitates a high degree of collaboration and communication—in person, if possible. If a developer makes assumptions about whether to make a date field mandatory, or whether a person's profile should have separate fields for given names and surnames, it can impact the kinds of scholarly arguments one can make with the data. Likewise, if a scholar doesn't make explicit their assumptions about what they'll be able to do with the data (e.g. "display all events that happened in the 19th century, and filter by person involved"), the content types may get structured in a way that doesn't support it. Restructuring content types can be time-consuming, even more so if data entry has already begun. Content types that are actively used will inevitably evolve to some extent, but investing work in getting the content types reasonably "right" for your project upfront will pay off considerably.
5.3 Drupal’s default content types: Basic Page and Article

If you’ve installed Drupal using the standard installation profile, your site will already have two content types: “Basic Page” and “Article”. Both will have a title field and a body field; “Article” additionally has fields for an image, and for tags. “Basic Page” and “Article” are essentially equivalent to the WordPress concepts of “Page” and “Post”, respectively.

If all the information that you want to capture about the content in your collection could be captured by those two default content types, that is a strong indicator that Drupal is not the right platform for your project. The payoff for the additional configuration work that Drupal requires is its ability to easily capture dates, connections between different types of data, locations, etc. in a structured form that can be leveraged in a variety of ways for display and navigation. If this is not necessary for your project, WordPress is the better platform choice.

There is nothing special about either of the default content types. They’re not protected in any way: you can delete them by going to Structure > Content types > Article > Delete, and you can add fields to them like any other content type (by going to Structure > Content types > Article > Manage fields or Structure > Content types > Basic page > Manage fields).

For our example project, we could potentially just use the default content types without making any changes. We could use “Basic Page” to store biographies of individual people, putting their name in the title field, and all the other information about them in the body field. “Article” could be used primarily to store images using the image field, with space for a title and description in the title and body fields, respectively, and a few descriptive tags.

Using the default content types in this way might be problematic for your project, even if the title/body field structure is generally a good fit. You might want to have an “about” page that provides an overview of your project, but if you’re also using the “Basic Page” to store information about people, you’ll encounter some difficulties when using the Views module to generate a display of all people. Informational pages will be mixed in with the biographies (since both use the “Basic Page” content type), and there’ll be no way for Drupal to differentiate the two kinds of content, even though you can do so easily as a human. For this reason, it’s best to use the default content types as intended— for generally static information (“Basic Page”) and blog posts, news, updates and the like (“Article”)— and create additional content types tailored specifically to your data, even if some of them look similar to “Basic Page” or “Article”, using the title and body fields.

In the case of our example project, storing the data using just a title and body field would drastically limit what we could do with it. There would be no way to connect an image of a person to the biography of that person. There would be no way to generate a map or timeline view of the events in the person’s life, because that information would just be stored as part of a big text field. There would be no way to sort people alphabetically by last name (unless we indicated that names must be entered into the title field, last name first). Finding all people affiliated with Howard University would be no easier than if we were working on paper.

Not all data needs to be stored in a structured way: for every project, it’s important to consider what kinds of visualizations and research questions your site will support, and how much additional data entry work would be required for each new aspect of the data you’re considering encoding. Even the small amount of time it takes to put information
about a person’s birth date into a specific date field can add up over tens or hundreds of individuals. Multiply this by the number of specific fields you wish to include in a content type, and it quickly becomes evident why a project team should carefully identify which fields are truly important for the research questions or displays they intend to support, and which would fail to provide a payoff commensurate with the data entry work required.

5.4 Data modeling for content types

We’ve concluded that for our example site we need more -- and more elaborate -- content types than Drupal provides by default. But how can you determine how many content types you actually need, or what fields they should have?

For most projects and data sets, there is no single correct data model: it all depends on what you plan to do with the data, and how you anticipate the project might evolve. Some degree of guesswork is needed, and you'll likely adjust your content types over time, but thinking through the following considerations before deciding on your initial set of content types should reduce the number of changes you'll have to make later.

5.4.1 Data vs metadata

For sites that will be storing data, a rule of thumb is to construct one content type for each kind of data, and use fields within those content types for metadata (information about the data.) Two projects based on the same data set may have different perspectives of what counts as data, and what counts as metadata, depending on how they want to present the content.

Suppose your data set includes the following:

- Full text of an interview, which includes references to important places
- Name of the interviewee
- Age of the interviewee
- Birth place of the interviewee

How many content types should you create for this data? It all depends on the project goals and focus, and how you can see the project expanding.

If the focus is primarily on the text of the interview, perhaps one content type ("Interview") is enough. It can contain some fields to store the information about the interviewee. (The choice of which kinds of fields is a more complicated one, and is discussed further below.)

If the people being interviewed are themselves a potential focus-- of equal importance to the interview text-- you might want two content types: "Interview" and "Person". "Person" would contain the information about the person (name, age, birth place), and
"Interview" would contain the text of the interview, and a node reference field pointing to the person being interviewed.

Technically, not much would change if you included the node reference field as part of the "Person" content type instead, and used it to point to the interview, but doing it that way feels a little less intuitive. There's something incomplete about an "Interview" content type that doesn't store information about who's being interviewed (even though you can use the Drupal Views module to call up that information, if it's stored as part of the "Person" content type), but information about interviews is not an essential part of a stand-alone "Person" content type.

What if the focus of the project has more to do with locations? In addition to "Interview" (and "Person", if you're splitting that into its own content type), you would want to create a "Location" content type. You would then use a node reference field to store the person's birth place, and you might add a (possibly multi-valued) field to "Interview" to capture the important place or places mentioned in the interview text.

5.4.2 Content type vs. taxonomy

The taxonomy system in Drupal 7 (but not previous versions) has some characteristics in common with content types. The simplest use of taxonomies involves creating different vocabularies, and either pre-populating them with a set of terms (which you may choose to organize hierarchically) or using them to store user-generated tags. By default, terms in any vocabulary have a name (the term itself) and optionally a description, but you can add fields on a vocabulary-by-vocabulary basis, which will then be available to all terms in that vocabulary (go to Structure > Taxonomy > Your-taxonomy-name > Manage fields to see the interface, which is nearly identical to the "manage fields" interface for content types).

The ability to add specialized fields to vocabularies can be useful. A project that invites users to list their institutional affiliation as part of their user profile could use a free-tagging a vocabulary for institutions in the user profile, and have a student assistant augment every term added with the geographic coordinates of the institution, in order to generate a user map.

On the other hand, it does blur the line between content type and taxonomy, which complicates the process of data modeling. As a rule of thumb, if you're considering adding more than two fields to a vocabulary, think about why you're not creating a content type for it. If you're still torn between content type and taxonomy, here's a few more factors to consider:

- The content of a node reference field (which you can use to point to content stored in a content type) appears as a link to that single piece of content. For example, if you have a "Location" content type, and use a node reference field
in a “Person” content type for the person’s birth place, clicking on the person’s birth place will take you to the corresponding “Location” node.

- The content of a term reference field (which you use to associate a taxonomy term with a piece of content) appears as a link to a page that primarily shows a list of all the content that’s been assigned that taxonomy term. The definition, and/or any other fields you’ve associated with the vocabulary, appear at the top. Drupal also automatically creates an RSS feed for content tagged with each taxonomy term. Therefore, if you use a vocabulary to store locations, clicking on the person’s birth place will give you a list of all people with that birth place, and any other content that’s been tagged with the same location, such as interviews where that location is important.

- Editing taxonomy terms (or adding them, outside the context of adding a new tag when creating content) requires a different set of user permissions than creating/editing content via content types. Drupal doesn't automatically have the fine-grained permission control for taxonomies that it does for content. Where Drupal differentiates "edit own content in Content Type X", "edit any content in Content Type X", "delete own content in Content Type X" and "delete any content in Content Type X", there's an all-purpose "Edit terms in Vocabulary X" and "Delete terms in Vocabulary X". Depending on who’s going to be entering the data, you might want the additional control offered by a content type.

- The Views module behaves in surprising, and generally undesirable, ways if you try to display data from a field stored in a taxonomy term as part of a view of node data. For instance, if you have a node reference field in an “Interview” content type pointing to a “Person” node, and you only want to display the person’s last name (which exists separate field as part of the “Person” content type) as part of a display listing all interviews and interviewees, you can easily substitute the last name field for the person’s first name. If you have people stored as taxonomy terms, with separate fields for first and last name as part of the taxonomy term, trying to make that substitution will generate duplicate listings for any interview with more than one person associated with it.

- You can use the Views module to create a taxonomy-like display of all content that refers to certain kinds of nodes, using a node reference field. You can also use Views to create an RSS feed for all content associated with those nodes. Essentially, there’s nothing unique about the way taxonomy content is presented by default-- you just have to do a little more work to get it.

In short, if there are factors that make it preferable to create a content type rather than an elaborate taxonomy vocabulary, but you prefer the way taxonomies display, go with the content type-- it's not terribly difficult to recreate that look using Views.

5.4.3 When to merge similar content types

What if you’ve looked at your data, broken it down into content types and fields, and discovered that some of the content types will have identical, or nearly-identical, sets of fields? An example might be content types for information about undergraduate students and graduate students: perhaps the “Grad Student” content type has an extra
field for dissertation topic, but is otherwise identical in form to the "Undergrad" content type. You could set them up as two different content types, but that means doubling the configuration work, both now and in the future. It's extremely likely that, at some point, you'll need to add another field to the "Grad Student" or "Undergrad" content type, and if it's one that applies to both, you'll have to remember to update it in two places. If these content types are truly so similar, and you anticipate that they'll generally remain so, you might want to consider creating a single "Student" content type, and include a "Text (list)" field for selecting whether the student is a graduate student or undergrad. To accommodate a small amount of variation, you can use the Conditional Fields module\textsuperscript{26} to have the dissertation topic field show up only after "graduate student" has been selected.

On the other hand, your site might evolve in a different direction, providing more (and more detailed) information about grad students than undergrads. How likely is it that, for example, you might want to provide a "short bio" field for your grad students (but not undergrads), or include the courses your grad students are teaching or TA-ing. Is the similarity between the "Grad Student" content type and the "Undergrad" content type more connected to the role that both play within the overall data set of your project, or is it because that aspect of your project (how much data to include, and/or what to do with it) isn't fully worked out yet? Moving data from one content type to another will be a pain regardless of whether you're splitting a content type, or merging two into one. At a certain point, you have to make your best guess about how your project will develop, and move on.

Usability is another factor to keep in mind here. "Undergrad" vs "Grad student" are easy enough to understand as options in a drop-down list as part of a "Student" content type. But what if your site stores the full text of 19th century poems, as well as the full text of user-contributed essays about those poems? Maybe in both cases, the only fields you have are the out-of-the-box title and body fields. Should you have one content type (where the user chooses "poem" or "essay" from a text list field), or two? In this case, chances are good that the users of your site think about the poems and essays as very different things, particularly if you're anticipating that scholars with a connection to this material will be contributing it. Creating a single content type might be easier for you to configure, but at the cost of making the site feel less intuitive to your users. In such a situation, it's often better to create two content types, especially if there's not a lot of field configuration work that would need to be done twice.

5.4.4 File types vs. content types
If your site uses images, audio, video, and/or documents in more than an incidental way, the Media module\textsuperscript{27} is essential, but it also complicates the data modeling for your site. Without Media, to store any sort of information about the file, you’d need to create a content type with fields to capture that information (e.g. a Photograph content type to record the photographer, the date the photo was taken, the location, the collection or archive where it comes from, etc.) The Media module provides a new “thing”, file types, that can store information that will be

\textsuperscript{26} https://www.drupal.org/project/conditional_fields

\textsuperscript{27} https://www.drupal.org/project/media
associated directly with the piece of media, which may appear in multiple places and in multiple ways on the site. Storing that information in a node where the image has been uploaded, as you would have to do without the Media module, does not associate the metadata with the file in such a way that the metadata comes with the file if the file is used in another context (for instance, embedded in a paragraph of text elsewhere on the site.)

When should you add fields to a file type? And when should you create a content type to store some of the metadata? It depends on the overall scope and nature of your particular data set. If your data contains photographs of paintings, you may want to add some fields to the file type in order to document things like the source of the image file (for instance, if you are using images gathered from different museum websites). You might also want to add a field for the person who photographed the painting.

In most cases, you should probably still create a content type for storing information about the painting itself -- the date it was painted, the artist, the dimensions, the media, etc.-- even though you could add those fields to the “image” file type, or create a new file type for “painting” that has those fields. One advantage of using a content type to store this information is that it makes the data entry experience more uniform: creating a blog post, and adding a new painting, would be done the same way. Due to a constraint in the Views module (which you will use to generate most of the lists and other displays of data on your site), Views can only display one kind of data at a time: nodes, users, or files. Storing most of the information about paintings in the same kind of data you might store additional information (e.g. biographies of painters) gives you more flexibility when developing Views later on.

5.4.5 User profiles vs. content types

If the data on your site is primarily about people, you face a choice between storing that information in user profiles (see section 10.4) and using a content type. The determining factor here is whether the people in question will themselves be logging into the site and editing their information. If people expect to be able to edit information about themselves, it’s easiest to store it as part of the user profile. Each user automatically has permission to edit their own profile information; all you have to do is create a user account, and they can edit their information as soon as they log in. If you use a content type in a situation where people expect to be able to edit their own information, you have to create a user account so the person can log in, then create a node for the person, and set the person’s user account as the author of the node. Then, you either need to provide the user with the URL of the node you created with their information, or create a block using views that will display nodes where the user is listed as the author, and put it somewhere visible. (See chapter 11 for more on blocks, and chapters 12 and 13 for more on views.)

If the data on your site is primarily about historical people, or people who aren’t part of the community of site users, using a content type to store information makes the most sense. It becomes more complicated if the people who make up the site’s data are a mix of users and non-users. If a significant number of people are non-users, it may be better to use a content type, even though it requires more work to set up each user who is also a person in the database. Setting up user accounts that you don’t expect will ever be accessed is less than ideal from a security standpoint, and since you have to use a unique email address for every account on the site, generating a large number of unused email addresses that you control can quickly become cumbersome.
5.4.6 Drupal-based considerations for determining content types

There are other Drupal components (both modules and core functionality) whose settings are based around content types (i.e. different content types can have different setting configurations, but all content of a single type-- like undergrads and grad students within a "Student" content type-- must have the same configuration.) Some of these include:

- **Hiding/displaying author and date information**: if you want some content to display "Submitted by [Drupal user name] on [date]" at the top, and other content to not display it, it's a factor in favor of different content types. If you want to customize that text, though, you may need to use the Views module anyway, in which case you can set up more specific conditions than just content type for when that text shows up.

- **Comments**: if you want some content to have comments enabled by default, and other content to not have comments by default, that's a factor in favor of different content types. Note that users with the right permissions-- "Administer comments and comment settings" -- can turn on comments for any new or existing piece of content, regardless of what the defaults are.

- **Pathauto**: if you want the automatically-generated URLs for two pieces of content to be radically different (e.g. "mysite.university.edu/2013/07/06/content-title" vs "mysite.university.edu/content-title"), that's one of the strongest factors in favor of different content types. Note that if you have just one example, or a handful of examples, of a piece of content you want to behave differently than the way you've configured Pathauto for its content type, you can always manually change the URL.

- **Permissions**: the Drupal permission system breaks down permissions by content type (add content type X, edit/delete own content type X, edit/delete all content type X). Using the example above, if you want some users to be able to create "undergrad" content, but not "grad student" content, that's a moderately strong factor in favor of different content types.

5.4.7 Drupal’s “title” field vs real-world titles

While content types can vary in the number and nature of their fields, every single content type must have the default “title” field. Every node (instantiation of a content type) must have a Drupal title. There are ways of hiding it from display, there are ways of automatically generating it, but there is no way to circumvent the fact that *every node must have a title* as Drupal understands it.

By default, the Drupal title is displayed at the top of the page when a user is viewing the node. In addition, if you have a content type that uses a node reference field (a field used for pointing to a different node), you'll have to use the Drupal title of the node you want to reference. (There are ways around this by setting up a View that displays some other information from the nodes to be referenced, but using the Drupal title-- either via
a select list or autocomplete, is by far the simplest.) Particularly if your site uses node reference fields, it's important to make good use of the Drupal title, and this might require not thinking of it primarily as a "title".

Depending on the nature of the content type you're creating, the default Drupal title field might be a good fit for capturing real-world titular data. If you have a "Project" content type, you can use the Drupal title field as-is to store the project title. Leaving the Drupal title field alone is also a good idea for the default “Basic Page” and “Article” content types (or their equivalents), since web pages and blog posts tend to have titles.

What do you do with a content type that doesn't naturally have a "title" in the same way that a blog post or a project does (e.g. a dictionary word), or where the real-world “title” associated with the content type shouldn't be given the same prominence as some other piece of information (e.g. a content type for a person, where their name should appear most prominently, not their professional title)? Out of the box, Drupal allows you to give the "title" field a different label on the node creation page (part of the "submission form settings" when you're editing a content type or creating a new content type, see below). For these two cases, you could change the label of the Drupal title field to "Word" and "Name", respectively. Somewhat confusingly, in the second case, you might want to add a field that is labeled "title" to the person content type, to store the person's professional title.

Even though the Drupal title field will be relabeled "Name", its *machine name* (how it’s stored in the database-- which is never displayed to the end user and only appears on a few administrative pages) will still be "title". If you add a field to store the person’s professional title, that field will be labeled "Title" but its machine name will be "field_title". Keeping an eye on the machine name, and whether it's prefixed by “field_” will be important in a few situations where you'll use the machine name as a placeholder for data stored in that field, such as when configuring Pathauto (see section 5.6).

What if there's no single piece of data in the content type that's a good fit for the Drupal title? As an example, take a content type for storing brief weekly updates a project. Drupal will automatically capture the username of the person creating the update, and the date and time it was created. The default "body" field can be used to store the actual text of the update, but what of the title? Something like "1/1/13 Update (Jane)" might make sense, but if you use the Drupal title field, the people working on the project would have to enter that information manually-- in spite of the fact that Drupal is already capturing two of the three bits of information (the date and the username of the person adding the update). You could include instructions about your conventions for the titles of project updates, but that means extra work for your project assistants that's more likely to generate inconsistencies than actually be useful. A better way to generate
useful Drupal titles in such cases, without increasing the human work involved, is to use the Automatic Nodetitles module\textsuperscript{28}, described in section 5.5.4.

5.5 Preliminary data modeling for the example site

5.5.1 The data

The data for this project consists of the following:

- Biographies of African Americans in medical professions, and specific events from their lives, including temporal and spatial information about those events.
- Images of African Americans in medical professions, including (but not limited to) photos of people whose biographies will also be stored on the site.
- Student essays about African Americans in medical professions.
- Student “exhibits” that present a sequential series of information about African Americans in medical professions.

**Example 1: Biography**

Alexander Thomas Augusta (March 8, 1825 – December 21, 1890) was a surgeon, professor of medicine, and veteran of the American Civil War. After gaining his medical education in Toronto, he set up a practice there. He returned to the United States shortly before the start of the American Civil War. In 1863, he was commissioned as major and the United States Army's first African-American physician and also the first black hospital administrator in U.S. history. He left the army in 1866 at the rank of Brevet Lieutenant Colonel.

- Alexander Thomas Augusta was born, March 8th 1825, Norfolk, Virginia
- Applied to study medicine at the University of Pennsylvania, was rejected, 1845, University of Pennsylvania, Philadelphia, PA
- Moved to California to earn money for medical school, 1846, California, USA
- Married Mary O. Burgoin, January 12th 1847
- Enrolled at Trinity College at the University of Toronto, 1850, University of Toronto, Toronto, Ontario
- Received degree in medicine from the University of Toronto, 1856, University of Toronto, Toronto, Ontario
- Left Canada for the West Indies, 1860, West Indies
- Given a Presidential commission in the Union Army, October 1862, Washington, DC, USA
- Received a major's commission as surgeon for African-American troops. This made him the United States Army's first African-American physician (of a total of eight) and its highest-ranking African-American officer at the time., April 4th 1863, Washington, DC, USA
- Assaulted; three people were arrested, May 1863, Baltimore, Maryland

\textsuperscript{28} https://www.drupal.org/project/auto_nodetitle
• Commissioned Regimental Surgeon of the Seventh U.S. Colored Troops, October 2nd 1863, Washington, DC, USA
• Wrote to Judge Advocate Captain C. W. Clippington about discrimination against African-American passengers on the streetcars of Washington, D.C.: "Sir: I have the honor to report that I have been obstructed in getting to the court this morning by the conductor of car No. 32, of the Fourteenth Street line of the city railway. I started from my lodgings to go to the hospital I formerly had charge of to get some notes of the case I was to give evidence in, and hailed the car at the corner of Fourteenth and I streets. It was stopped for me and when I attempted to enter the conductor pulled me back, and informed me that I must ride on the front with the driver as it was against the rules for colored persons to ride inside. I told him, I would not ride on the front, and he said I should not ride at all. He then ejected me from the platform, and at the same time gave orders to the driver to go on. I have therefore been compelled to walk the distance in the mud and rain, and have also been delayed in my attendance upon the court. I therefore most respectfully request that the offender may be arrested and brought to punishment."
   February 1st 1864, Washington, DC, USA
• Awarded a brevet promotion to Lieutenant Colonel, March 1865, Washington, DC, USA
• Accepted an assignment with the Freedmen's Bureau, heading the agency's Lincoln Hospital in Savannah, Georgia, October 1866, Savannah, GA
• Left military service at the rank of Lieutenant Colonel, October 13th 1866, Washington, DC, USA
• Testified before the United States Congressional Committee on the District of Columbia with regard to Mrs. Kate Brown. Mrs. Brown, an employee of Congress and an African American, had been injured when an employee of the Alexandria, Washington, and Georgetown Railroad forcibly ejected her from a passenger car. The railroad was prohibited by its federal charter from discrimination against passenger because of race, February 26th 1868, Washington, DC, USA
• Began teaching anatomy at Howard University, November 8th 1868, Howard University, Washington, D.C.
• Returned to private practice in Washington, D.C., 1869, Washington, DC, USA
• Received honorary MD from Howard University, 1869, Howard University, Washington, D.C.
• Attending surgeon to the Smallpox Hospital in Washington, D.C., 1870, Washington, DC, USA
• Received honorary AM from Howard University, 1871, Howard University, Washington, D.C.
• Stopped teaching anatomy at Howard University, July 1877, Howard University, Washington, D.C.
• Alexander Thomas Augusta died, December 21st 1890, Washington, DC, USA

Example 2: Image
5.5.2 The goal

At the inception of this project, the data was primarily stored as a series of Word files. A subset of the data was available on the website of the Center for the Study of the American Constitution, part of the UW-Madison department of history. The web presentation was static; 15 individuals were chosen for inclusion, and each individual had a page that included a link to the bio (stored as a PDF), and a list of quotes about the person.
The project began without a specific research question or agenda; instead, it was an experiment in what could be done fairly expediently with Drupal to make the data explorable in new ways. "Build a database that lets people look at the data in interesting ways" isn't a goal that's very helpful in resolving questions that come up while working through a data model. It's much better, if at all possible, to work towards a more specific goal or research question (e.g. "I want to explore the differences between how the founders described themselves, and how other people described them" or "How do the ways Alexander Hamilton's contemporaries describe him change after his death?"). The final shape of the project is still under development, and so a lot of the possibilities and options described below don't have a resolution, one way or another.

5.5.3 Breaking down the data

The most obvious way to break down the data is into two content types: "Person" and "Image". The "Person" content type will store information about people, and the "Image" content type can store images that are unrelated to the people featured elsewhere on the site. So, we could have content types as simple as the following:

- **Person**
  - Name (can rename default Drupal title field and use that)
  - Biography (can rename default "body" field and use that)
- **Image**
  - Title (can be used to store a brief description of the image)
  - Description (can rename default “body” field)
  - Image (a file field; can choose the existing “Image” field that exists as part of the “Article” content type)
  - Tags

This kind of setup will essentially recreate the sources of your data.

By putting all the information about a person in a single text field, the information is human-readable, but there's not much more you can do with it, because the data is not structured in any way. There's no way to pull out dates for use in a timeline, or locations for use in a map, because they're mixed in with many other kinds of data.

The image content type is much like Flickr, with a title, description, and tags. Images that are stored this way can be difficult to find, because everything depends on the tags, an uncontrolled list of terms that may vary in quality. Two different images may use different words that express the same concept in their tags. There's also no way to easily pull out images of a person whose biography is stored on the site.

This simple data model essentially only supports the display of information in the form in which it was entered. If this is sufficient for your project’s needs, Drupal is probably not the best choice of platform for your project. WordPress would offer the technical capabilities you need, with less configuration required.
5.5.4 Enriching the content types

The more you structure the data using fields, the more opportunities you'll have to browse, compare and analyze the data in different ways. The cost of more fields is data entry time-- it's faster to copy and paste a big block of text that contains an undifferentiated mass of text, names, dates and locations than to put dates into individual fields, names in another set of individual fields, etc. (Note: you can bulk import content into Drupal, too, but you'd still need to spend time preparing your data for the import, by differentiating the content that will be imported into different fields.) If you know the data set well, you should be able to tell what data should be put in a separate field (because of a reasonably high potential for yielding interesting insights), and what data is unlikely to be worth the effort.

Assuming for the moment that we have two content types, "Person" and "Image", here are some possible ways to enrich those content types using more fields.

5.5.4.1 Person

Sometimes, data that starts out as prose doesn't need to be stored or presented as prose. One could create a set of fields as part of the "Person" content type that capture facets of individuals' lives that can be shared across many individuals. For instance:

- Birth date
- Birth location
- Schools attended
- Marriage date
- Marriage location
- Spouse
- Medical program attended
- Medical program graduation date
- Residency date
- Residency location
- Employment locations
- Death

This kind of spreadsheet-like list of dates and locations may render the data in the biography more accessible to Drupal for reuse in other visualizations, but it makes the data much less accessible to a human audience. In this case, it would be best to maintain at least a short narrative biography as part of the "Person" content type.

Providing narrative prose is not always necessary, as not all content types are meant to be seen directly by users. If the only place a human is going to encounter the data stored in this content type is as part of a table, map or other presentation form generated by the Views module, there's nothing wrong with replacing prose with fields of data.
Even if you want to maintain the human-readable prose display, with sentences and paragraphs, it is worth considering whether it makes sense to additionally store some of the data in individual fields, so it can be used in other contexts.

Which bits of data do we want to store as fields? It depends on the questions we want to be able to explore with the database. Sometimes it makes sense to pick only the data that seems most likely to yield interesting results, develop the site with those fields, and go back and add more fields later if the site is a success, and you encounter new streams of funding (or student labor). The more fields you include at first, the greater the risk that you'll never get a sufficiently comprehensive set of data entered in order to launch the site.

For our example site, let's include the following fields as part of the “Person” content type:

- First name - separating the person’s given name from their surname will allow you to sort the historical figures alphabetically
- Middle name(s) - see above
- Surname - see above
- Birth date - can be used to identify the person's age at the time of any event in his/her life
- Death date - indicates the end of a person’s timeline.
- Image - the image that will be used as a general-purpose thumbnail for the person throughout the site.
- Medical institution attended - a person in the database has affiliations with many institutions, but the institution where the person received medical training is a particularly influential affiliation, and we want to capture that information in a way that we can easily use to sort and group individuals.
- Profession - the person’s profession(s) within the medical field (e.g. doctor, nurse, researcher, etc.)
- Specialization - the person’s medical specialty.
- Biography - a brief prose biography.

Note that there are no fields within the “Person” content type that that allow you to store data from the “Biography” in a more granular way, to enable its use in other visualizations. Sometimes, in the process of enriching content types, it becomes clear that what originally seemed like a single content type should in fact be two or more. In the case of our example site, with the “Person” and “Image” content types, the more natural place to capture some of the information surrounding events in a person’s life would be part of the “Person” content type. Such an arrangement would be problematic both on a technical level and on an architectural level.

It’s easy to imagine a set of repeating “event” fields as part of the “Person” content type: a date, a location, a description of the event, and perhaps some other metadata. Repeating groups of fields has been a technical stumbling block for Drupal for many
years, though the development and stabilization of the Field Collection module\(^\text{29}\) has gone a long way towards addressing this need. While storing event information as part of the “Person” content type is now technically possible, it would significantly complicate the work involved in creating views to display timelines, maps, etc.

The possibility of events involving more than one person clarifies the need for a separate “Event” content type. This data set includes events where two people in the database were equal actors, and if we were to store event information as part of the “Person” content type, we’d have to choose between 1) having to create the event twice, once for each person involved; and 2) having to choose which person to store the event under, possibly with a pointer (such as a node reference) to the other person.

It is much simpler, and more intuitive, to additionally create an “Event” content type, which includes a node reference pointing to the person or people involved in the event. The node reference field should be part of the event, pointing to the person -- and not the other way around -- because information about the person or people involved is essential to the event. Information about any specific event is less essential to a complete “Person” listing.

The existence of the “Event” content type may cause you to reevaluate the fields you have as part of your “Person” content type. Since birth and death are both events in a person’s life, do you need the “birth” and “death” fields in the “Person” content type?

There’s no right answer, it’s fundamentally a matter of preference. For our example site, we want the birth and death dates of the person to appear alongside their biography, and the easiest way for that to happen is to have “birth” and “death” fields as part of the “Person” content type. However, one consequence of that approach is that birth and death information would have to be entered twice: once as an event (so it can show up on timelines) and once as part of the creation of a “person” node. To eliminate this duplication in data entry, you could configure a “List (text)” field as part of the “Event” content type where the person doing data entry can indicate whether the event being entered is a “birth” event or a “death” event via a dropdown menu or set of checkboxes. You could then use Views to have information from the “birth” and “death” events for a person appear on their profile page.

For the purposes of constructing our example site, we will accept data entry duplication as preferable to the additional configuration, and will keep the “birth” and “death” fields as part of the “Person” content type.

### 5.5.4.2 Event

The “Event” content type will store the following information:

\(^{29}\) [https://www.drupal.org/project/field_collection](https://www.drupal.org/project/field_collection)
- Person - the person or people who were involved in the event.
- Date - the date of the event.
- Location - where the event took place.
- Description - a description of the event.
- Event type - what kind of event it is, e.g. marriage, graduation from medical school, children, etc.
- Institution - if relevant, what institution the event is connected to (e.g. if the event is graduation, what institution was the person graduating from?).
- Main timeline - should this event appear on the main site timeline, or just the personal timeline for an individual?

5.5.4.3 Image

The image content type might include the following fields:

- Description - a description of the image.
- Date - when the image was created (the original image, not the digital surrogate).
- Person - people with biographies on the site who are depicted in the image.
- Profession - a controlled vocabulary indicating different professions within the medical field (e.g. nurse, physician, researcher, surgeon) depicted in the image.
- Institution - a standardized way to capture what institution(s) the image depicts or is connected to.

Once you have those fields, you will want to use the “Automatic title generation” setting (on the content type editing page) provided by the Automatic Nodetitles module\(^{30}\) to take care of the Drupal title field for the image content type, since many images may not have a canonical title.

6. Configuring content types and fields

6.1 Overview
This chapter is a practical guide to configuring content types and commonly-used fields.

6.2 Creating new content types

Go to Structure > Content types > Add content type (/admin/structure/types/add). Give the content type a name and, optionally, provide a description. The description will appear on the "Add content" page, which your users may or may not see, depending on how you configure the site (e.g. you might decide to provide a link in the sidebar for authenticated users to "Add a new project", which takes them directly to the form where they can do that, bypassing the "Add content" page entirely.) It's still a good idea to

\(^{30}\) https://www.drupal.org/project/auto_nodetitle
include a brief description of your content type, as a form of documenting your work for subsequent site developers.

The following settings are provided as vertical tabs underneath the title and description. Additional options may be available there, depending on which modules you install.

6.2.1 Submission form settings
- Title field label: how the Drupal title field is labeled; see 5.4.6 for a discussion of Drupal titles.
- Preview before submitting: whether to allow or require users who are creating content to see how it'll look before it's saved
- Explanation or submission guidelines: additional text that can appear at the top of the content creation/editing interface

6.2.2 Publishing options

The default settings are generally fine, and two of the options ("Promoted to front page" and "Sticky at top of lists") are only relevant for specific kinds of Drupal sites, and digital projects tend not to be among them. Note, too, that you're configuring the default options for content creation-- for any given piece of content, users with the correct permission will be able to make different choices.

- Published: enabled by default. One of Drupal's built-in permission options is "view published content". Having a public site means that you want anonymous (non-logged-in) users to be able to view published content. (Note: there are many ways to refine access permissions, such as only allowing anonymous users to view certain content types; see the section on permissions for more information.) You might want to disable this if, for instance, you want to set up a simple review workflow where a student assistant creates content, but you review it before you assign it the status of "published".
- Create new revision: Drupal can store multiple versions of a piece of content, reflecting different revisions (much like Wikipedia does). Drupal doesn't do it by default, but if you check this box, it will.

6.2.3 Display settings

Display author and date information: enabled by default, this puts a bit of text that reads "Submitted by [user's login name] on [date, by default a long-format version like "Sat, 09/07/2013 - 07:25"] towards the top of the piece of content. There are ways to configure this information (e.g. replacing the user's login name with their first and last name, pulled from their user profile, or changing the date format), or you may want to remove it altogether (it's disabled on the default "basic page" content type, for instance; for blog-like content, maybe you want to display a thumbnail of the user's picture and a brief bio in the sidebar instead).
6.2.4 Comment settings

If you've enabled comments (and if you used the default, rather than the "minimal", Drupal installation profile, comments are enabled by default), this section will be available.

**Default comment setting for new content**

By default, comments are open for new content types. If you don't want users to be able to comment on a particular content type's content, change the "Default comment setting for new content" to "Hidden". Even if you only want comments on only one content type out of ten on your site (e.g. your project's blog, but nowhere else), you'll have to change the default comment setting for every content type where you *don't* want comments.

There's another option for comments, "Closed". This shows all existing comments (if they exist, which may be the case if comments were previously set to "Open"), and a message that commenting is now closed. It doesn't make much sense to use this as a default comment setting for new content, because unlike "Hidden", it will display a conspicuous message that comments are closed.

**Allow comment titles**

Enabled by default, this is a setting that may seem odd if you're familiar with comment interfaces on non-Drupal sites. It's a consequence of the fact that Drupal has its own internal "title" requirement, which applies to comments as well as nodes (content created using a content type); see section 5.3.5 for more information. It may be worth disabling this option, for the sake of user familiarity-- when people leave comments on the internet, they don't often think of the comments as having a title. Drupal will create its own "title" for the comment for internal purposes, consisting of the first few words of the comment text, and the user won't have to deal with this particular quirk.

6.2.5 Menu settings

You can choose which menus are available for the content creator to choose from when creating a menu link, as well as a default parent for new menu items.

Note that these settings do *not* automatically add new content to a menu, they just define the options available when the content creator specifically chooses to do so. The Auto Menu module\(^{31}\) can automatically add new content to a menu.

---

\(^{31}\) [https://www.drupal.org/node/1138946](https://www.drupal.org/node/1138946); there is currently no official Drupal 7 release, but a functioning version is available in this thread.
Once you're done with these configuration options, choose "Save and add fields" to save your settings and move on to the area where you can define what information you want your content type to store, and how.

6.3 Adding and configuring fields

There are hundreds of modules on drupal.org that have been written to store particular kinds of data in particular kinds of ways. The content types you configure, and the fields you use to do so, are a significant part of what defines your site relative to all the other websites that run Drupal. The Drupal developers have included a few very basic field options as part of the Drupal core, but you should fully expect to install modules that provide additional field options.

6.3.1 "Manage fields" interface

You can access the "Manage fields" interface by going to Structure > Content types > [Your content type] > Manage fields. This administrative screen allows you to add new fields to your content type, and edit fields that you (or Drupal) have already created. It also introduces some new Drupal jargon:

- **Label**: This is the "human-readable" name of the field, which will appear on the content creation/editing form, and may appear as part of how the content is displayed. (The label "Body", for instance, would generally not be displayed as part of the content, but it still appears as part of the editing form.)

- **Machine name**: This is generally a variant of the "human-readable" name that meets a stricter set of requirements (lower-case, using underscores instead of spaces, etc.) It's used by Drupal's database and code, and appears in places like the CSS classes and IDs (information stored in the HTML to facilitate fine-grained web design) that Drupal generates. All fields you add have a machine name starting with "field_", and Drupal automatically generates a machine name based on the label you create; there's generally no need to change this. While it's possible to change the label of a field later, you can't generally change the machine name.

- **Field type**: These are defined by modules; the field types that are automatically available are defined by modules that are part of Drupal's core code. See more below.

- **Widget**: Of all the new Drupal terms, this is the most important to remember, as it is most likely to be useful when searching help forums. A field type defines what information is stored, whereas a widget defines how that data can be entered. Two fields of the same field type will display the same way by default after the content is saved, but choosing different widgets will change what those fields look like on the content entry/editing form. For instance, when adding a date field (as defined by the Date module), you can choose between

---

32 https://www.drupal.org/project/date
a calendar-selector widget and a text-entry widget. The choice changes the experience of the person entering the date, but doesn't change the nature of the data that's being stored by Drupal. If, however, you were to instead create a set of integer fields to store your dates, that would change the nature of the data that's being stored.

Both default fields and those fields you add to the content type can be arranged in any order, by clicking and dragging the + sign to the left of the field name up or down (see below). Be sure to save after you've rearranged fields.

Clicking and dragging the + sign next to the field label will allow you to rearrange fields.

The order you use for arranging fields here will be reflected in the node creation/editing interface for that content type, but not the display of the nodes you create. A separate order can be created for the display under the "Manage Display" tab (or by going to Structure > Content types > Your-content-type > Manage display).

6.3.2 Adding fields

By default, every new content type comes with two fields: the Title (which you may have labeled something else, but it will retain the machine name "title"), and a Body field. Not every content type will need a "Body" field or a functional equivalent; feel free to delete it if that's the case for your content type.

To add a new field, give it a label, select a field type, select a widget (the options available will depend on the field type, and any additional modules you may have installed), and hit "Save". You'll be taken to a series of administrative screens where you can further configure the field. The options available on these pages (particularly the first one, "Field Settings") will vary depending on the field type you've chosen; see below for specific notes on individual field types. Options available for all fields (on the second configuration screen) include the following:

- Required field: if a user doesn't enter a value for this field, will they be allowed to save the content, or will they encounter an error saying they need to fill out the field?
- Help text: text that appears below the field on the content creation/editing form
6.4 Core Drupal fields

The following default field types are the most commonly useful ones for digital scholarly projects. If you don't see some of these, go to the Modules page (admin/modules) and under the list of "Core" modules, make sure that "File", "Image", "List", "Number", "Options" and "Text" are enabled.

6.4.1 Text
This is a field for entering a short amount of text (less than a sentence), like a name, or other information where the user should be able to enter free text-- as opposed to selecting from a list-- and there's no need to try to standardize people's responses to a common set of terms (e.g. by using an autocomplete interface, though the Autocomplete Widgets module can provide such an interface for text fields). Unless you've installed an extra module like Autocomplete Widgets, the widget will always be “text field”. Configuration options include:

- **Maximum length**: by default 255 characters, but you may wish to expand this if your data might be lengthy (e.g. full 18th century book titles).
- **Text processing**: in most cases, you'll want to use the default "Plain text" setting, but if for some reason you need users to be able to enter HTML, you can choose “Filtered text (user selects text format)”. (See [future chapter] for more about text formats.)
- **Size of textfield**: the length the actual text field box on the screen is; this does not impose a limit on how much text the user can put in. The default value, 60, is generally reasonable. Having this box be too long may cause display problems on smaller devices, such as mobile phones and tablets.

6.4.2 Long text

This is a field for entering a larger amount of text-- a sentence or more. Unless you specifically need a "summary" version of a text field (e.g. if you're creating a content type for blog posts, and want to have a "teaser" that's something other than a shortened version of the body), the "Long text" field is the one to choose when you want a large text field.

---

33 [https://www.drupal.org/project/autocomplete_widgets](https://www.drupal.org/project/autocomplete_widgets)
With long text fields, you may want to make it easier for site contributors to format the text without directly entering HTML. Unlike WordPress, Drupal does not have a WYSIWYG editor (What You See Is What You Get -- which provides a set of buttons at the top of the field for things like bold, italics, lists, links, etc) by default, though this feature will be introduced in Drupal 8. Also unlike WordPress, WYSIWYG editors that you can install in Drupal don’t automatically include a button for uploading images or other media that you can embed in the text.

To provide WYSIWYG and media upload options for your long text fields, install and enable the CKEditor\(^{34}\) and Media\(^{35}\) modules, and configure them as described in sections 8.2 and 8.3.

Configuration options include:

- **Text processing:** unlike with Text fields, “Filtered text (user selects text format)” is often the right choice for Long Text fields. Filtered text is necessary if you want to use a WYSIWYG editor for that field.
- **Rows:** defines the actual size of the text box. The default value of 5 is usually fine.

6.4.3 List (text)

This is a field type that lets you define a set of options for users to select. By default, Drupal has two widget options here: 1) check boxes (if more than one value is allowed) or radio buttons (if only one value is allowed), or 2) select list. The select list widget is a little awkward, requiring users to hold down the "Ctrl" key if they want to select multiple options; there are modules like Chosen\(^{36}\) you can add that provide other widgets that may be easier to use.

In many cases, you’ll want to use a taxonomy field rather than a text list, particularly if the list of options is long, or will see frequent additions. Also, if the name of individual options might change, a taxonomy is a better choice. For instance, if you begin by listing universities in simple alphabetical order (University of Washington) and later decide to reverse the order (Washington, University of), this will be much easier if that data is stored as taxonomy terms.

As part of the field settings, you have to define an "Allowed values list", one value per line in the format key|label (e.g. grad_student|Graduate student). It’s best to follow the same conventions as machine names for fields when putting in the key values (lower case, underscores instead of spaces), but the labels can be more "human readable". One downside of these key/label values is that-- once users have created content using them-- you can no longer change them (in contrast to taxonomy terms, where you can

\(^{34}\) https://www.drupal.org/project/ckeditor
\(^{35}\) https://www.drupal.org/project/media
\(^{36}\) https://www.drupal.org/project/chosen
change the name of a term and all existing content using the term will be automatically updated).

On the second configuration page, if you increase the number of values the field can accept to more than one, the widget will change from radio buttons to checkboxes (if you're using the radio buttons/checkboxes widget.)

**6.4.4 Term reference**

This is a field type that allows you to present the user with a set of choices pulled from a Drupal vocabulary. It's similar to the "List (text)" field type, but more flexible (and complex). It has the same widget options as "List (text)", plus "Autocomplete term widget (tagging)", which provides autocomplete suggestions as the user types.

The field settings for the term reference field type allow you to choose which vocabulary to pull terms from. By default, Drupal configures a "tags" vocabulary, but you can add new vocabularies by going to Structure > Taxonomy > Add vocabulary. It’s best to create the vocabulary before you create the term reference field, so the vocabulary is available for you to reference. If you aren’t going to use “Autocomplete term widget (tagging)”, you should add some terms to the vocabulary so they appear as options. Additional configuration options for term reference fields are much the same as for the "List (text)" field type.

**6.4.5 File and the media browser widget**

This is a field for uploading one or more files, including audio and video files. The default “File” widget provides a limited set of options, largely self-explanatory. If you will be uploading files (documents, images, or other media) to your site with any regularity, it’s worth installing and configuring the Media module37, and using the “Media browser” widget with your “File” fields rather than the default “File” widget. It takes more work upfront, but it makes your files much more usable (by providing a way to store metadata along with files) and reusable (by making it easy for you to insert an image that’s been uploaded as part of your site’s data into a blog post or announcement, without having to re-upload it to a different node.)

The “File” field type, combined with the “Media browser” widget, is recommended not only for documents, but also for audio and video uploads, as well as pulling in media from third-party hosting providers (like YouTube or Soundcloud); see section 6.4.5.2.

---

37 https://www.drupal.org/project/media
6.4.5.1 Configuration for uploaded files

Install and enable the Media module. Go through the configuration settings for the Media module as described in section 6.6.

Add a File field, and choose the “Media browser” widget.

On the next configuration screen, you can choose whether users should have the option of choosing to display the file or not on the node where it’s been uploaded; you can check the second checkbox to have them displayed by default. In most cases, you won’t need this option. You can also choose an upload destination (private or public files); see section 6.6.1 for a description of these options.

On the following configuration screen, in addition to the usual field configuration options (help text, number of values, etc.), there are a number of Media-specific options.

- Enabled browser plugins: these determine the user interface of the media browser widget. If users are supposed to upload a file, be sure to enable the “Upload” plugin. “Library” allows users to choose from files that have already been uploaded, and “My files” shows files that the user has personally uploaded.
- Allowed file types: the kinds of files the user can upload or select from the library. You can configure the file type options (including what file extensions are associated with which file type) as described in 6.6.2. If you leave these checkboxes unchecked, all options will be available.
- Allowed URI schemes: leave this with the default value, which should correspond to the private vs. public decision you made in the previous configuration step.
- Allowed file extensions for uploaded files: specify the file extensions that users can upload. For file fields, this is limited to txt (plain text files) by default, so be sure to change this.
- File directory: if you want files uploaded in this field to be put in a specific sub-directory of the files directory (e.g. if this is a field for mp3 recordings of lectures, you might want to put them in a “lectures” sub-directory), you can indicate the directory here. If it doesn’t already exist, Drupal will create it.
- Maximum upload size: if you want to restrict the file size (in megabytes, kilobytes, etc.) that users can upload, you can specify that limit here.
- Enable description field: leave this unchecked; it is redundant with configuration options provided by the Media module.

6.4.5.2 Configuration for externally hosted files

The “Media browser” widget can be configured for uploading files, pulling in files from third-party hosting sites, or both. If you want to be able to pull in media from third-party sources, be sure to enable the “Media Internet Sources” module. You should also download, install and enable the add-on modules for Media that provide integration with the individual media hosting sites you...
want to use. A full list of these modules\(^{38}\) is actively maintained on drupal.org. Options include YouTube\(^{39}\), Vimeo\(^{40}\), and Soundcloud\(^{41}\).

Choose the multimedia browser options you want to enable for the field. If you want to use allow people to embed media from third-party providers, be sure to enable the “Web” browser plugin. This works for YouTube as well as other providers. The “YouTube” browser plugin is slightly different: it allows users to search YouTube from within your Drupal site and choose a video. If you simply want to be able to paste a YouTube link, the “Web” browser plugin in sufficient.

Under “Allowed URI schemes”, choose the third party providers that you want to enable, along with (or instead of) public or private uploads.

Otherwise, the configuration options are identical to those outlined above.

6.4.6 Image

This is a field for uploading one or more images. While you can use the file field type to upload images, using the image field type provides you with the same configuration options as the file field type (see section 6.4.5.1), plus additional ones specific to images.

On the first configuration page, in addition to private/public upload, you can choose a default image for the field, or leave it empty.

On the second configuration page, the “Image” file type is enabled by default (instead of “Document”, which happens with “File” fields), and the default file extensions are png, gif, jpg, jpeg. You can also specify maximum and minimum image resolutions, in pixels.

6.5 Fields provided by modules

Many scholarly projects will find at least one of the following field types (provided by modules) to be useful:

6.5.1 References

The node reference fields provided by the References module\(^{42}\) are the glue that holds complex data models together in Drupal. Unless you only need one content type to store your data, and pieces of data are not connected to other pieces of data, chances

\(^{38}\) https://groups.drupal.org/node/168009
\(^{39}\) https://www.drupal.org/project/media_youtube; be sure to download a version after 7.x-2.0-rc4, due to a bug in that version that renders it unusable (https://www.drupal.org/node/2401799). Until a new release is available, use the development version.
\(^{40}\) https://www.drupal.org/project/media_vimeo
\(^{41}\) https://www.drupal.org/project/media_soundcloud
\(^{42}\) https://www.drupal.org/project/references
are you'll need a Node Reference field. Even if not, the References module also includes a User Reference field, which can be useful on sites with a lot of registered users. For instance, in order to credit multiple authors on a critical essay, you could have a multi-valued User Reference field, and display that information at the top while hiding the default “author” information, since Drupal only allows one “author” per node.

First, install the References module, then enable References, along with Node Reference and/or User Reference (found under the "Fields" category on the module page). On the "manage fields" page for your content type (Structure > Content types > Your-content-type > Manage fields), you'll have an option for creating a "Node reference" or "User reference" field. Your widget options are "check boxes/radio buttons", "select list", or "autocomplete text field"; in most cases, the number of nodes or users to choose from will be large enough that "autocomplete text field" is your best bet, though "select list" (or one of the modules that provides an improved select list) may be another viable option.

On the field settings page, you can choose what content type can be referenced by this field. If you created a user reference field, you can limit the users by role (by default, "authenticated user" and "administrator" are the only options, but you can create and assign additional roles; see the section on user management), or status (active/blocked). If you have the Views module installed, you'll see a drop-down for "Views - nodes/users that can be referenced". This allows you to use the power of Views to put more complex restrictions on the nodes or users that can be referenced than just content type or role/status, respectively. To use this, you have to first create a view and add the "References" display type.

If you use the autocomplete widget, on the next page you'll have an option for choosing the type of autocomplete matching. The default value is "Contains" (i.e. it matches based on whether the text you've entered is present anywhere in the node title of candidate nodes), but if the node titles are predictable, changing it to "Starts with" might be more intuitive for people doing data entry.

Depending on your project's data entry workflow, you may want to add the Node Reference Create\(^{43}\) (noderefcreate) module. Imagine a project where the data consists of letters written between members of a community of 20-30 people, with biographies for each person. Every Letter node will have pointers to at least two people: the sender and the recipient. For this to work, those Person entries have to already exist in the system before you can add a Letter that points to them, which has implications for the order in which we enter data. But what if you want to assign one student assistant to entering people, and assign another to entering letters, without requiring that they closely coordinate their work? Node Reference Create provides a new widget for node reference fields, which allows you to create pointers to nodes that don't yet exist, creating them in the process.

\(^{43}\) https://www.drupal.org/project/noderefcreate
If, for instance, a Person node for Person A has been created, but Person B has not been created yet, the student assistant can still enter a letter from Person A to Person B. The autocomplete widget will find a match for Person A in the “author” field. In the “recipient” field, the student assistant will enter the name of Person B. No existing Person nodes will appear for them to choose from, but as soon as they save the letter, a Person node will be created for Person B (with no data other than a title) and the Luote node they just created will be linked to it. The only potential issue is that the student assistant entering the Person data will need to make sure the Person node they intend to create hasn’t already been created by Node Reference Create as a side effect of Letter data entry.

One downside to Node Reference Create is that it doesn’t work if the nodes it creates use a content type where all titles are generated using Automatic Nodetitles; see section 5.5.4.2 for further discussion and workarounds.

6.5.2 Date

The Date module provides a field for storing date information. It seems to have been developed with event data in mind—i.e., for current, or near-future dates—and it can easily accommodate information that includes a start-date and an end-date. The date module is very widely used across many different kinds of Drupal sites, and thoroughly tested. Depending on how closely your date information resembles modern event data (with a day, month, year, and optionally, a time), the date module may be sufficient to meet your needs.

If your data includes dates of different granularity (some consisting of year, month, day, and others consisting only of year and month), date ranges (e.g. 18th century), or incomplete dates (spring 1642), the Date module can’t accurately capture your data. The Partial Date module (see section 6.5.3) is intended to address these situations, but as of version 7.x-1.0-beta1, it does have a number of bugs and feature requests (such as Feeds integration, to enable data import) that have gone unaddressed for a long time.

First, install the Date module, and enable the following sub-modules (grouped under "Date/Time"):

- Date
- Date API
- Date Views
- Date Popup -- optional, if you want to enable a calendar pop-up widget for selecting dates; if you don’t enable this, you’ll be able to enter the date numerically

44 https://www.drupal.org/project/date
45 https://www.drupal.org/project/partial_date
• Date All Day -- optional, if you want to create "all day" events
• Date Repeat API and Date Repeat Field -- optional, if you want to create repeating dates

Once you’ve enabled those modules, you’ll see a notice about setting up the site timezone and first day of the week (Configuration > Regional and Language > Regional settings), as well as the date format settings (Configuration > Regional and Language > Date and time). Chances are, you already chose a site timezone when you installed Drupal, and the default first day of the week is Sunday. The default date formats use a 24-hour clock to display the time, and the medium and short formats use year-month-day ordering (e.g. 2013-09-15). If you find these default settings agreeable, you can skip any further configuration here, and head to the content type where you want to add a date field (Structure > Content types > Your-content-type > Manage fields).

If you want to change the long, medium, or short format, or if you want to add a new date format, go to Configuration > Regional and Language > Date and time > Formats > Add format and enter a format using the PHP conventions, which are linked below the field. For easy reference, here are some examples:

- F j, Y January 3, 2014
- M j, Y Jan 3, 2014
- m/d/y 01/03/14
- F j, Y g:i a January 3, 2014 2:15 pm
- M j, Y g:i A Jan 3, 2014 2:15 PM
- m/d/y G:i 01/03/14 14:15

Once you’ve defined a new format, you can go back to Date and Time (Configuration > Regional and Language > Date and time) and change any of the existing Date Types (Long, Medium or Short) to use that format. Or, you can create a new Date Type that uses the format.

On the fields list for your content type, three new field type options will be available: Date, Date (ISO format), and Date (Unix timestamp). Unless your data is server logs, you should almost certainly choose "Date". You'll have a choice between a text widget and a select widget (unless you enabled the "Date Popup" module, in which case you can also choose that option).

The field settings page lets you specify the granularity of the date data you want to collect. A year is required, but any combination of month, day, hour, minute, and/or second in addition to year is allowed. You can also choose whether to collect an end date, and what time zone to use (for instance, on a site with an international user base that submits events, you might want to select "User's time zone" rather than "Site's time zone", so that when a user adds an event with a time associated with it, it doesn't save the time in the site's time zone, which may be many hours off from the actual time of the event.) All your data needs to match the granularity you specify here, unless you use the Partial Date module.
The second configuration screen looks deceptively generic, but two of very important settings are buried under "More settings and values" (beneath the "help text" box): Date entry and Default values. Date entry is the required order for the text-based inputting of date information: e.g., month first, or day first? There's a list of options, or you can define a custom one. Chances are, most projects using the date field will want to change the default value, which sets the date to "now" (i.e. the current date and time when new content is saved). Even if the date field isn't set to be required-- which might leave you to assume that a blank date will simply be left empty, as with most other fields-- the Date module will fill in the current date and time unless you change it to "no default value".

6.5.3 Partial date

6.5.3.1 Partial date formats
The Partial Date module provides two fields: “Partial date and time” and “Partial date and time range”. Both draw on module configuration that can be found at Configuration > Regional and language > Date and time > Partial date formats (/admin/config/regional/date-time/partial-date-formats). That configuration screen provides four format options you can choose between: Short (default), Medium, Long, and Custom. For each one, you can configure the following things:

- Uppercase or lowercase for AM/am and PM/pm in times
- Whether to use BC/AD or BCE/CE notation for years, and whether to mark only dates prior to the year 0, or both those before and after
- How different components should be separated (e.g. through the use of “/” to separate month, day and year, and “:” to separate hours and minutes)
- The order in which the components should appear, arranged through a drag-and-drop table
- How each component should be formatted; for instance, for “Day” the options are:
  - Day of the month, 2 digits with leading zeros, 01 through 31
  - Day of the month without leading zeros, 1 through 31.
  - Day of the month, 2 digits with leading zeros with English ordinal suffix.
  - Day of the month without leading zeros with English ordinal suffix.
  - A full textual representation of the day of the week.
  - A textual representation of a day, three letters.
  - Numeric representation of the day of the week 0 (for Sunday) through 6 (for Saturday).

After you have configured the partial date formats, you can add a “partial date and time” field or a “partial date and time range” field to your content type. The only widget option is “partial date”.

On the next configuration screen, there are toggle-down options for “Base estimate values” and “Minimum components”.

6.5.3.2 Base estimate values
Base estimate values allow you to define approximate time periods at different levels of granularity (year, month, day, hour, minute, second). These only become relevant if you enable an “estimate year”, “estimate day”, etc. field on the next configuration page, so if your data doesn’t need approximate dates, you can skip this section. The module provides the following set of default options for year, month, day, and hour. The earliest and the latest time values for the approximate time period are separated by a “pipe” character (|), and a final pipe character separates the time values from the label for the approximate time period:
- Year
  - -6000|1600|Pre-colonial
  - 1500|1599|16th century
  - 1600|1699|17th century
  - 1700|1799|18th century
  - 1800|1899|19th century
  - 1900|1999|20th century
  - 2000|2099|21st century
- Month
  - 11|1|Winter
  - 2|4|Spring
  - 5|7|Summer
  - 8|10|Autumn
- Day
  - 0|12|The start of the month
  - 10|20|The middle of the month
  - 18|31|The end of the month
- Hour (note the use of a 24-hour notation for each hour)
  - 6|18|Day time
  - 6|12|Morning
  - 12|13|Noon
  - 12|18|Afternoon
  - 18|22|Evening
  - 0|1|Midnight
  - 18|6|Night

Any of these options can be changed or deleted, or you can add new ones ("long 18th century", "dawn", etc.)

6.5.3.3 Minimum components
This option allows you to add validation to partial date fields, so that users are required to enter at least as much information as you specify under “Minimum components”. If users can leave the partial date field empty, you can leave all the boxes under “Minimum components” unchecked.

If you have chosen a “Partial date and time range” field, the minimum components will all begin with “From” or “To”.

6.5.3.4 Field configuration
On the next configuration screen, there are a set of options that only apply to the content type where you’ve currently added the field. (You can configure these options a different way if you reuse the field as part of a different content type.)

- Time zone handling: addressing time zone handling is mostly a Drupal requirement.
  Choosing the site time zone for this is generally the best choice.

46 The different options do have some implications: for instance, if you are creating fields for an upcoming event with a specified time, and the time zone handling is set to UTC, Drupal may list the event along with “past events” even before the event has started if time zone handling is set to UTC but the event takes place in California.
• Date components / date component estimates: these are the options that will be available for users to fill in. Uncheck any that are not applicable (e.g. hour, minute and second are not relevant for many historical dates.) By default, the “estimate” options are unchecked. The date components and date component estimates are not mutually exclusive; you could make both available for one or more levels of granularity (e.g. to allow either “April” or “spring”), but if you do include both, be sure to include some guidance in the help text for the field so the people doing data entry know what you’re expecting. For instance, if they know the precise month, should they also include the corresponding estimate information, so it says both “April” and “spring”, or is the estimate only for when no more precise information is available?

• Increments: because partial date fields provide a drop-down for entering data other than the year, the second / minute increments configuration allows you to choose between 1, 2, 5, 10, 15, and 30.

• Short / long description: these add a supplemental field for a textual description of the date, which can be used along with or in lieu of storing the date in a numeric way when that is impossible or undesirable. This makes it possible for you to store vague, arguable, or poetic dates (“the peak of Enlightenment philosophy”, “when the nightingales sing”) alongside more precise dates.

• Approximate checkbox: including a checkbox to indicate that a date is approximate may be useful when your data includes a mix of precise and approximate dates, but you don’t want to go all the way to using date estimate fields

• Hide the “remove date” checkbox: the “remove date” checkbox provides an easy way to remove a date, without having to individually clear out each of the dropdown fields. In most cases, it’s best to keep it available.

• Inline help: this allows you to include additional help information for the short/long description fields, the approximate checkbox, the dropdown for specifying dates, or the “Remove date” checkbox.

You can also define a default value for the partial date field, though in most cases that won’t be relevant.

As with all fields, you can choose how many values to allow (one, up to 10, or “unlimited”).

6.5.4 Link

The Link47 module (grouped under the category "Fields" on the module-enabling page) allows you to add fields for URLs. There are no widget choices or field settings; all configuration is done on the second page.

The default settings are fairly sensible. By default, the user must enter a URL (and Drupal will make sure it’s valid), and they can optionally enter a title. You can make the title required, or set a static title (e.g. if this is a field for a link to a user's personal website, you could just set the title to be "Personal website").

---

47 https://www.drupal.org/project/link
6.5.5 Geofield
There are multiple modules and groups of modules that provide mapping functionality in Drupal\textsuperscript{48}. The Geofield\textsuperscript{49} field type module is compatible with a majority of these approaches. It has multiple prerequisites; you must first install geoPHP\textsuperscript{50}, and you also need the Geocoder\textsuperscript{51} module to turn the text field indicating a location into coordinates that can be stored in the geofield. Install and enable these three modules, as well as the Geofield Map module (part of the Geofield module, but found under the “Other” section, along with Geocoder, on the modules page.

In your content type, create a text field where the text indicating the name of the location will be stored. This text can be as specific as an address, or as general as a country, and the Geocoder module will attempt to find reasonable coordinates to correspond to what you’ve entered.

Next, create a Geofield that will correspond to the text field, and choose the widget “Geocode from another field”. While text fields may be the most intuitive source for geocoding, Geofield can also work with geographic information stored in the EXIF data of images stored in an image field (e.g. from a cell phone photo that recorded the phone’s GPS data). Configuration options include:

- Geocode from field: select the text field you just created
- Geocoder: select the service you want to use for geocoding. Selecting Google Geocoder gives you access to advanced configuration settings, such as rejecting results at certain levels of precision (e.g. if you only want precise address coordinates, you can reject the results from Google if they are approximate)
- Multi-value input handling: by default, if the text field you’re geocoding from has multiple values, it will encode one point for every value of the text field

Geofield can also be used by itself, without involving the Geocoder. If you know the geographic coordinates of a location, you can choose that Geofield widget. If you have data encoded as Well Known Text (WKT, a markup language for encoding objects on a map), there’s a Geofield widget that supports that, too.

6.5.6 Field group
Field group\textsuperscript{52} is a module that allows you to display groups of fields together, on the node creation/editing form and on the node display. Field groups have to be set up separately for the node creation/editing form (which is done on the "manage fields" interface where you add new fields), to impact the data entry form, and on the "manage display" interface, to impact how the data stored in the node is displayed. Just because a field appears in a field group under "manage fields" does not mean that field group will carry over to "manage display" or impact how the data is displayed.

\textsuperscript{48} A comparison of these modules is available on drupal.org: https://www.drupal.org/node/1704948
\textsuperscript{49} https://www.drupal.org/project/geofield
\textsuperscript{50} https://www.drupal.org/project/geophp
\textsuperscript{51} https://www.drupal.org/project/geocoder
\textsuperscript{52} https://www.drupal.org/project/field_group
If your content type contains a lot of fields, it may be convenient to break up the data entry in some intuitive way. You can use the field group module to turn a long node creation/editing page into a more succinct tabbed interface. If your content type is for a location, you might have one tab for the name and a description, another for geographic coordinates, another for uploading an image, and another for a multi-value node reference field for similar places. You can do the same for the node display (though you'll have to do it separately, on the "manage display" configuration page) -- perhaps there's some information in the content type that you want to make available for those who are interested, but you feel it would be distracting for non-specialist users of your site. You can use field groups to create a collapsed fieldset to store those fields, and they won't appear unless a user expands the fieldset.

It is also important to understand what the field group module does not do. It does not create a new "thing" in the Drupal database with independent field-like properties. Putting fields inside a field group together doesn't create any real relationship between them. For example, let's say you have two fields -- a user reference field ("contributor") and a text list ("activity", with options like "author", "illustrator", "editor", etc.) You want your content type to potentially have multiple contributors, with each contributor having a corresponding "activity". You may be tempted to try putting "contributor" and "activity" in a field group together, and searching for a way to make that field group have multiple values -- the same way Drupal provides you with an "add another" button when you have an autocomplete field that can store multiple values. This won't work -- it isn't possible to make a multi-valued field group, that configuration option isn't available. There's been efforts among Drupal developers to address this "multigroup" problem for many years; the most widely adopted option for Drupal 7 is a separate module called Field Collection\(^\text{53}\), but it's somewhat buggy; don't even consider using it with fields that already have data stored in them without backing up your site first in case it deletes data.

The field group module has a lot of configuration options, though they all appear on the "Manage fields" or "Manage display" screen-- there's no separate-screen configuration options like with fields. The "widget" options (which, confusingly, will appear vertically under "field type" once you've saved the field group) determine how field group will set apart its containing fields. Some of the field group options should contain other, nested, field group options, rather than fields themselves. Once you've created a field group, you can drag and drop fields (or other field groups) into the field group, by clicking on the icon that looks like a + to the left of the field name, and dragging from there. The label of a field that's in a field group should appear slightly indented, underneath the name of the field group. If you need to delete a field group where you've already put fields, you don't have to remove those fields first-- Drupal will not delete fields along with the deleted field group.

The field group options are as follows, including the configuration options for each one (accessible by clicking the cog icon, to the left of "delete", which appears once you've chosen a widget for the field group):

- HTML element - a user-specified HTML element (e.g. div, section). A bare-bones option that makes most sense if you want to control the styling through the CSS of your site's theme; probably not the best choice in most cases
  - You can choose which HTML element, and add attributes and CSS classes that will appear along with it

\(^{53}\) https://www.drupal.org/project/field_collection
• Div - uses the HTML div element to set apart the fields, but unlike the "HTML element" option, it comes with some pre-configured, souped-up display settings that you can control.
  ○ Fieldgroup settings: collapsible (open by default, if content has been entered in the fields it contains, but can be collapsed), collapsed (closed by default), and open
  ○ Effect and Speed refer to the animation when the fieldgroup is opened or closed
  ○ Label element and Show label have to do with whether or not to display the label you entered for the field group above the fields it contains, and how big it should display ("Header 2" will display bigger than "Header 3")
  ○ ID and Extra CSS classes refer to things that will be embedded in the HTML, which you can leverage when editing the CSS for your site's theme
• HTML5 - much the same as HTML element option, except it provides a set of HTML5-compatible options
• Fieldgroup - a good default choice; similar to the Div option, but with simpler configuration options (only settings for whether or not the group is collapsible)
  ○ Vertical/horizontal tabs group - a recommended option; should be used to contain 2+ "Vertical/horizontal tab" field groups type, to create said tabs. Only one of the tabs in a tabs group should be open; the rest should to be set to "closed", until a user clicks to open it.
  ○ Accordion group / accordion item - similar to vertical tabs, except without the tab design-- a set of vertically stacked field groups that can toggle open/closed, with a transition animation.
  ○ Multipage group / multipage - allows you to create a multi-step process for data entry, where the person doing data entry must proceed through the steps in a particular order.
  ○ "Multipage group" fieldgroups allow you to define the page title (whether to show some variant of "Step 1 of 10"), and whether the "submit" button for the content type and other configuration settings should be on the last page of the multipage.
  ○ "Multipage" fieldgroups allow you to put in a description, and define whether that particular field group should be the starting point of the multi-step process. Only one such group should be set as the start. The other multipage fieldgroups and their contents will appear in the order in which they're nested beneath the "Multipage group" fieldgroup.

6.6 Media

The Media module includes more than just a widget, it also provides a new type of “thing” for storing content, equal in a sense to content types or taxonomy vocabularies. A media file can have its own metadata that’s stored with the file you’ve uploaded (or pulled in from an online hosting site, like YouTube or Soundcloud), and the metadata profile can vary depending on the type of media (i.e. you can have a one set of fields for describing images, and another for describing documents.)

Once you’ve installed and enabled the Media module, you can add media files by going to Content > Files (admin/content/file) and clicking the “Add file” link in the upper left. You can also add files by using a field within a content type, where you have enabled the “Media browser”

---

54 https://www.drupal.org/project/media
widget for that field. Regardless of which option you choose, any media file that has been uploaded to the site will appear on the files list at Content > Files.

6.6.1 Configuring the file system
Go to Configuration > Media > File system. The default locations for public, private and temporary files are fine unless you have a specific reason to change them. You may, however, wish to change the default download method. By default, the “public” method is enabled. This means that if an un-authenticated user gets the URL of a file (for instance, if someone emails it to them), they can download it, even if the link for that file only appears on a page with restricted access. Using the private file method allows you to make use of Drupal’s permissions system to restrict access to files.

6.6.2 Configuring file settings
Go to Configuration > Media > File settings (admin/config/media/file-settings). On this page, you can set a maximum upload size, if you have concerns about server space. You can also modify the default list of allowed file extensions (xml, for instance, is not on the list by default). Note that you can override this list by enabling, or restricting, file extensions for each field where you use the “Media browser” widget.

Towards the bottom, under “File upload wizard”, you should check the box for “Skip scheme selection”. This will eliminate a potentially confusing step in the upload process where users are asked to choose between the public and private download method (described in 6.6.1). Decisions about public vs. private download are generally best made at the project level, and users should not be asked to decide this every time they upload a file.

6.6.3 Configuring media browser settings
If you go to Configuration > Media > Media browser settings (admin/config/media/browser), you can make various changes to the media browser interface. You can choose to restrict some of the options, selecting among “Upload”, “Library” (previously uploaded media), and “My files” (files that the current user has uploaded); if you’ve enabled the Media Internet Sources module, you’ll also see “Web” (for putting in a URL to media on a supported site). Some of the modules that provide integration with third-party media hosting providers add further options, such as “YouTube” (which allows users to search YouTube from within Drupal).

You can use the “File directory for uploaded media” field to create potentially elaborate directory structures for your media files, rather than having them in one large directory. You can use tokens (as illustrated in section 6.7) to do things like create directories for each date, and within the directory for a given date, for each user who uploads something on that date. In most cases, though, it’s fine to leave this blank.

6.6.4 Configuring file type profiles
Go to Structure > File types (admin/structure/file-types). By default, there are four file types: Image, Video, Audio and Document. Each of them has an associated “name” field (required, just like Drupal’s node titles), but the Image file type also has an alt-text field and a title-text field. Including a description in the alt-text will improve the accessibility of your site. The title-text is for additional information that appears when a user hovers their mouse over the image.
For most cases, these four types and their existing fields are sufficient. For sites with a particular focus on media, however, you may want to consider adding other fields. If many of your images come from archives, you may want to add an “Archive” field to the image file type, to store information about the source archive in a way that’s directly associated with the image, regardless of where or how you display the image on the site. If you have multiple different kinds of images (for instance, those taken during fieldwork and those from archives) in the same site, you can create additional image file type profiles, each with its own set of fields for capturing important metadata about the file. In such a case, after you upload a new image, the Media upload interface will ask which kind of file type it should belong to, and will then present you with the correct set of metadata fields.

To add fields to an existing profile, simply choose the “manage fields” link for the profile; this will bring you to an interface that looks just like the “manage fields” interface for content types (described above).

6.7 Conditional fields

The Conditional Fields\textsuperscript{55} module allows certain fields to remain hidden until “triggered” by a value entered in another field. For instance, if you have a text list field, and one of the options is “other”, you can have a text field that will appear if the user selects “other”, to allow them to be more specific.

When you have installed and enabled the Conditional Fields module, a new tab, “Manage Dependencies”, appears when you’re editing a content type, along with “Edit”, “Manage Fields”, “Manage Display”, etc.

To set up a dependency relationship, first create all the fields that will be involved: both those that will appear/disappear, as well as those whose values will trigger that behavior. Dependencies can be set up for each content type (by clicking on the “Manage Dependencies” tab, alongside “Manage Fields”) or they can be configured on the screen that shows all dependencies for all content types, taxonomy terms, comments and user profiles (Structure > Field dependencies).

On the dependency configuration screen, the field that will appear and disappear is listed first, and is referred to as the the dependent field. The field that will trigger that behavior is the dependee field, and is listed second. The third column describes the behavior; the default options-- that the dependent is visible when the dependee “has value…” (where that value is specified on the next configuration screen)-- will generally be the one you’ll need to use. If you explore the other options, you can see that one could use Conditional Fields to trigger other behaviors as well, such as emptying a field (or unchecking a checkbox) when the user enters a value in another field, etc.

\textsuperscript{55} https://www.drupal.org/project/conditional_fields
The dependency configuration screen.

After you click “Add dependency”, the second configuration screen lets you define the conditions. Because our dependee field is a text list, and we only want one value (“other”) to trigger this field, we can use the default method “Insert value from widget” and select “other” from the radio buttons.

Selecting a value from a text list field.

If the dependee field is a term reference field (pointer to a taxonomy), the configuration is slightly more complicated due to a bug in the Conditional Fields module (as of 7.x-3.0-alpha1). In order for Conditional Fields to work with taxonomy fields, for the “Values input mode”, choose from one of the options under “set of values”; “Any of these values” is probably the most useful. Unfortunately, you can’t enter the name of the taxonomy term(s) you want to use as a trigger, you have to find the term ID.

For the vocabulary you’re using for this term reference field, go to Structure > Taxonomy > Your-taxonomy-name. You’ll see a list of all the terms in the vocabulary. Hover the mouse over
the “edit” button for a term you want to use as a trigger. At the bottom of your browser, you’ll see the URL for that link appear; part of that URL is the term ID:

In this case, the term “Other” has the term ID “10”, so we enter that field in the “Set of values” box:
A dependent field can have multiple dependees (i.e. be triggered by multiple different fields). For instance, you might have a “dissertation topic” field that appears as part of a graduate student’s profile if their enrollment date was more than 4 years ago (based on an “enrollment year” date field), or if they’ve chosen members of their dissertation committee (if there’s a value in a user-reference field for “dissertation committee”). In this case, you’d set up each of these dependencies and specify “OR” in the “Interaction with other dependencies” field. “AND” requires that both conditions be met, and “XOR” requires that one be met and other not (if they’ve specified members of their dissertation committee, their enrollment date has to be less than 4 years ago; alternately, if their enrollment date is more than 4 years ago, the dissertation committee field should be empty.)

6.8 Configuring Automatic Nodetitles

Install and enable the Automatic Nodetitles module, as well as the Token module, if you haven’t already. A new section called “Automatic title generation” will appear on the content type configuration page, above “Display settings”, “Comment settings”, and the other configuration options described in section 5.4.

Even though Automatic Nodetitles will be available as part of the first step of the content type configuration process (i.e. before you add fields), you should wait to configure Automatic Nodetitles until after you’ve created the fields you’ll want to use as part of the automatic title generation. Once you’ve added those fields to your content type (see section 5.5), return to the “Edit” area for your content type (Structure > Content types > Your-content-type) to access the “Automatic title generation” settings.

In most cases where you’re using Automatic Nodetitles, it’s best to choose the option “Automatically generate the title and hide the title field”. That will ensure that your nodes are titled more consistently than if you leave open the possibility of node authors’ entering a title on their own. Once you’ve selected that option, fill in the pattern you want to use for the automatically generated titles.

If you’ve installed and enabled the Token module, you’ll see a table below the “Pattern for the title” box with arrows that can be toggled to see more options. To access the fields you’ve created, click the toggle arrow next to “Nodes”.
If you first click in the “Pattern for the title” box, clicking on the blue token link text that corresponds to a field you want to appear as part of the pattern will put that token text into the pattern box. You can also type text directly into that box. Because we want the Person nodes to have titles that include their full name (which we’ve broken up into individual fields to enable sorting by last name), our pattern for the title might look like: [node:field_first_name] [node:field_middle_name] [node:field_last_name].

Things become more complicated if you want to include dates as part of automatically-generated titles. Imagine a project where the data consists of quotes by famous people about one another. This project uses a Person content type to capture information about the people, and Quote to capture the quote, pointers to the people involved, and a date. By default, a date field token includes hours, minutes, and seconds (all “00” if unspecified), and takes the format year-month-day hours:minutes:seconds. The result would be: “Melancton Smith about Alexander Hamilton 1788-06-28 00:00:00”. This isn’t ideal, but depending on how you want to display the content with automatically generated node titles, it may not be a problem. In the case of our example site, we plan on displaying the quotes as part of each person’s profile page; users won’t ever directly view individual nodes of the Quote content type, and so they’ll never see this odd-looking title. If you do want to use date fields as part of automatically-generated titles that users will see, some advanced configuration settings for Automatic Node titles are addressed below.
If at some point you decide to change the pattern you use for your automatically generated titles, the existing nodes that have had their titles generated automatically won’t immediately be updated. However, the next time one of those nodes is saved, its title will be updated. If you want to update all of the old nodes at once, you can use Views and the Views Bulk Operations (VBO) modules to re-save all the nodes; for more on how to configure VBO, see [section in future chapter].

If you decide you no longer need Automatic Nodetitles and disable the module, the Drupal title field will become visible. All the titles that have been automatically generated will be editable, as if they were normal user-generated titles.

### 6.8.1 Advanced Automatic Nodetitles configuration with Entity Tokens

To customize how date fields are displayed as part of automatic title generation, install the Entity API module\(^{56}\) and enable both Entity API and Entity Tokens. When you return to the automatic title generation screen, you’ll notice a number of new options:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TOKEN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current date</td>
<td>[node:field_attributed_to]</td>
<td>Node reference field.</td>
</tr>
<tr>
<td>Current page</td>
<td>[node:field-attributed-to]</td>
<td>Field &quot;field_attributed_to&quot;.</td>
</tr>
<tr>
<td>Current user</td>
<td>[node:author]</td>
<td>The author of the node.</td>
</tr>
<tr>
<td>Nodes</td>
<td>[node:field_date]</td>
<td>Field &quot;field_date&quot;.</td>
</tr>
<tr>
<td>Body</td>
<td>[node:body]</td>
<td>The main body text of the node. The following properties may be appended to the token: value (Text), summary (Summary), format (Text format)</td>
</tr>
<tr>
<td>Comment count</td>
<td>[node:comment-count]</td>
<td>The number of comments posted on a node.</td>
</tr>
<tr>
<td>Comments</td>
<td>[node:comments]</td>
<td>The node comments.</td>
</tr>
<tr>
<td>Comments allowed</td>
<td>[node:comment]</td>
<td>Whether comments are allowed on this node: 0 = no, 1 = closed (read only), 2 = open (read/write).</td>
</tr>
<tr>
<td>Content ID</td>
<td>[node:rid]</td>
<td>The unique ID of the content item, or &quot;node&quot;.</td>
</tr>
<tr>
<td>Content type</td>
<td>[node:content-type]</td>
<td>The content type of the node.</td>
</tr>
<tr>
<td>Creates revision</td>
<td>[node:revision]</td>
<td>Whether saving this node creates a new revision.</td>
</tr>
<tr>
<td>Date</td>
<td>[node:field-date]</td>
<td>Field &quot;field_date&quot;.</td>
</tr>
<tr>
<td>Custom format</td>
<td>[node:field-date custom:]</td>
<td>A date in a custom format. See the PHP documentation for details.</td>
</tr>
<tr>
<td>Long format</td>
<td>[node:field-date.long]</td>
<td>A date in 'long' format. (Friday, October 10, 2014 - 20:09)</td>
</tr>
<tr>
<td>Medium format</td>
<td>[node:field-date.medium]</td>
<td>A date in 'medium' format. (Fri, 10/10/2014 - 20:09)</td>
</tr>
</tbody>
</table>

Every token option with an underscore (e.g. `node:field_attributed_to`) now has a variant with a hyphen (`node:field-attributed-to`), provided by the Entity Tokens module\(^{57}\).

\(^{56}\) [https://www.drupal.org/project/entity](https://www.drupal.org/project/entity)

79
Furthermore, you can now toggle down the token for the Date field to select your display format, and you can choose one that does not display hours/minutes/seconds, or define one using the “Custom format” option.

Once you have Entity Tokens installed, you can also access fields that are part of a content type referenced using a node reference field. For instance, if you wanted your generated titles to look like "Melancton Smith (1744-1798) about Alexander Hamilton (1755-1804)", you would need to access the birth and death dates for each person--even though that data is not stored as part of the Quote content type. But because the Quote content type has a node reference field pointing to the Person content type where that data can be found, you can get to that data using the entity tokens available if you toggle down the node reference fields for “Attributed to” and “Source”. To get titles as described above, your pattern for the title would look like:


This uses a custom format for the date, assuming you don’t already have a date format set up that only shows the year. The “Y” in the custom format comes from the PHP conventions for representing dates; you can access the relevant page in the PHP documentation via a link under the entity token for a date in a custom format.

Currently, a bug in the Entity Tokens module (version 7.x-1.5) leads to problems when a field you’ve included in an automatically generated node title is left empty. Rather than showing nothing, the literal text of the token will appear as part of the automatically generated title (e.g. George [node:field-middle-name] Washington). For date fields that are left empty, Entity Tokens fills in the current date. As a result, it’s best to only use the entity token variant (with the hyphen; [node:field-middle-name]) for required fields, and stick with the token variant (with the underscore; [node:field_middle_name]) if you want to include optional fields.

Another bug with the Entity Tokens module that may lead you to adopt the regular token variant relates to some punctuation marks (including ampersands, quotation marks, greater-than and less-than signs, and many letters with diacritics). The token variant displays these without a problem, but the entity token variant transforms them into their equivalent HTML entities (so “John & Jane” displays as &quot;John &amp; Jane&quot if generated by an entity token). If you expect your token values to include some of these punctuation marks, you should use the token variant.

---

57 If you have hundreds of fields on your site, the number of options provided by Tokens and Entity Tokens combined may cause this screen to load slowly, or even cause an error. This problem remains under discussion in this issue on the Entity API module: https://www.drupal.org/node/1272560
6.8.2 Automatic Nodetitles and Node Reference Create
There is an inherent conflict between Automatic Nodetitles and Node Reference Create. The way Node Reference Create is supposed to work, the information you put into the node reference field becomes the title of the new node that’s created, if it doesn’t match a node that already exists. No other data is added to that new node—crucially, including the data that Automatic Nodetitles will try to use to automatically create a title. Automatic Nodetitles will override the title that Node Reference Create tries to provide to the new node, and the result will be a new node whose title is the literal value of the tokens you’ve set up to automatically provide the title, such as “[field_given_name] [field_surname]”.

One way to work around this is to use the Automatic Nodetitles configuration “Automatically generate the title if the title field is left empty” (rather than having Automatic Nodetitles hide the title field), and add some text to the content type’s Submission Guidelines (which appear at the top of the node creation screen, and can be configured under “Submission form” settings, on the same page where you configure Automatic Nodetitles for the content type) instructing users to leave the title field empty.

6.9 Improving node URLs with Pathauto
As you create your content types, it might make sense to enter in some dummy data, to see if the content type form is behaving the way you expect. However, before you begin entering the actual data for your project, you should install, enable and configure the Pathauto module. Pathauto lets you establish patterns that will be used to generate the visible URLs for the nodes you create. Without Pathauto, nodes you create will have a URL that looks like “yoursite.org/node/123”, where the number after “node” reflects the order in which the node was created. Using Pathauto, you can set up URLs that are more meaningful, such as “yoursite.org/people/alexander-hamilton” or “yoursite.org/quotes/alexander-hamilton/melancton-smith-1788-06-28”. Keep in mind that these more user-friendly URLs are actually URL aliases: node/123 will always work as a way to access the node with the node ID of 123, no matter how you configure Pathauto.

The configuration of Pathauto is much like the configuration of Automatic Nodetitles (see section 5.5.4); both require the Token module. Once you’ve enabled Pathauto, go to Configuration > Search and Metadata > URL aliases > Patterns. Each content type and taxonomy vocabulary will be listed, and you can set up a pattern for the URL alias for each one. If you toggle down the “Replacement Patterns” link, you’ll have the same token navigation interface available on the Automatic Nodetitles configuration screen. If you first click inside the field for any content type, you can then click on blue token links under “Replacement Patterns”, and they’ll automatically appear in the path field for that content type.

7. Further data modeling and applied content type creation
7.1 Overview
This chapter combines the theory of data modeling with the hands-on skills of content type creation from the previous two chapters, and applies them to the example site.
7.2 Configuring content types for the example site
In section 5.5 we identified a preliminary data model for our example site. At that point, our data model included only which fields we wanted to use to store the data, but not what kind of fields they should be, or how we would configure those fields. Let us now revisit our data model, with an eye towards actually creating and configuring these content types.

7.2.1 Person
The person content type stores information about individual people in the database. Create a new content type by going to Structure > Content types > Add content type (/admin/structure/types/add), and naming it “Person”.

7.2.1.1 Content type settings
Under “Display settings”, uncheck “Display author and date information”; it doesn’t matter who enters the information about people.

Under “Comment settings”, set “Default comment setting for new content” to “closed”.

Otherwise, the default settings are fine.

7.2.1.2 Fields

Given name
This should be stored as a text field, with the label “Given name”. All the default settings (255 maximum characters, 60 textfield length, “plain text” text processing, 1 value) are fine. You may want to make this a required field to ensure that any student assistants helping with data entry don’t accidentally omit it. You may also want to put in some help text (which will appear under the field), though this field is quite self-explanatory.

Middle name(s)
See configuration for “Given name”, above. Because not all people in the database have a middle name, this field should not be required. However, some people in the database have more than one middle name. You could handle this by increasing the number of values to 2 (if no one has more than two middle names), but doing so may needlessly complicate the data entry screen. If you’re not going to use the multiple-middle-name data for anything (e.g. displaying only those people with more than one middle name, or listing all the second-middle-names), it’s probably simpler to have a single field and include instructions as part of the help text indicating that when a person has multiple middle names, they should be entered together, separated by a space.

Surname
See configuration for “Given name”, above.

Biography
Edit the existing “Body” field and rename it “Biography”.

Birth date
All of the people in the database have known birth and death dates (day, month, year). This consistency within the data allows you to use a “Date” field (just “Date”, not one of the other variants) to store this information. If you don’t see “Date” as one of the options for field type, make sure you have the modules described in section 5.5.2.2.2 enabled. Use the label “Birth date” and the widget “text field”. Since some of the people in the database were born and/or died in the 19th century, the “select list” widget option would require a lot of scrolling, and it’s easier to just type the values in.

Under “Date attributes to collect”, choose only “year”, “month”, “day” and uncheck the boxes for “hour” and “minute”.

On the next configuration screen, you can enter help text if you want, but be sure to toggle down “More settings and values” right under the help text box. Select the date entry format you want to use (see section 5.5.2.2.2 for how to configure additional date entry formats), and change the default date from “now” to “none”. That way, if the person doing data entry needs to add a birth date after creating the Person node, Drupal won’t erroneously fill in the current date as the person’s birth date.

**Death date**
Should have the same configuration as the “birth date” field.

**Image**
There are two ways you could handle the storage of an image to be used as a thumbnail for the person throughout the site. The most straightforward would be to create an image field that allows you to upload an image file as part of the Person node. The image would automatically appear alongside the other information in the Person content type.

The downside to this approach is that you’d end up with two different groups of images, with different metadata about each of them. On the Person content type, you could add some extra fields to capture information about the image you’ve uploaded for the Person (description, date, etc.) but that information is not necessarily relevant when the picture appears in the context of the Person node, and in creating those fields, you’re significantly duplicating the Image content type. Alternately, you could enable the image “alt” and/or “title” fields, or use a module like Image Field Caption\(^\text{58}\), to present information about the image in an unstructured way, but it doesn’t change the fact that images attached to the Person content type would be treated differently than those attached to the Image content type.

Another way to handle this is through the use of a node reference field, pointing to the Image node for the image you want to use as the person’s thumbnail. This way, all images are uploaded the same way, with a consistent set of metadata fields you only have to configure once.

If you have a specific reason for treating the thumbnail images separately from all other images on your site, go ahead and create an image field as part of the Person content

---

\(^{58}\text{https://www.drupal.org/project/image_field_caption}\)
type-- there’s nothing inherently “wrong” about doing it. Perhaps you want your Image content type to only store historical images, but your team includes an artist who’s drawing stylized portraits of each person for use as their thumbnail image. In such a case, uploading the portrait directly to the Person content type makes sense.

To do the node reference variant, add a “node reference” field; if you don’t see that as one of the field options, make sure you have the References and Node Reference modules enabled as described in section 5.5.2.2.1. The label should be “Thumbnail image” and the widget should be “autocomplete text field”. The number of images on the site will quickly grow too large for either checkboxes or a select list to be practical.

On the next configuration screen, you’ll face a conundrum: you can select what content type(s) can be referenced, and if you’re starting with creating the Person content type, the Image content type you need to choose won’t be an option. This sort of situation is fairly common when working on Drupal configuration: sometimes you can progress through the steps of setting up a site in a linear fashion, and sometimes you have to go back and forth between different aspects of the site configuration, in this case, creating a new content type in order to be able to reference it from elsewhere, and waiting until later to configure that new content type.

At this point, you can skip the checkboxes indicating which content type will be referenced and finish the rest of the configuration for the “Thumbnail image” field, or you can go directly to creating the Image content type. Even though you haven’t hit “Save”, the “Thumbnail image” field will still be there as part of the “Person” content type when you return.

Let’s go directly to creating the Image content type. Go to Structure > Content types > Add content type in the administration menu (or /admin/structure/types/add), and name the new content type “Image”. Then click the “Save content type” button at the bottom of the screen. For now, you can completely skip all the content type settings (e.g. whether or not to show author information) -- the only thing that matters right now is that there exists a content type called “Image” that you can use when configuring your node reference field.

Go back to the Person content type field settings by going to Structure > Content types > Person > Manage fields (/admin/structure/types/manage/person/fields) and click the “Edit” link corresponding to the “Thumbnail image” field59. This time all the configuration options are on a single page. You can enter some help text, explaining that this is a pointer to the image that will be used to represent the person in various places on the site (like timelines). Under “Autocomplete matching”, you may want to stick with the default “Contains” unless you expect your site will have many hundreds or thousands of

---

59 Depending on how recently the administration menu has been updated, you may be able to go directly to the configuration page for the “Thumbnail image” field by going to Structure > Content types > Person > Manage field > Thumbnail image. It can sometimes take a few minutes for the administration menu to update after you’ve created a new field. To speed this up, hover over the house icon in the upper left corner, and choose “Flush all caches”.

84
nodes of that content type. In such a case, choosing “Starts with” will improve the performance of the autocomplete.

The default textfield size is fine, and you probably don’t want to choose a default value. One exception might be if your site will include many people with no known photograph. In such a case, you might want to upload a generic picture (an icon, a silhouette, etc.) to use in those cases, to ensure some uniformity in the site display. In such a situation, you may want to put that generic picture in as the default, and put instructions in the help field telling data entry assistants to change the default image for people with a known photograph. To do this, however, you need to at least create a dummy node of the “Image” content type (which you’ll ultimately use to upload that generic image) before you can point to that node as the default in the “Thumbnail image” configuration screen. You can even create such a dummy node before you’ve added in the fields you’re going to use for the “Image” content type -- including, crucially, the media upload field itself. Much like the “Image” content type, all that matters is that the “Image” node that will ultimately house your default image exists in the database, even if you change its name and add any number of fields later.

The default number of values should be 1, because there should only be a single thumbnail image for the person.

At the bottom of the configuration screen, you’ll once again see the checkboxes with the content types that can be referenced, but now “Image” should appear as one of the choices. Check the box for “Image”, and save.

**Medical institution attended**
The purpose of having this as a field as part of the “Person” content type is to make it easy to sort and group individuals by where they received medical training. There are a number of possible field choices for capturing this data:

- **Text (list)**
  - *Cons:* This is a terrible option; you’d have to pre-populate the options with the name of every single institution that anyone attended for medical training. You’d then have to scroll through all those options when entering a quote (unless you use a module like Autocomplete Widgets.)

- **Text field**
  - *Pros:* you can populate this field as needed, as opposed to having to pre-populate with a text (list) field.
  - *Cons:* it's just text, and other than searching using that text field, it's not actually connected to anything else in the database. This might be less of a problem for some institutions (e.g. if there are institutions that only appear in the database once, connected to one individual), but it limits what you can do with the data. If you wanted to look at all the people who did a medical residency through a hospital associated with the institution where they received medical training, it would be difficult to pull together the data you need. Additionally, a text field provides you with no way to attempt to control for misspellings and variation in the name of an institution, or easily merge entries that use a non-standard name variant into the correct variant.
● Term reference field
  ○ Pros: you can populate this field as needed, and/or seed the list with commonly-referenced institutions. Term reference fields can use an autocomplete widget, to cut down on spelling variation. You can also change the name of the institution (e.g. to use acronyms) after the fact without having to update the nodes that point to that term, since the underlying pointer (to a system-internal identifier for each term) remains the same. It’s also fairly easy to merge spelling variants, if they arise, into a single preferred term. The default display of a term reference field is a link for the term, where clicking on the link takes you to all nodes (People, Images, etc.) that use that term, which is a reasonable and fairly useful way to display the data. You can also easily modify the default display using Views. In addition, if there’s a little bit of extra information you need to store as part of each term (for instance, the geographic location of the institution), you can add fields to taxonomy terms.
  ○ Cons: fields in taxonomy terms are not as easy to access when developing Views that use a lot of data stored in nodes. If you want to store a lot of additional information about taxonomy terms (in addition to geographic coordinates, also when it was founded, an extensive description, images, etc.), you should consider whether this indicates that the information (in this case, information about institutions) is a sufficiently important part of the data to merit being treated as an additional content type, in which case you’d use a node reference field to point to it, rather than a term reference field.

In this case, we don’t have so much information about institutions that we need to treat them as content types. Treating institutions as taxonomy terms and using a term reference field to point to them will be sufficient.

To streamline the configuration process, you should first create a vocabulary called “Institutions”. Go to Structure > Taxonomy > Add vocabulary (admin/structure/taxonomy/add), and name it “Institution”. You can optionally put in a description; this will only appear as part of some administration screens, but it’s helpful for future maintainers of the site to have that information available. In this case, the description would indicate that these are institutions with which people in the database were affiliated in some way; you don’t want to be more specific and say that it’s for institutions where people received medical training. You’ll use this same set of terms for the “Institution” field as part of the “Event” content type, and the relationship between the person and the institution there can vary a great deal.

Once you save your new vocabulary, you could add some terms to help seed it, but since we’ll be using the Autocomplete widget for this term reference field, the easiest way to populate the vocabulary is to just enter institutions as they arise as part of data entry for the Person content type. As such, let’s return to configuring the Person content type Structure > Content types > Person > Manage fields (/admin/structure/types/manage/person/fields).

Add a new field, “Medical institution attended”, and choose “Term reference field” and “Autocomplete term widget”. On the next configuration screen, choose “Institutions” as the vocabulary. Provide some help text if desired, and then save.

Profession
See “Medical institution attended”, above -- the same discussion generally applies here. In this case, a text list field would not be ridiculous, since the list of types of medical professions is much smaller than the list of relevant institutions. One could potentially imagine going to the
other extreme, and having a content type “Profession” that includes long essays on what it meant to be a doctor or nurse, with additional fields for storing bibliographic references to scholarly literature on the topic. In this case, a term reference field pointing to a “Professions” vocabulary (like the “Institutions” one described above) is still probably the best fit, although a text list field would be a reasonable option. It largely depends on whether you want the person’s profession to appear, by default, as a clickable link that shows you all other people with that same profession. If so, use a term reference field. If you want it to just be plain text, you can use a text list.

Create the “Professions” vocabulary, but after you save it, click on the “Add term” link corresponding to “Professions”. Enter all the professions you want to provide as options, one at a time. When you’re done, return to managing fields for your Person content type.

Add a term reference field called “Profession”, but this time choose the “Check boxes/radio buttons” widget. On the next page, choose the “Professions” vocabulary.

On the next configuration page, enter some help text, and allow unlimited values. Some people in the database have more than one professional role, but you don’t necessarily want to guess ahead of time what the maximum potential number of roles is. Making this change will turn the default radio buttons into checkboxes on the node creation/editing form. Save the field.

**Specialization**

Specialization is an intermediate case between “Profession” and “Medical institution attended”. There are enough options -- including nested options -- that the checkboxes used for “Profession” might be cumbersome. Text list fields don’t support parent/child term relationships, so that is not a good option in this case. At the same time, there are few enough options that the autocomplete text widget might be overkill. The autocomplete text widget would also allow people to add new terms, which you may not want.

The best way to store this data is going to be as a vocabulary and taxonomy terms. Create a new vocabulary, as described above, called “Specializations” and populate it with terms as follows:

- **Blah**
  - Stuff
  - Things
- **Argh**
  - More stuff
  - More things

To create parent/child term relationships (e.g. indicating that Stuff is a sub-term of Blah), first create the parent term (e.g. Blah). Then, when creating the child term, click the “Relations” toggle-down, and select the parent term from the list of terms under “Parent terms”.

Don’t worry about putting a numerical value under “Weight”; by default, terms sort alphabetically. If you want to put them in some other order, it’ll be far easier to drag and drop them into that order on the vocabulary overview screen (Structure > Taxonomy > Vocabulary-Name-Here) than to remember which term you assigned what numerical value.

When creating the term reference field for “Specialization”, we don’t want to allow users to add new values to this taxonomy, so that rules out the “Autocomplete text widget”. Using
checkboxes could be awkward, as the list of options would be quite long. The “select list” widget is not particularly intuitive for users selecting multiple options -- they have to remember to hold down the “Control” (Windows) or “Command” (Mac) key to choose a second option without deselecting the first.

There are a few modules that provide additional widgets that might be useful in situations like this. If you want to make the most use of the hierarchy you’ve set up as part of your taxonomy, the Hierarchical Select module and Taxonomy Term Reference Tree modules may be useful. The former walks users step-by-step through the levels of a hierarchy, allowing them to choose among the options available at each level. The latter provides sophisticated checkboxes that more clearly show the levels of the hierarchy, can collapse options beyond a certain level so they’re not all visible and taking up a lot of screen space, and can do things like automatically selecting the parent term(s) of any term that the user selects. The Simple Hierarchical Select module is a lightweight alternative to Hierarchical Select, which also can be used as a user-configurable filter in Views.

In this case, the hierarchy is convenient, but not essential. It’s also fairly limited at only two levels. The autocomplete text widget is close to what we need, other than allowing people to add new terms. As such, the Chosen module is a good substitute. It allows users to type in text, but only choose from the options that already exist in the database.

The Chosen module itself is just a wrapper for a Javascript library (i.e. the necessary plumbing to hook up a user interface element that could be used with other platforms or programming languages, with Drupal specifically). To fully install the Chosen module, you’ll have to upload the Chosen Javascript library by accessing the Drupal codebase directly; you can’t do this part through the Drupal interface. See section 4.4.2.3 Installing modules that use libraries for instructions.

After installing and enabling Chosen (including uploading the Chosen library to sites/all/libraries/chosen), you should see it among the options when you create the term reference field “Specializations”.

7.2.1.3 Automatic nodetitles

One thing on the field settings list that we have not yet addressed is the “Title” field. One option for this field is to rename it to “Name”, but that would mean a lot of repeated data entry, since we already have fields for each name component.

To fulfill Drupal’s title field requirement without doing additional data entry work, enable the Automatic Nodetitles module and the Token module, if you haven’t already, and edit the Person content type (Structure > Content Types > Person). Under “Automatic title generation” (most likely the first of the configuration options at the bottom of that screen), select “Automatically generate the title and hide the title field”. If you don’t see “Automatic title generation”, make sure that Automatic Nodetitles is enabled.

---

60 https://www.drupal.org/project/hierarchical_select
61 https://www.drupal.org/project/term_reference_tree
62 https://www.drupal.org/project/shs
63 https://www.drupal.org/project/chosen
To define the pattern for the title, first click in the text field under “Pattern for Title”. Then, using the token browser table (“Replacement Patterns”) below that text field, click on Nodes, then click on the blue links for each of three fields for the components of the person’s name: [node:field_given_name][node:field_middle_name_s_][node:field_surname]. Go back up to the text field, and you’ll see that each of the fields has appeared in the box. Be sure to put a space between them, otherwise all the names will be smushed together in the generated titles.

If you have the Entity Tokens module enabled, you’ll also have access to similar tokens with hyphens rather than underscores as part of the name. Do not use the variants with the hyphens, at least for the middle name field. If a person does not have a middle name, and you use the hyphenated variant, the literal text of the token will appear as part of that person’s name instead (e.g. “John [node:field_middle_name_s_] Smith”).

7.2.1.4 Pathauto
Go to Configuration > Search and metadata > URL aliases > Patterns (/admin/config/search/path/patterns). If you don’t see “Patterns” as an option, make sure that you’ve installed and enabled the Pathauto module.

Scroll down to the “Content Paths” section, and under “Pattern for all Person paths”, first type people/ then toggle down the replacement patterns and choose the same tokens you did for the Automatic Nodetitles setting: [node:field_given_name][node:field_middle_name_s_][node:field_surname]. This time, instead of putting a space between them, put a hyphen, so that the final value of the “Pattern for all Person paths” field is: people/[node:field_given_name]-[node:field_middle_name_s_]-[node:field_surname]. Scroll to the bottom of the screen and hit the “Save configuration” button.

Using the default tokens, rather than entity tokens (discussed in section 6.7.1), is the best choice here, because not all the fields you’re using may exist for a given field. With default tokens, if the person has no middle name, nothing will appear in the [node:field_middle_name_s_] slot. If you were to use entity tokens, which replace the underscores with a hyphen, someone with no middle name would appear with the text “[node:field-middle-name-s-]” between their first and last name.

Entity tokens will be necessary for configuring the title field for the Image content type; see section 7.2.4.3.

7.2.2 Event
The event content type will store information about events in a person’s life. An event can be shared by more than one person. Create a new content type named “Event”, as you did with “Person”.

7.2.2.1. Content type settings
Under “Display settings”, uncheck “Display author and date information”; it doesn’t matter who enters the information about people.

Under “Comment settings”, set “Default comment setting for new content” to “closed”.

89
Otherwise, the default settings are fine.

7.2.2.2 Fields

Person
This is a node reference field that points to the person or people involved in the event. The content type it should point to is “Person”, and unlimited values should be allowed.

Date
The question of how to store information about the date is directly tied to what you want to do with that information. Here are some options for the types of fields you might choose:

- **Text field** - if you only want to display the date, and don't want to do anything else with it (e.g. sorting events chronologically), you can use a text field. In most cases, much of the point of including the date is to enable sorting, and using a text field means that dates in April will appear first, followed by ones in August, if you type in names of months. Numeric date entry won't necessarily fare much better. In most cases, a text field is not a good choice for storing dates. That said, it can be useful as a stopgap measure if you're using a Partial Date field (see below) -- which does not have support for bulk-importing data. You could temporarily create a text field for the Date to import the data into your site, and have a student assistant edit each Event node to enter that data into a Partial Date field. A text field doesn't discriminate with date formats, so it's a decent way to temporarily store dates with multiple levels of granularity (some with full day / month / year, some with just a year, some with a season and a year). Until all the data is stored in a Date or Partial Date field, however, you'll be limited in what you can do with it.

- **Date field** - the best-supported way to store dates in Drupal. Date fields have integration with Views, which means you can use them to generate chronological listings of events. Date fields are also required by the more sophisticated timeline modules (Views TimelineJS Integration\(^64\) and Timelinr\(^65\)). Additionally, you can import data in bulk into a Date field. The downside is that all dates in a date field have to have the same granularity (year, month, day, hour, minute, and second are all options.) If you choose “year” because that's all the information you have for some of the dates, you won't be able to store month or day information about any of the dates. You can establish project conventions for how to create dummy data for dates where you only know a year (e.g. deciding that people doing data entry should enter “January 1” if the month and day are unknown), and you can set up filters using Views to hide some of the dummy data, but this could lead to confusion if you have any real data that coincides with your dummy data. To address this sort of coincidence, you could add a boolean (single on/off checkbox) field that indicates that the date/month are unknown, but all these workarounds are somewhat awkward solutions.

- **Partial Date field** - the need for flexible date granularity motivated the development of this module, which not only allows you to have dates with different granularity, but also allows you to define time periods (spring, 19th century, etc.) and make those available as options. Partial Date fields can be used for sorting events chronologically, but the only timeline module that Partial Date fields are currently compatible with is Simple Timeline\(^66\). As its name indicates, this timeline is less customizable than the more

---
\(^64\) [https://www.drupal.org/project/views_timelinejs](https://www.drupal.org/project/views_timelinejs)
\(^65\) [https://www.drupal.org/project/timelinr](https://www.drupal.org/project/timelinr)
\(^66\) [https://www.drupal.org/project/simple_timeline](https://www.drupal.org/project/simple_timeline)
advanced options. The fact that it’s currently impossible to bulk-import dates into a Partial Date field is a major drawback.

To store dates as part of the Event content type, we’ll combine two approaches. We’ll use a Partial Date field as the primary way to store information, because not all dates will have day / month / year information. However, because this limits the kind of timeline modules we can use, we’ll also use a Date field to store the year. In this case, the ability to use the TimelineJS module is sufficient payoff for the repeated data entry, as the year will have to be entered twice.

Create a Date field as described in “Birth date”, above, and call it “Year”. For “Date attributes to collect”, just check “year”. Be sure to go into “More settings and values” to set the default value to “None”.

Then, create a “Partial date and time” field, and call it “Date”. On the next configuration screen, toggle down “Minimum components” and choose “Year”. This indicates which fields must be filled in. If you choose any of the “Estimate” options, you can toggle down “Base estimate values” to define ranges for centuries, seasons, etc.

On the next page, under “Time zone handling”, choose “Site timezone”. Enter help text if you’d like (perhaps something about entering as much date information as is available), and at the bottom, choose “Year”, “Month” and “Day” as the date components. This determines what granularity options are made available on the data entry page.

There should only be one value allowed for this field.

**Location**

Location is another piece of data where we’ll need to add two different fields: in this case, a text field for entering the location information, and a geofield for converting that text to geographic coordinates.

Create a text field called “Location”. It should have a single value.

Create a Geofield called “Location coordinates”, with the widget “Geocode from another field”. If you don’t see this option, make sure the Geofield module is installed.

For “Storage Backend”, go with “Default” and move on to the next configuration page.

No help text is needed for this field, because it will be invisible to users. All users will see is the “Location” text field that you created above, and the “Location coordinates” field will do its work behind the scenes.

Under “Geocode from field”, choose “Location” -- the text field that you just created.

For Geocoder, choose “Google Geocoder”. There are other geocoder options available as well, how to configure the Google Geocoder is described below.

With the Google Geocoder, you can reject results that don’t meet a certain level of precision -- for instance, if you want to only store precise street addresses, you can reject everything other than “Rooftop”. Often, issues of precision have more to do with what you put into the “Location”

---

67 See this issue for updates: https://www.drupal.org/node/1660852
field than what Google’s geocoder can find. If your data just says “Philadelphia, PA”, the geocoder will necessarily identify approximate coordinates (often around the city center). In most cases, it’s best to leave all the “Reject results” checkboxes unchecked.

“Multi-value input handling” doesn’t apply here because the “Location” field is set to take one value, but the default setting (Match Multiples) is probably the most sensible, generating one set of coordinates for each value.

The “Location coordinates” field should also only take one value.

**Description**
A description of the event. Rename the default body field “Description”.

**Title**
There is no natural “title” for events, but it will be useful to have a succinct but informative text snippet for each event, for use on timelines. The fuller description of the event can go in the “Description” field. On the content type editing screen (Structure > Content Types > Event), under “Submission form settings”, change “Title” to “Brief description” and save. Keep in mind that the Drupal title field (where the brief description will be stored) is limited to 255 characters.

**Event type**
There are certain kinds of events shared by many or most people in the database (graduation from college, marriage, birth of children, graduation from medical training, etc.), and it may be useful to be able to pull together these common kinds of events across multiple different people. As such, this information is best captured through a term reference field. Create a new vocabulary, “Event types”, and populate it with the following terms:
- Birth
- Death
- Marriage
- Children
- Divorce
- Completion of medical training
- Residency
- Employment

Then, create an “Event type” term reference field that points to that vocabulary, and use the “Select list” widget. If you enrich the taxonomy with a more elaborate way of classifying events, you may want to consider some of the widget options described under “Specialization”. If you’re just using the terms listed above, this field should have one value, but that might change if your taxonomy is more elaborate.

**Institution**
Instead of creating a new field for this, you’ll reuse a field you already created as part of the Person content type. Instead of going to “Add new field” on the “Manage fields” page for the Event content type, look at “Add existing field”, and choose “Term reference: institution_attended (Institution attended)” from the drop-down list. Even though Drupal will pre-populate the title of the field as “Institution attended”, you can change it to just be “Institution”. You can add in new help text (indicating that this is an institution associated with the event).

When you created “Institution Attended” as part of the Person content type, you linked it up with the “Institutions” vocabulary and said that one value was allowed; those configuration settings have to be the same here, as well. You can have multiple term reference fields pointing to the
same vocabulary, so if it’s important that “Institution” take multiple values in the context of an event, you can create a new field instead of reusing the old one. When the configuration of two fields is going to be identical, however, it makes the list of fields on your site less cluttered to just reuse an existing field instead of recreating it.

**Main timeline**
All events associated with a person will appear on that person’s timeline, but some events should also appear on an overall site timeline. A boolean field with a single on/off checkbox will allow us to differentiate between those events that belong on the overall site timeline and those that don’t.

Create a Boolean field called “Main timeline”, with the widget “Single on/off checkbox”.

Leave the on/off values empty and move on to the next configuration page.

Put in some help text indicating the guidelines for what kind of events should appear on the main site timeline.

Underneath the help text box, there’s a small checkbox that’s easily overlooked entitled “Use field label instead of the "On value" as label”. Check that box.

This field should only have one value.

7.2.2.3 Pathauto
Go to Configuration > Search and metadata > URL aliases > Patterns (/admin/config/search/path/patterns).

Scroll down to the “Content Paths” section, and under “Pattern for all Event paths”, first type events/ then use the “Replacement Patterns’ box to select [node:title], so the final value for the “Pattern for all Event” paths field is events/[node:title].

It may be tempting to make the title more complicated, perhaps including some form of the date, and/or the name of the person associated with the event. There’s not a lot to be gained from doing that, in this case. While including more information in URL aliases can be helpful for determining when, for instance, certain information should appear in a sidebar (when configuring block visibility; see chapter 11 for more), users will never be looking at an event node by itself, as a stand-alone page: they’ll only encounter the information in the context of a timeline. In fact, you may ultimately want to configure the Event content type so it can’t be viewed on its own by non-administrators. Recall, too, that an event can be associated with more than one person, and if you include [node:field_person] in the pattern, you can end up with multiple names separated by hyphens (e.g. events/john-smith-jane-smith/event-title), which wouldn’t even be helpful for block visibility.

7.2.3 Image
By following the instructions above, you have already created an Image content type, but you haven’t done anything to configure it beyond the defaults.
7.2.3.1 Content type settings
On the content type configuration page (Structure > Content Types > Image, or admin/structure/types/manage/image), under “Display settings”, uncheck “Display author and date information”; it doesn’t matter who enters the information about people.

Under “Comment settings”, set “Default comment setting for new content” to “closed”.

7.2.3.2 Fields
To a large extent, you’ll be reusing fields you’ve already created in the Image content type.

Profession
The same term reference field you used as part of the Person content type. Include the help text “Select what profession(s) are depicted in the image.”

Institution
The same term reference field you used in the Person and Event content types. Include the help text “Enter the institution(s) depicted in the image, if any.”

Person
The same node reference field you used in the Event content type. Include the help text “Enter the names of the people in the database depicted in the image, if any.”

Image
An image field, using the “media browser” widget.

The upload destination should be to “public files”, and leave the default image field empty. Save the field settings.

Include the help text “Click the “Browse” button below to access the image uploading screen. After you’ve uploaded the image, put the title of the image in the “Name” field. For better accessibility, include the title in the “alt” field as well.”

Under “Enabled browser plugins”, check the box next to “Upload”.

Under “Allowed file types”, make sure that just the “image” box is checked.

You can leave the rest of the fields (for maximum/minimum resolution and size, etc.) blank.

Body
Rename the “body” field to “description” and include help text about entering an extended description of the image.

Title
Because you’ve already entered the title of the image in the image name field as part of the upload process, we’ll use Automatic Nodetitles to pull in that information so you don’t have to enter it again.
7.2.4.3 Automatic Nodetitles
To be able to pull in the information about the file you uploaded, you'll need to enable the Entity API\(^{68}\) and Entity Tokens modules (the latter is part of the set of modules available once you install Entity API). Much like how Entity Tokens was used to pull in more specific date information in section 6.7.1, we will use it here to turn the image name into the node title. If you don't have Entity Tokens installed and instead try the default \([\text{node}:\text{field}\_\text{image}]\) token, nothing will happen, and titles will just appear problematically blank. The \([\text{node}:\text{field}\_\text{image}]\) token is only useful for making the image itself appear, whereas what we need is some text-based metadata about the image.

Back on the content type configuration page, under “Automatic title generation”, select “Automatically generate the title and hide the title field”. Click in the “Pattern for the title” box, and use the token browser to find \([\text{node}:\text{field}\_\text{image}:?]\). Click on it so it appears in the “Pattern for the title” box, and replace the “?” with “file”, so it reads \([\text{node}:\text{field}\_\text{image}:\text{file}]\). A bit counterintuitively, in this case specifying “file” won’t get you the file itself, but will get you the file name, which is what we want.

File names are required, so there’s no risk of getting nodes where the title literally is \([\text{node}:\text{field}\_\text{image}:\text{file}]\)-- this can be a problem if you use entity tokens (the variant with a hyphens instead of underscores in the name, like \([\text{node}:\text{field}\_\text{middle}\_\text{name}\_\text{s}]\) instead of the default token \([\text{node}:\text{field}\_\text{middle}\_\text{name}\_\text{s}]\)) for data-- such as middle names-- that may or may not exist for a given node.

7.2.4.4 Pathauto
Go to Configuration > Search and metadata > URL aliases > Patterns (/admin/config/search/path/patterns).

Scroll down to the “Content Paths” section, and under “Pattern for all Image paths”, first type \(\text{images/}\) then use the “Replacement Patterns’ box to select \([\text{node}:\text{title}]\), so the final value for the “Pattern for all Event” paths field is \(\text{images/}[\text{node}:\text{title}]\).

7.4 Conclusion
In this chapter, we have explored the configuration of content types and their fields. We have discussed important considerations when doing data modeling for your content types: both those connected to the data and your particular research questions, and those connected to constraints and quirks around Drupal’s underlying architecture. We discussed some considerations for when you should create separate content types, and when you should create a single content type that can accommodate slightly divergent data, possibly through the use of conditional fields. This chapter also covered different fields you can use when creating your content types, how to use Automatic Nodetitles to provide a “Drupal title” for content that doesn’t naturally have a title, and how to use Pathauto to configure more user-friendly URLs for your content. Finally, we discussed how to configure the display of your content, with special attention paid to date fields and image fields.

---

\(^{68}\) https://www.drupal.org/project/entity
8. Configuring input forms and adding content

8.1 Overview
Entering data is the best way to validate the content types you have created for your site. Even if you ultimately plan to do a bulk import of your data (see chapter 14), it may be worthwhile to spend some time manually entering a few nodes of each content type to make sure that your data aligns well with the fields you have created. This process also exposes you to the layout and configuration options in the interface for adding and editing content, allowing you to tweak them before making them available to data entry assistants or other contributors to the site.

This chapter will cover Drupal's text input formats, configuring a WYSIWYG editor (which provide a word processor-like interface for text fields), adding inline images, and linking to other nodes on your site.

While it does not apply to our example site, section 8.4 will address how to organize and present fields in groups on the new content / content editing form using the Fieldgroups module.

Section 8.5.1 will cover settings that users with the right permission levels can use to configure node authors and dates, published / unpublished status, and URLs when creating or editing new nodes.

8.2 Text formats
8.2.1 Field configuration
Every text field (including both short and long text fields) has one of two text processing options. When you edit a text field, you'll see radio buttons with the label "Text Processing" (under the box for help text) with two options: Plain text and Filtered text (user selects text format).

For short text fields -- such as the “Given name”, “Middle name(s)” and “Surname” fields in the example Person content type-- it usually makes sense to stick with the default “Plain text”. In the case of those fields, there’s no reason to include any formatting. We may want to apply various kinds of formatting when the names appear on the site in different places (perhaps in bold on a timeline, or as header text at the top of the node that contains the person’s profile), but there’s no formatting that’s inherently part of that data.

The likelihood of formatting being an inherent part of the data that must be captured increases when the text field is larger. If you are using the default Article content type to maintain a blog for your project, chances are that the “body” field of many of the Article nodes might include italics (e.g. for titles of books or films mentioned in the prose), URLs, or even inline images (e.g. a screenshot illustrating progress in the site’s development). For that reason, long text fields such as the “body” field, or other long text fields you create from scratch, have “Filtered text (user selects text format)” as the default text processing setting. If you want to offer users a WYSIWYG interface (similar to word processing software, with buttons that allow them to turn various formatting options on and off), you must select “Filtered text” as the text processing setting.
Once you have fields as part of your content types that use the “Filtered text” settings (i.e. any long text field, including the body field), you should check on text formats available for your site and ensure that users have permission to use them.

8.2.2 Text format configuration
Go to Configuration > Content authoring > Text formats (/admin/config/content/formats). By default, there are three formats:

- **Filtered HTML** - provides a limited subset of HTML elements that can be used within the text. By default, anonymous, authenticated, and administrator users can use this.
- **Full HTML** - does not limit the HTML elements that can be used, but does provide some checks such as correcting some faulty HTML.
- **Plain text** - displays any HTML entered into the field as plain text (e.g. if someone enters “<em>Moby Dick</em>”, it will display as “<em>Moby Dick</em>” rather than “Moby Dick”).

You should make sure that anyone who needs to edit a text field that uses the “Filtered text” text processing format has permission to access at least one text format in addition to “Plain text”, otherwise they’ll see an error message in the field stating “This field has been disabled because you do not have sufficient permissions to edit it.” The default configuration, which gives all three default roles on the site (anonymous, authenticated, and administrator) access to “Filtered HTML”, is correct, but keep this in mind if you make changes.

8.2.3 Configuring a text format
Click the “configure” link next to “Filtered HTML” to see the available options. On this screen, you can change the name of the text format, as well as select which user role(s) can use the text format. Note that you can also change the permissions settings for each of your text formats using the giant Permissions table (People > Permissions), discussed in depth in chapter Z.

You can also select which filters are enabled and, below that, the order in which those filters process the text, via a drag-and-drop table. At the bottom of the screen, there’s another table where you can configure settings for individual enabled filters. For the “Filtered HTML” text format, “Limit allowed HTML tags” and “Convert URLs into links” are enabled by default. The settings for “Limit allowed HTML tags” let you specify which tags you want to allow people to use, and the settings for “Convert URLs into links” lets you specify a cut-off point for the display of very long URLs.

The default settings for allowed HTML tags for “Filtered HTML” is very limited, permitting the creation of links (<a>), italics (<em>), bold (<strong>), citing titles (<cite>), block quotes (<blockquote>), code (<code>), bulleted and numbered lists (<ul>, <ol>, <li>) and lists with descriptions (<dl>, <dt>, <dd>). It does not permit tables, the inclusion of images, using various levels of headers, superscript/subscript, or many other things.

Nonetheless, “Filtered HTML” may be a good text format to use in cases where what you fundamentally want is for users to input a number of sentences of plain text, with minimal formatting or markup. If you can expect that your users will be comfortable using a little bit of HTML to handle things like italics and lists as needed, simply using “Filtered HTML” as the text format for a text field will ensure that your site is minimally affected by bizarre formatting cruft that inevitably results from copying and pasting text written in word processing software (be it Microsoft Word or Google Docs or Open Office) into a text area that attempts to maintain
existing formatting. Text pasted into a simple “Filtered HTML” field will have any existing formatting stripped out, and users will be quite constrained in terms of what formatting they can put back in using HTML.

However, if your users are not comfortable entering any HTML, if they expect that formatting from the source document appear along with their text when they copy and paste, or if they would prefer a more word processor-like interface, you will need to install and configure a module that provides a WYSIWYG (What You See Is What You Get) interface, as well as some supporting modules that address things like image upload and text format; see section 8.3.

8.2.4 Additional filters
While they may only be relevant for certain circumstances, these modules provide additional filters that you can enable and configure (if applicable) for individual text formats:

- **Footnotes**[^69] - allows you to insert footnotes on the page that appear at the bottom. The superscript numeral (or other, user-defined characters) of the footnote link to the footnote text, and the corresponding numeral or characters in the footnote links back to the place in the main text.
- **BeautyTips**[^70] - allows you to create definitions, explanations, etc. that appear in a hovering bubble next to specified words.
- **Table of contents**[^71] - makes it possible to insert a table of contents anywhere within a text field, which makes use of header (<h1>, <h2>, etc.) tags to define the structure of the table of contents.
- **Video filter**[^72] - allows you to easily embed videos from YouTube and many other providers into pages. (The default embed code from those sites often uses an <iframe> tag which Drupal will filter out, preventing the video from displaying.)
- **SpamSpan filter**[^73] - obfuscates email addresses using Javascript to reduce the likelihood of them being picked up by spammers. Could be useful if, for instance, you have a page of project staff that includes their email addresses.
- **Collapse text**[^74] - allows you to have sections of text that are hidden, and must be expanded to be visible. Could be useful for a pedagogically-oriented site with homework or quiz questions where students can check their own work.
- **Code filter**[^75] - if your site includes snippets of code, this filter allows you to use PHP and <code> tags without having to change all < characters to &lt; for them to display correctly.
- **Caption filter**[^76] - allows you to easily add captions to images, similar to how captions are done in WordPress.
- **Paging**[^77] - makes it possible to break up a very long text into multiple “pages”, without having to create multiple nodes.

[^69]: https://www.drupal.org/project/footnotes
[^70]: https://www.drupal.org/project/beautytips
[^71]: https://www.drupal.org/project/tableofcontents
[^72]: https://www.drupal.org/project/video_filter
[^73]: https://www.drupal.org/project/spamspan
[^74]: https://www.drupal.org/project/collapse_text
[^75]: https://www.drupal.org/project/codefilter
[^76]: https://www.drupal.org/project/caption_filter
[^77]: https://www.drupal.org/project/paging
8.3 WYSIWYG configuration

There are numerous modules that provide a WYSIWYG interface for editing text. If you just want an interface that provides familiar buttons for different kinds of formatting, linking, etc., it will likely be easiest to install and configure one such module by itself. CKEditor\textsuperscript{78} is the WYSIWYG module that will be included by default in Drupal 8. The default configuration for CKEditor doesn’t require you to install a library; instead, it points directly to the code on the CKEditor server. This conveniently saves some configuration, but it only works for sites that are connected to the internet (i.e. not for a site that you’ve set up to run on your own laptop, with the goal of being able to work on it on airplanes, etc.) You can find detailed instructions for installing and configuring CKEditor on its own on the CKSource website\textsuperscript{79}. (The website also talks about the enterprise version of CKEditor that you can purchase, but you don’t need it.)

For our example site, we want a WYSIWYG interface that integrates with the Media module, to make it easier for us to upload new images to display inline (e.g. to illustrate a blog post or an essay), and to reuse images that have been uploaded to the site. This is considerably more difficult with the standalone CKEditor module. Instead, we will be using the WYSIWYG module\textsuperscript{80}, which provides a framework for installing different WYSIWYG libraries. The WYSIWYG module can provide the same type of integration “plumbing” for the CKEditor library that the CKEditor module does, but it’s more flexible.

8.3.1 Installing modules and CKEditor library

Download, install and enable a version of the WYSIWYG module released after October 2014; currently, this means using the dev version of the module. A major bug was fixed in October 2014 that addresses a problem with the module being incompatible with the latest release of the CKEditor library, but the fix has not yet been incorporated into a non-dev release\textsuperscript{81}.

Also enable the WYSIWYG Media module, which is part of the Media package of modules.

Download the “Full Package” CKEditor library, from http://ckeditor.com/download. When you unzip it, you should get a folder called “ckeditor”. Place that folder into the libraries directory in your Drupal code base (/sites/all/libraries), so that it’s available as sites/all/libraries/ckeditor.

8.3.2 Selecting CKEditor in the WYSIWYG module

Go to Configuration > Content authoring > Wysiwyg profiles (/admin/config/content/wysiwyg); clear the Drupal cache, by hovering over the house icon in the upper left and selecting “Flush all caches”, if you don’t see this option.

If you’ve correctly installed the CKEditor library, you should see a screen that lists all of the text formats on your site, along with a drop-down for each that currently says “No editor”. If you toggle down the “Installation instructions” option, you should see CKEditor at the top, in green, with the version of the library that you installed.

If you haven’t correctly installed the CKEditor library, you won’t see the list of text formats, and instead the “Installation instructions” section will display all the WYSIWYG libraries compatible

\textsuperscript{78} https://www.drupal.org/project/ckeditor
\textsuperscript{79} http://docs.cksource.com/CKEditor_for_Drupal/Open_Source
\textsuperscript{80} https://www.drupal.org/project/wysiwyg
\textsuperscript{81} https://www.drupal.org/node/1853550
with the module, along with instructions for where you should put the library. If you see this screen after you’ve tried to install the CKEditor library, make sure that you’ve unzipped the CKEditor library file you downloaded (you need to upload to your Drupal code the folder you get when you unzip that file, not the file itself), and make sure that it’s in the sites/all/libraries folder.

If you’ve successfully installed the CKEditor library and see the list of text formats, use the drop-down to select CKEditor for “Filtered HTML”, and save.

8.3.3 Configuring CKEditor for the Filtered HTML text format

Once you’ve selected CKEditor for “Filtered HTML”, an “edit” link will appear on the WYSIWYG configuration screen. This screen lets you select which buttons will appear as part of the WYSIWYG interface. While you can check however many boxes you’d like, you may need to edit the list of permissible HTML tags allowed by “Filtered HTML” if you want to go beyond the very small default list (see section 8.2.3).

The following buttons are all compatible with the default HTML tag list for the “Filtered HTML” text format:

- Bold
- Italic
- Bullet list
- Numbered list
- Indent
- Outdent
- Link
- Unlink
- Source code

If you expect that users will be copying and pasting text from Microsoft Word, you should check the box for “Paste from Word”. This will strip out much of the formatting that automatically comes with text that’s been copied from Word, which would otherwise make the text display in an odd way.

You should also select the “Media browser” button, to enable integration with the Media module, and uploading and displaying inline images.

For the additional configuration options at the bottom, the default settings are fine. Save the configuration. You should see a warning message: “The Convert Media tags to markup filter must be enabled for the Filtered HTML format in order to use the Media browser WYSIWYG button.” Click the link on “Filtered HTML format” to go to the configuration page for that text format, enable the “Convert Media tags to markup” filter, and save the configuration. This filter makes it possible to display inline images uploaded by means of the Media module, even without adding <img> to the list of acceptable tags.

If you want a fuller range of formatting options to be available (for instance, to administrators), you can repeat the process for the “Full HTML” text format. First select CKEditor from the drop-down list for “Full HTML” on the WYSIWYG configuration screen (Configuration > Content authoring > Wysiwyg profiles), then choose the buttons you want to include. If you include “Media browser”, you’ll again have to edit the “Full HTML” text format itself to include the “Convert Media tags to markup” filter.
8.3.4 Facilitating internal linking

Most of the pointers from one node to another on the example site (e.g. from an Event to a Person) will be by means of node reference fields, which provide an autocomplete interface for creating such pointers. Sometimes, though, it can be useful to be able to link to other pages on your site just as links within text. If this is something you expect you'll need to do often, you might want to use the CKEditor Link\(^{82}\) module, which provides a more sophisticated interface for the CKEditor “Link” button and makes it easier to link to other pages on the same site.

Install and enable the CKEditor Link module, then go to the WYSIWYG configuration screen (Configuration > Content authoring > Wysiwyg profiles). Edit the profile where you want to use the CKEditor Link module, and check the box for “CKEditor Link”. Save.

This doesn't create a new button, but rather, improves the interface for the “Link” button (which you should also have enabled). Now when you click the “Link” button, it defaults to “Internal path”, and provides an autocomplete box for choosing the page you want to link to. You can also change the “Link type” to URL to put in an external URL.

8.4 Organizing content editing forms

Take a look at what the content creation / editing interface looks like for your content types by going to Content > Add content > Your content type. If you’re happy with what you see, you can skip this section.

Depending on your data model, you may have some complex content types with many fields that may not apply to all pieces of content that will be stored in that content type. If this is the case for one or more content type, be sure to take a close look at the content creation / editing interface for those content types. Will users, or data entry assistants, find it easy to understand what fields to fill out? Does the ordering of the fields follow a fairly intuitive progression?

If you’re not happy with the editing interface for one or more content types, consider using field groups (see section 6.5.4 for configuration details) to organize related fields into groups that appear as vertical or horizontal tabs, or collapsible sections.

Screenshots!

Field groups are good if you want to make all fields available, but not have them stack up so that the user has to scroll extensively to fill out the content type. If you have fields that are clearly only relevant for some kinds of content, consider using conditional fields (see section 6.6 for configuration details) to hide fields that aren’t applicable to certain subsets of content. For instance, if you have a Person content type that uses a term reference field or a text list to differentiate “Student” from “Faculty”, you can use conditional fields to make any fields that apply only to students appear only after a user has specified that the person content they’re creating is a student.

A less common way of breaking up the display of long content types is to turn them into multi-step forms. The Multistep module\(^{83}\) provides a new type of field group, “Step”, that turns the content type creation / editing form into a set of sequential pages. Multistep does not currently

\(^{82}\) https://www.drupal.org/project/ckeditor_link
\(^{83}\) https://www.drupal.org/project/multistep
have a recommended Drupal 7 release, but nearly a thousand sites use the development version.

8.5 Adding content
Finally, it is time to begin adding content to your site. If possible, use real content (not fake placeholder text) when trying out the content types -- at least for the content types that are meant to store the core “data” of your site (e.g. information about people, events, places, etc. where there are multiple fields, in contrast to blog posts and narrative essays where the content types consist mostly of a title and body field.)

Go to Content > Add content > Your content type for each content type you will use on the site, fill out the form, and save. Depending on the node reference fields used by each content type, you may need to do this in a particular order (e.g. creating Person nodes before creating Event nodes so the Events can have a Person to reference).

8.5.1 Configuration options
When creating a new piece of content, take note of the set of options at the bottom of the content creation / editing form. These should all be available to you if you’re logged in using an account with the Administrator role.

- Menu settings - allow you to add the content you’re creating to one of the menu(s) specified on the content type editing screen (see chapter 10 for more on menus).
- Revision information - Drupal can store multiple revisions of a piece of content, allowing you to compare and revert to previous versions. This isn’t really relevant when you’re creating a new piece of content, but it does matter when you edit content: should Drupal simply overwrite the old version, or store your changes as the new version? See section 8.7 for more about revisions.
- URL path settings - once you’ve configured Pathauto (see section 6.9), a box will be checked here for “Generate automatic URL alias”. If you want a page to deviate from the pattern you set up in Pathauto, you can uncheck the box and type the URL path you want to use in the box below.
- Comment settings - defaults to the setting defined for the content type, but you can change it on a node-by-node basis (e.g. turning off comments for a blog post on a particularly sensitive topic).
- Authoring information - by default, the user who is logged in is set as the author of nodes they create. This section allows you to specify some other user as author instead. The autocomplete field for “Authored by” is based on a user’s login name for the site, which you may not know. Using the Real Name module84 (discussed in section 10.4.1) can let you search by a field you’ve created to store a user’s actual name.
  - The “Authoring information” section also allows you to change the date stored in the database for when the node was authored. If you leave this blank, it will store the time and date when you first hit “save”. Note that putting in a future date does not postpone publication until that date, as you might expect from WordPress. Pre-scheduling the future publication of nodes requires the Scheduler module85. For more complex scheduling and workflow needs, the Workflow module86 is an alternative.

84 https://www.drupal.org/project/realname
85 https://www.drupal.org/project/scheduler
86 https://www.drupal.org/project/workflow
• Publishing options - The “promoted to front page” and “sticky at top of lists” options aren’t very useful, but the “published” checkbox is quite important. A node that is saved with the “published” checkbox checked is, by default, publicly available. (You can configure the permissions settings so that only authenticated users can see content on the site; see chapter 10 for details). A node that is saved with the “published” checkbox unchecked is a draft. By default, drafts are only visible to the user who created them (whose username is in the “Authored by” field). If you want other users (or some subset of other users, such as instructors or project managers) to be able to view drafts, install and enable the View Unpublished module⁸⁷, which provides a new set of checkboxes on the Permissions screen to allow people to view unpublished (draft) nodes. By default, unpublished nodes are filtered out of Views of content (see chapter 12 for more on Views).

8.5.2 Previewing content
Once you’ve entered data and made any changes to the node configuration, you can choose to either save the node, or preview it. The “preview” is a bit deceptive: while it will show things like text formatting (e.g. whether your text format has been configured correctly so as to not filter out the formatting done in your WYSIWYG editor) and the presence, absence, and/or alignment of field labels (see chapter 9), it is far from an accurate representation of what your content will look like once it has been saved. Many things, from the positioning of images uploaded to image fields to the font and color of text, are determined by your site’s theme. The node preview doesn’t reflect your site’s theme. For a more representative preview, you can install and enable the Page Preview module⁸⁸, which replaces the default preview (using an iframe—a webpage embedded within another webpage) that comes much closer to an accurate preview. Page Preview is currently only available as a -dev version, but because of the limited way it interacts with the database, it should be safe to use.

8.6 Viewing and editing content
When you save your content, you will probably be disappointed -- particularly for content types with many fields. By default, field labels appear above the content, which leads to a very awkward-looking display. Chapter 9 will cover how to improve content display.

When you’re logged in as an administrator (or a user account with the right set of permissions) and looking at a saved node, you’ll see a set of tabs right above the content: “View” and “Edit”. If you’re looking at a node with at least one revision, you’ll also see a “Revisions”. To edit a node, click the “Edit” link; this will make the editing form appear as an overlay.

8.7 Revisions
To make it easier to track changes on your site, particularly with multiple project collaborators, it may make sense to install the Revision All module. While you should still make adding a revision log message (in the “Revision information” tab at the bottom of the node creation/editing page) part of your data entry workflow, “Revision All” ensures that each change is saved as a new revision whether or not people remember to include a log message.

⁸⁷ https://www.drupal.org/project/view_unpublished
⁸⁸ https://www.drupal.org/project/pagepreview
Install and enable Revision All and click the “Settings” link for the module (or go to Configuration > Content authoring > Revision all). There, you can choose to enable revisions for all content types, or individual ones. There are also options for proactively enabling Revision All for all future content types, and preventing individual content types or individual nodes from disabling saving as a revision. If you don’t check “Prevent Node Revisioning Overrides”, the “Create a revision” box on the node creation / editing form is checked by default, but a user can uncheck it. If you do check it, the “Create a revision” box is both checked and grayed out, with a note indicating that it is disabled due to Revision All.

When viewing an individual node with revisions, the “Revisions” tab lists all revisions, the author of each revision, and the date of the revision. Clicking on the date allows you to view the revision. There are also links for deleting revisions, and reverting to a previous revision. However, by default Drupal lacks the ability to usefully compare revisions, the way you might find on Wikipedia.

The Diff module provides this missing feature by adding a column to the “Revisions” tab where you can select two revisions, and click the “Compare” button to see the changes highlighted.

9. Node display

9.1 Overview
The way fields are displayed by default in Drupal—with the field label above the content—is usually not what you want. Additionally, depending on the kinds of fields you’re using (images, dates, locations etc.) there may be a variety of options to choose from in terms of how the data you enter is displayed.

This chapter will cover how to improve the display of content stored in nodes, including hiding or changing the alignment of field labels, defining and using image styles, and repositioning fields into sidebars or other areas of a page.

9.2 Configuring node display
Node display is configured on a per-content-type basis. To configure the display of the Person content type on our example site, for instance, go to Structure > Content types > Person > Manage display.

9.2.1 Label and field visibility
On this configuration screen, you can specify where the label (the title of the field) appears above the corresponding data, inline (next to it), or whether it’s hidden. Similarly, you can hide the content of various fields. This is useful for our example site, where the person’s given name, middle name, and surname are already appearing as the node title (as we configured it using the Automatic Nodetitle settings). We don’t need them to appear additionally as individual fields, so we’ll hide the “Given Name”, “Middle Name(s)” and “Surname” fields. The most efficient way to do this is to immediately set the “Format” to “<Hidden>”. This automatically pulls the field to the “Hidden” section at the bottom of the screen. When you hide the content of a field, the label for the field is automatically hidden as well, but the same does not hold in reverse, and for good reason. In many cases, you might want to hide the “Body” label of a body field (in fact, this is the default behavior for the body field, even if you rename it) while still displaying the contents.

89 https://www.drupal.org/project/diff
9.2.2 Labels and visibility for Person content type
All labels should be inline, with the exception of “Biography” and “Image”, where the label should be hidden.

The given, middle, and surname fields should be hidden.

Arrange the visible fields in the following order:
- Image
- Birth
- Death
- Profession
- Specialization
- Institution attended
- Biography

9.2.3 Labels and field visibility for Event content type
In a sense, the display of the event content type doesn't matter. The event content type is intended to store data that will be displayed to users through a View, never by itself. If you were to configure events for individual display, you’d want to hide the “Year” field (because the “Date (partial)” field would have a more complete representation of the date) as well as the “Main timeline” field.

You would want to hide the label for “Date (partial)”, in part because the parenthetical annotation “Partial” would be more confusing than helpful for viewers. By default, there is no way to have one label for a field on the data entry screen and another for the display, but the Field Display Label module[^90] provides that feature via an additional option on the configuration screen for each field. Changing the label of a field as part of the display is also a feature of the Display Suite module[^91], which gives you much more control over node display. For more about Display Suite, please visit the Drupal for Humanists website.

“Person”, “Event type”, “Location” and “Institution” should have inline labels, and the label should be hidden for “Location coordinates”.

9.2.4 Labels and field visibility for Image content type
The label should be hidden for the “Description” field (this is configured by default, since “Description” is a renamed “body” field) and the “Image” field. “Profession”, “Institution” and “Person” should have an “Inline” label.

Arrange the visible fields in the following order:
- Image
- Person
- Profession
- Institution
- Description

[^90]: https://www.drupal.org/project/field_display_label
[^91]: https://www.drupal.org/project/ds
9.3 Configuring date fields

Date fields (“Birth” and “Death” in our “Person” content type) have a number of display options to configure. The format “date and time” is a good fit here, although there are additional options such as “time ago” (which will state how long ago the person was born, in years and months by default, though you can add additional units) and “plain” (which simply provides the format “1872-05-13 00:00:00”, for May 13, 1872, with no time specified). Using the “date and time” format, you can specify which of the default formats to use, or a custom format if you’ve created one.

9.3.1 Date fields in the example site

To get Drupal to display the date in a full, written-out format (such as May 13, 1872), go to Configuration > Regional and Language > Date and time > Formats > Add format and enter: F j, Y which is the PHP date shorthand corresponding to “May 13, 1872”. Then go to the Date and Time settings (Configuration > Regional and Language > Date and time) and use the dropdown to change the “Long” date type to use that format.

Because date fields are automatically configured to use the “Long” format, updating the “Long” format to the setting you want fixes the display of all your date fields. If you wanted to use a date format other than “Long”, you’d just need to click on the cog on the far right of the date field you were configuring, and select the date format from the dropdown that appears.

The exception is the “Year” field in the Event content type. That date field is part of the content type to give you more flexibility in your choice of timeline modules to use. The date field that will be displayed is the partial date “Date (partial)” field, and the “Year” field is redundant. For that reason, hide both the label and the content of the “Year” field.

9.3.2 Configuring partial date fields

You can set up “short”, “medium”, “long” and “custom” partial date formats at Configuration > Regional and language > Date and time > Partial date formats (/admin/config/regional/date-time/partial-date-formats). For each one, you can configure the following things:

- Uppercase or lowercase for AM/am and PM/pm in times
- Whether to use BC/AD or BCE/CE notation for years, and whether to mark only dates prior to the year 0, or both those before and after
- How different components should be separated (e.g. through the use of “/” to separate month, day and year, and “:” to separate hours and minutes)
- The order in which the components should appear, arranged through a drag-and-drop table
- How each component should be formatted; for instance, for “Day” the options are:
  - Day of the month, 2 digits with leading zeros, 01 through 31
  - Day of the month without leading zeros, 1 through 31.
  - Day of the month, 2 digits with leading zeros with English ordinal suffix.
  - Day of the month without leading zeros with English ordinal suffix.
  - A full textual representation of the day of the week.
  - A textual representation of a day, three letters.
- Numeric representation of the day of the week 0 (for Sunday) through 6 (for Saturday).

Back on the “Manage display” screen for a content type with a Partial Date field (such as Event), clicking on the cog for that field allows you to choose a date format other than “Medium” (the default), or choose the short description, long description, short or long description (either one, preferring the short if available), or long or short description (either one, preferring the long if available.) The non-date options only make sense if you selected “Short date text” and/or “Long date text” as possible components when you created the Partial Date field. In most cases, you will simply select the “Use date only” option and choose one of the date formats.

9.3.3 Partial date field in the example site
The example site has a partial date field as part of the Event content type which, as noted earlier, is not meant to be seen by users by itself. For that reason, the default display for the partial date field (using the “Medium” format) is fine.

9.4 Configuring geospatial fields
There are two geospatial-related fields in the configuration we have used: the text field that is used for geocoding (where you enter the name of the location), as well as the geofield that stores the coordinates. In most cases, it makes sense to display both of them: the text field can provide a label of sorts for the geofield, which will display as a map.

In many cases, it makes sense to keep the label for the text field, and have it display inline. The format of the text field can remain set to “Default”. The label for the geofield should be hidden. The default format for geofield display is Well-Known Text, which provides one kind of textual representation for the geographic coordinates. Instead, choose “Geofield map”. The default settings provide a map that takes up 100% of the width of the screen, and is 300 pixels high. You may wish to adjust this depending on how you want the node to display (for instance, to 600 px wide and 400 px tall, for a medium-sized rectangular map display.) You can also change other configuration settings, for example, if you want the map to be zoomed further in or out.

The “Geofield map” display is based on Google Maps, and the standard Google Maps configuration options are available (e.g. choosing between a street map, terrain map and satellite map). If you have the Leaflet module enabled, you can choose a “Leaflet” display which provides a different set of options, including choosing from among a much broader array of map types. You can install the Leaflet More Maps module to access additional maps, including a number of artistically rendered maps.

9.4.1 Geospatial fields in the example site
On the example site, geospatial fields appear only in the Event content type, which won’t be displayed to users. For the sake of practice, set the label for the “Location” text field to be inline. Hide the label for the “Location coordinates” field, and set it to display as a Geofield map, 600 px wide and 400 px tall. Leave the rest of the settings with their default values.
9.5 Configuring image fields
The configuration of image field display is closely connected to the concept of image styles. By default, the version of the image that’s displayed as part of the node is the version that was uploaded. This is often inconvenient, particularly if you want to upload the highest-resolution version of the image, but display something smaller and more manageable as part of the node itself. Or perhaps you want to standardize the display, ensuring that the image is the same dimensions across all Person nodes. If you click the cog next to the image field, a dropdown list appears with a set of image styles to choose from. However, you are not limited to the default set of options.

9.5.1 Configuring image styles
Drupal comes pre-configured with three image styles: Thumbnail (100 x 100 pixels), Medium (220 x 220) and Large (480 x 480). The Media module adds another image style, Media thumbnail (also 100 x 100 pixels, but cropped to a square). The three default formats use the “Scale” transformation effect—the largest dimension (either horizontal or vertical) is scaled to the largest acceptable size (100, 120, or 480 pixels, respectively), and the smaller dimension is whatever value retains the original proportions. Media thumbnail uses “Scale and crop”, resizing the smaller side of the image to 100 pixels, and cropping the larger side to the same. When configuring the display of an image field, you can choose from one of these four options, or you can configure additional image styles.

To modify these default image styles, or add your own, go to Configuration > Media > Image styles. To add a new style, click the “Add style” text, or you can use the Administration Menu to go directly to Configuration > Media > Image styles > Add style. Give the style a name; in this case, we’ll choose “Medium square”. On the next configuration screen, you can start adding effects. In order to attain a uniformly-sized square image style, regardless of the dimensions of the original image, we’ll choose the “Scale and crop” effect from the drop-down list and hit “Add”. We’ll define the maximum height and width, 300 pixels each, and click “Add effect”. The preview image with the hot air balloons shows what will happen: the smaller dimension will be scaled to 300 pixels, and the larger dimension will be cropped, resulting in a 300 x 300 square.
After you’ve finished defining this image style, it will be available among the options for displaying the image field on your content type.

9.5.2 Advanced image style configuration
The Imagecache Actions\(^\text{92}\) module provides additional effects that you can use when defining image styles, including watermarking with a transparent image, overlaying text (including text from tokens, as used in Pathauto and Automatic Nodetitle configuration), color shifting, desaturating (making the image black-and-white), posterizing (limiting the number of colors), and others. Depending on your project, this module can save you a considerable amount of time on image preparation, as you can define image styles that can automatically process all images in multiple ways.

9.5.3 Colorbox
The Colorbox module\(^\text{93}\) provides a way to display larger images in an overlay over the page when you click on a smaller version of the image. Its overlay functionality can also be used to display video, audio, external links, etc., but here we will focus primarily on its use for images.

After you have installed and enabled the Colorbox module, you can choose it in lieu of the “Image” format on the “Manage display” page. It has the following configuration options:
- Content image style: the image style used to display the image on the node page
- Content image style of first image: should be the same as “content image style” in most cases. If you are configuring an image field where you can upload multiple images, you can use this field to make the first image appear bigger (e.g. using the Medium default

\(^{92}\) https://www.drupal.org/project/imagecache_actions
\(^{93}\) https://www.drupal.org/project/colorbox
style), while setting the “Content image style” to be something smaller (like the Thumbnail default style), to avoid having too many big images on the page.

- Colorbox image style: the size the image should be when it appears in the overlay, after a user clicks on the image. Should be a size larger than the “Content image style”.
- Colorbox gallery type: by default, Colorbox is set up to provide a gallery -- allowing users to click through to the next image from within the overlay. In the context of configuring the display of a single node “per post gallery” and “per page gallery” will both create a gallery based on all the images uploaded to all the fields in the node. There are other circumstances where you can use Colorbox (e.g. in Views or Panels) where a “page” may draw content from multiple nodes. If you have a content type with multiple image fields that each allow multiple image uploads, and you want each image field to have its own gallery, use the “per field in post/page” gallery setting. If you don’t want the gallery option, set it to “No gallery”. If there is only one image, this setting is irrelevant.
- Caption: when you hover over an image in a colorbox overlay, by default, a text caption appears. This setting specifies the source of that caption. When you’re using the Media module for image upload, “title” is the best option.

9.5.4 Image fields on the example site

In the Person content type, for the “Image” field, use the Image format and the “Medium square” image style.

In the Image content type, for the “Image” field, use the Colorbox format. Set the content image style to “Large”, and the colorbox image style to “Original”. Use “Title” for the caption. Since there is only one image uploaded per image field, you can ignore the gallery setting.

9.6 Configuring file fields and multimedia

File fields (i.e. fields where you upload PDFs, Word documents, spreadsheets, audio, video, or embedded content from third-party providers like YouTube, etc.) have an extensive, but not particularly useful, set of display options by default.

- Generic file: provides an icon associated with the type of file, with the title of the file linked.
- Table of files: creates a table with an “Attachment” column that looks like the “Generic file” setting, and a “Size” column that shows the size of the file.
- URL to file: provides the text of the file’s URL. It is not linked, so it’s a poor choice for using to display the file; it is useful in other contexts, particularly Views.
- Rendered file: looks the same as the “Generic file”, but allows administrative users to edit or delete the file without having to edit the node first. Hover your mouse over the link to the file, and a small cog appears that you can click to choose “edit” or “delete”.
- Download link: allows you to choose the text for the link users will click to download the file.

The Media module provides some additional options:

- Audio: provides an audio player; if the file is not an audio file, however, this will appear blank
- Video: the same as audio, but for video files
- Large filetype icon: a large, gray icon that differentiates between documents, audio and video.
In most cases, the Generic file option is a sensible choice, as its icon indicates the filetype, and users can easily download the file.

If you want to make it easier for users to preview files, or simply view them in the browser, the PDF reader module\textsuperscript{94} provides that functionality. Once you install and enable the module, a “PDF reader” format becomes available for many file types\textsuperscript{95} on the Manage Display page. You can specify how big the viewer should appear, the renderer (Google Viewer is the default, but two other viewer options are available), whether to show a download link, and whether to use Colorbox to have the viewer appear in an overlay (if Colorbox is installed). If you select Colorbox, be sure to include text in the “Colorbox text” field; this is the text users will click on to access the overlay. There is currently a bug in the PDF reader module that hides the download link if you select both it and colorbox, so don’t use the colorbox option if you want people to be able to download the file\textsuperscript{96}.

9.7 CCK Blocks
Not all the elements of a node page can be configured through the “Manage display” interface for the content type. There is no option for configuring where and how the Drupal title is displayed, for example, nor for configuring where and how the author/date information (if enabled) appears. Comments will appear at the bottom of the node, regardless of any configuration done here. To configure your node displays with that level of granularity, the Display Suite\textsuperscript{97} module provides you with significantly more options (including setting up different displays in different contexts, such as search results vs. RSS feed teasers), albeit with a steeper learning curve.

Another module that may be useful when configuring node displays is CCK Blocks\textsuperscript{98}. CCK (Content Creation Kit) was the module that allowed you to add fields to your content types in Drupal 5.x and 6.x. When you enable CCK Blocks, it provides an additional option as part of your field configuration screens, allowing you to indicate whether or not you want that field to be provided as a block. (For more on blocks, see chapter 11.) If you choose to use the block display for a field (for instance, in order to put that field’s content in a sidebar), you’ll want to set both the label and the field to “hidden” when configuring the node display, so the content isn’t repeated.

If you need data from multiple fields to appear as a block (e.g. birth date \textit{and} death date), you can use the Views module (discussed in chapter 12) to generate a block that combines both pieces of data, rather than creating two blocks with CCK Blocks.

\textsuperscript{94} https://www.drupal.org/project/pdf_reader; see also this comparison list of modules with similar functionality: https://www.drupal.org/node/1781960
\textsuperscript{95} .PDF, .DOC, .DOCX, .XLS, .XLSX, .PPT, .PPTX, .PAGES, .AI, .PSD, .TIFF, .DXF, .SVG, .EPS, .PS, .TTF, .XPS, .ZIP, .RAR
\textsuperscript{96} A user in the forums has provided a snippet of code that you can manually add to the module that purports to fix this: https://www.drupal.org/node/2315113.
\textsuperscript{97} https://www.drupal.org/project/ds
\textsuperscript{98} https://www.drupal.org/project/cck_blocks
10. Users and permissions

10.1 Overview
Most sites have more than one user account, if only because it's good practice for every person who works on the site to have their own. Drupal offers minimalist user profiles by default, but user profiles can be built to be almost as complex as content types. The skills you developed when creating content types can be applied directly to user profiles.

You can create roles to group users for the purpose of assigning them greater access to the site -- its content, its administration, or both. This is accomplished using the permissions screen, an enormous table that grows bigger with each module you install, and every content type, taxonomy and file type on your site.

If you have anything on your site open to the internet at large, you should install some kind of spam protection before launching your site.

10.2 User configuration settings
Configuration settings related to users can be found in two different places. Under People (admin/people) you can view all the user accounts on the site, make changes to them, create and configure roles (groups of permissions), and assign roles to users.

Under Configuration > People (admin/config/people) you can change the site's "account settings", which include everything from who can create an account to what fields appear (and how) on user profiles. There are important changes to make in both groups of settings, and each will be addressed individually.

10.3 Account settings
The account settings area (Configuration > People > Account settings, or /admin/config/people/accounts) allows you to specify how user registration can happen, the language of various automated system emails related to users, user signatures, user pictures, and a few other miscellaneous settings.

The default configuration for anonymous users, contact settings (if you have the core “Contact” module enabled), and administrator role are fine in most cases. To cut down on spam, it is essential that you change the default configuration under the “Registration and cancellation” section.

10.3.1 Registration and cancellation
This section of the “Account settings” page lets you configure the process for user registration, as well as account cancellation.

10.3.1.1 Account registration
These settings are very important. If your site isn't one where most people should be creating user accounts (e.g. a public web presence for an organization, where only the people who administer the site have any reason to log in), you should change the settings so that only administrators can create an account. Even if you have no visible link to the user registration page on the site, spammers know the URL for the Drupal user registration page, and you may receive emails from spammer-created user accounts that are pending administrator approval.
On the other hand, if you have a site where you want people to create accounts, having administrator approval is an additional burden on you and on them. In those cases, you may want to change the setting to allow anyone to create an account, but before you do so, make sure the user registration form is protected by an anti-spam module; see section 10.6.

10.3.1.2 Account cancellation
Chances are, most of the time you'll be cancelling an account because it's a spam account. If you have site assistants who are no longer working for you, it's better to change their account status to "blocked"-- which prevents them from logging in, but doesn't do anything to content they've created. The option "Delete account and its content" will remove both the spam account and any spam the user has created.

10.3.2 Personalization (user avatars)
This section allows you to enable signatures and user pictures (i.e. avatars) for users.

The signature option is of very limited utility, and only appears appended to comments that the user posts-- and not any other kind of content they create.

The user picture option allows users to upload a photo as part of their profile. By default, that photo will appear on nodes that the user creates, along with their name and the date, if you've enabled "Display author and date information" for that content type. You can also make use of user pictures as part of views you create (see chapters 12 and 13).

The configuration options here allow you to specify a default user picture for users who don't upload their own picture, as well as the default display style, maximum file size and resolution for pictures, a sub-directory where user pictures should be stored, and a help box is available where you can put guidelines for photos, which will appear near the photo upload interface when the user is editing their profile.

This area is the "right" place to configure user pictures. In general, user profiles work a lot like content types, insofar as you can create and associate any number of fields with the site's user profiles. But if you want users to upload a picture that will be associated with them on the site, be sure to configure it here-- NOT by adding an image or file field to the user profile. Depending on how you use the image, the difference may be trivial (e.g. if you're only displaying the user picture in the context of Views you've created, it doesn't really matter where the picture is coming from), but Drupal core and other modules assume that user pictures will be uploaded using the interface generated by the settings you configure here. As a result, having the user picture automatically appearing along with the user's name on nodes they create only works if you use the user picture feature as intended.

10.3.3 System emails
The text of emails that Drupal sends to users can be managed through "Account settings". There are email templates configured for:

- New users created by an administrator
- User accounts that have been created but are awaiting approval
- User accounts that have been created, and no approval is required
- Accounts that have been activated (e.g. after approval)
- Accounts that have been blocked (e.g. for bad behavior)
- Confirmation of account cancellation
- Account cancellation (no confirmation required)
• Forgotten password
The default text for each of these is fairly sensible, but you may want to customize it. At the bottom of the screen, you can click on "Browse available tokens" to get a list of the bits of variable text (like the site name, or the user's name) that could be used in those emails. If you click in the editing window for the text of one of the emails, and click on a token from the "browse available tokens" pop-up, that token will be inserted into the editing window.

10.4 Creating user profiles
The user profile configuration is a set of sub-menus under Configuration > People > Account settings: Manage fields (/admin/config/people/accounts/fields) and Manage display (/admin/config/people/accounts/display). The configuration of these two sections works exactly the same way as the corresponding sections when editing content types. The only difference is that you have an option to check whether a field you've added should be displayed on the user registration form. Fields that aren't displayed on the user registration form aren't available when creating a new user, but become available when the user subsequently edits his profile.

10.4.1 Real Name
There are many places on a Drupal site where user names are displayed. For instance, if you chose to leave the "Show author and date information" box enabled for any content type, the user name will appear at the top of nodes belonging to that content type. In most cases, however, the default configuration is not an accurate representation of the user's real name, because it uses their login name (i.e. what they use along with their password to sign in).

The Real Name module can fix this by drawing on data stored in fields you've added to the user profile. After you've installed and enabled Real Name, ensure that you have added a field or fields to the user profile (by going to Configuration > People > Account settings > Manage fields) that can be used to store the person's real-life name. You might want to add a "Name" field, or you may want to add separate names for "Given name" and "Surname", for instance, if you ever want to be able to sort users alphabetically by surname.

To configure Real Name, go to Configuration > People > Real name (/admin/config/people/realname), and in the “Realname pattern” field, enter the tokens corresponding to the name fields, just like how you configured Automatic Nodetitles (see section 6.8.) If any users haven't filled in the fields you defined for the "Realname pattern", their login name will be used until they do.

10.5 Roles and permissions
Drupal has an extremely granular permission system. Almost every module comes with a set of permissions associated with it, in addition to numerous permission settings associated with Drupal core. There are even modules (like Content Access and Field Permissions) that provide even more granular permissions.

Drupal doesn't let you assign permissions to individual user accounts. Instead, permissions are granted to roles, and roles are given to users. The "administrator" role is automatically granted

---

99 https://www.drupal.org/project/rename
100 https://www.drupal.org/project/content_access
101 https://www.drupal.org/project/field_permissions
full permission to do everything on the site; every time you add a new module, the administrator role gains extra permissions. Drupal also automatically creates roles for "authenticated" and "anonymous" users. In general, you should be very careful about who you assign the "administrator" role to. Anyone with this role can thoroughly make a mess of the site, accidentally or deliberately. You also should probably put strict limits on what the anonymous user can do (e.g. anonymous users probably shouldn't be able to post any content), or else spammers will take advantage of it. It's best to think through what the real-world roles are for site development and maintenance (e.g. content creator, undergraduate data entry assistant, Drupal site builder) and create Drupal roles with permissions to match.

To create a role, go to People > Permissions > Roles (/admin/people/permissions/roles). Once you've created a role, go to the permissions page at People > Permissions (/admin/people/permissions) to check the boxes for the permissions that role should be granted. If you have a lot of roles, and it becomes awkward to read the titles of the permissions and see the role you want to edit, you can go to the Roles page and select "edit permissions" for that role.

If you have the Module Filter module installed (one of the modules on the recommended list in section 4.4), a search field at the top of the permissions screen will be enabled. If you type the name of a module, the permissions list will be filtered to only the permissions available for the module. You can also use syntax like perm:access or perm:view to show the corresponding kind of permission (access or view) for all modules where it is relevant.

10.5.1 Content permissions
By default, the Drupal permissions table has options for:
- Viewing published content (of all content types)
- Viewing one's own unpublished content
- Creating new nodes of a specific content type
- Editing one's own nodes of a specific content type
- Editing anyone's nodes of a specific content type
- Deleting one's own nodes of a specific content type
- Deleting anyone's nodes of a specific content type

While there are many options related to creation, editing and deletion, viewing is still all or nothing: either you can view all published content, or none of it. The Content Access module addresses this imbalance, and also allows you to add per-node permissions, so that individual nodes can have different settings from the defaults for their content type.

After you install and enable the module, a warning message will appear at the top of the modules screen that the content access permissions need to be rebuilt. Click the "rebuild permissions" link to do this. The same will happen if you disable the module.

Next, go to Structure > Content types > [the content type you want to edit] > Access control. This page will display the view, edit, and delete options for the content type, along with a list of the roles on your site. If you make changes to edit and/or delete permissions, these will be reflected on the overall site Permissions page, where you can also edit them. Specific viewing

102 https://www.drupal.org/project/content_access

115
permissions for content types can only be edited on the “Access control” screen for each content type.

If you check the box for per content node access control settings, next time you view or edit a node of that content type, you'll see an “access control” tab at the top, next to “view” and “edit” and “revisions”, if there have been revisions saved of the node. This tab has the same options as the access control page for the content type, and it has the same values as the content type access control setting by default. Changing the settings on an individual node’s access control page only affects that node.

10.5.2 Field permissions
The field permissions module lets you define the visibility and editability of individual fields within a content type, based on role. For example, you might want to have a “completion” field that only faculty collaborators can change, but all users can see. Or, if your site collects information about members of an organization using the Profile 2 module (as described in section 10.4.2), you might want to have a field that only the member and administrators can edit and view, such as for contact information.

Once you enable the field permissions module, a new option for “Field visibility and permissions” appears on the editing form for every content type, towards the bottom under the “field settings” section. By default, all fields are public, but you can also select “private” to make the field only viewable and editable by the author of the node. For more specific sets of permissions, select the “custom” option, and a table will appear that displays all the roles on the site, where you can grant for each of them any of the following options:
- Create own value for field
- Edit own value for field
- Edit anyone’s value for field
- View own value for field
- View anyone’s value for field

These field-level permissions don't override the viewing or editing settings set at the content type level. If a user doesn't have permission to view or edit the content type where the field appears, they won't be able to view or edit the field even if you grant them the field permission.

10.5.3 Permissions for unpublished nodes
The only default permission setting available for unpublished nodes is “View own unpublished content”. This may be problematic for sites where less-experienced assistants do a first pass at data entry, and more-experienced project staff review and publish those drafts. Only people with “administrator” permission (i.e. full permission for all configuration on the site) can view unpublished nodes.

The View Unpublished module\(^{103}\) provides options that allow users to view unpublished content of a particular content type. All these options are available on the Permissions page, grouped under a “View Unpublished” header.

10.5.4 Permissions for multi-author nodes
Many of the permissions related to nodes draws on a single-user concept of authorship. The way Drupal defines whether a node “belongs” to someone (e.g. for “delete own content”) is

\(^{103}\) https://www.drupal.org/project/view_unpublished
whether the user’s name appears in the “Authored by” field, under “Authoring information” at the bottom of the node creation and editing screen. There is no way to put more than one username in that field.

Even though Drupal does not technically support multi-author arrangements, many projects contain material that was created by more than one person. The Node Access User Reference module\(^{104}\) provides a way to give more than one user the rights reserved for the singular “author” of a particular node.

If you try to enable Node Access User Reference without first enabling User Reference (part of the References module package) or Entity reference (an alternative to References), a warning will appear and Node Access User Reference won’t be enabled. You must also click the “rebuild permissions” link before Node Access User Reference options will be available as part of the configuration for a user reference field.

Once you have done this, edit an existing user reference field in the content type where you want to allow multiple users to have “author” level permissions, or create a new user reference field. Under the help text box, a toggle-down menu for “Node access user reference” will appear.

The first set of checkboxes, “Grants for referenced users on the node”, will be the most useful. If you check “update” and/or “delete”, users referenced in this field will be able to update and/or delete the node, even if you haven’t configured general permissions so that anyone can edit or delete any node. You can set it so referenced users’ heightened permissions only apply after the node is published, or they can apply to published and unpublished nodes. The second set of checkboxes, “Grants for referenced users to create content”, is less useful. In most cases, you’ll want to use roles to define who can create what content types.

If you want permissions for referenced users to not apply for all nodes that use this user reference field, you can specify a View (see chapters 12 and 13) as well as an optional argument, so that the permissions only apply to the nodes that appear in that view.

Keep in mind that this module only addresses the permissions-related aspects of a multi-author situation. If you have enabled the display of author and date information for the content type, the user who actually created the node on the site will still be listed as the one and only author. To get multiple authors in a user reference field to display as multiple authors, hide author and date information for the content type. You may want to use CCK blocks (see section 9.7) or Display Suite and Views to make the user reference more closely resemble a typical author attribution.

10.5.5 Taxonomy access control

On sites that are maintained by a number of long-term groups of users, the Taxonomy Access Control module\(^{105}\) can be helpful. Instead of adding the names of every member of the group to all the group’s nodes using the Node Access User Reference module (section 10.5.4), you can create a role and taxonomy term for each group, and use Taxonomy Access Control to connect the two.

\(^{104}\) https://www.drupal.org/project/nodeaccess_userreference

\(^{105}\) https://www.drupal.org/project/taxonomy_access
Create a vocabulary for “group” (Structure > Taxonomy > Add vocabulary), and add a term for each group. Create a role for each group (People > Permissions > Roles). Install and enable the Taxonomy Access Control module, and go to Configuration > People > Taxonomy access control (admin/config/people/taxonomy_access); if it doesn’t appear as an option right after you’ve installed it, flush all caches by going to the house icon in the upper left of the administration menu, and choose “Flush all caches”.

On the taxonomy access control screen, for each role, click “edit access rules”. Under “new”, select the taxonomy term corresponding to the group. Then, under “update” (and, optionally, “delete”, if you want group members to be able to delete content), choose the option labeled “A”. Also, check the “add tag” box. This will allow all users with that role to edit and/or delete all nodes that are tagged with the corresponding group name. To complete this setup, be sure to add a term reference field that points to the “group” taxonomy to all the content types where you want to control access this way.

10.6 Spam prevention
If unauthenticated users are allowed to create anything at all on the site -- including submitting a request for a user account, creating a new user account, commenting, filling in a survey / webform -- you should install a module that can inhibit spam. Without it, even sites that aren’t widely publicized will battle spam.

10.6.1 Honeypot
For the greatest accessibility, a module like Honeypot is the best approach. Honeypot creates an invisible field on specified forms. No real human user can fill in the field, because it doesn’t visibly appear on the page, but it appears in the code. Because most spam bots are automated and access the form code directly, they will likely fill in the hidden field. Any submission that includes a value in the hidden field has revealed itself to be spam, and can be safely discarded.

Once you install and enable Honeypot, a message will appear asking you to configure it. You can click the link in that message, or go directly to Configuration > Content authoring > Honeypot configuration.

To activate Honeypot, either check the box for “Protect all forms with Honeypot”, or select individual forms towards the bottom of the page. If you directly control the creation of accounts on the site and are confident that no spam accounts can be created, you can limit Honeypot to only those forms available to unauthenticated users. Protecting all forms is easiest, as long as no form has a field with the same name as the “Honeypot element name”. By default this is “url”, but you can change this to another generic term (email, webpage, etc.) if necessary. If you’re interested in tracking spam attempts, you can also check “Log blocked form submissions”.

There’s also a time limit option. Any form submitted before the specified number of seconds passes after the page loads will be thrown out as spam. This can add another layer of defense against spam, but it interferes with page caching. If server load is a concern for your site, you may want to set this to 0.

106 https://www.drupal.org/project/honeypot
10.6.2 CAPTCHA

Another popular anti-spam tactic involves the use of CAPTCHAs, puzzles that a human user should be able to solve, but a spamming program should not. The most popular module for this is the CAPTCHA module\(^\text{107}\), which provides a framework that supports many different kinds of CAPTCHAs in addition to the ones that are included with the module itself. Add-on modules for CAPTCHA (including Google’s reCAPTCHA, which uses distorted book scans) provide alternative puzzles.

Install and enable CAPTCHA, as well as the Image CAPTCHA module included with it. Image CAPTCHA provides images with distorted letters and numbers that users have to type in. After you enable CAPTCHA a notice will appear at the top of the modules page with a link indicating you can now configure CAPTCHA. Click on the link, or go to Configuration > People > CAPTCHA (admin/config/people/captcha).

On this page, you can set a default CAPTCHA, choosing from the available options (by default, “Math” and “Image CAPTCHA”). The setting for each form defaults to “no challenge” (i.e. it is unprotected by CAPTCHA), but you can use the dropdown to choose the default challenge, or a specific challenge from among the available options. The CAPTCHA module uses the database name for each form, which is less intuitive than the Honeypot interface, but the names should be relatively transparent (e.g. “contact_site_form” is the sitewide contact form; “user_register_form” is the form where users can sign up for a new account; “user_pass” is the password reset form.)

The rest of the default settings are fairly reasonable, though you may want to change the prose in the description of the CAPTCHA or turn it off. You can also check the box for “Log wrong responses” to monitor the number of spam attempts on your site.

If you’d like more CAPTCHA choices, there’s a list on the CAPTCHA module page. The process for configuring each one differs slightly, and for some (like Google’s reCAPTCHA) you may need to register with the CAPTCHA provider and obtain an API key (a unique string of characters that grants you access to the CAPTCHA service). Be sure to read the module documentation for each one. When additional CAPTCHA modules are installed, they appear as options alongside “Math” and “Image CAPTCHA” on the CAPTCHA configuration screen.

11. Menus and blocks

11.1 Overview

Before moving on to creating new displays of the content that has been added to the example site, it is useful to better understand two more aspects of Drupal’s architecture: menus and blocks (bits of content or functionality that can be positioned somewhere on a page, like a list of recently published blog posts, a snippet of text crediting a funding agency, or a login interface.)

11.2 Blocks

Blocks are (generally small) containers of content or site functionality (such as a user login box, or a menu) that can appear in different places on the site (e.g. footer, right sidebar of blog posts, top right corner of the front page). Drupal core provides some

\(^{107}\) https://www.drupal.org/project/captcha
blocks (like a list of users currently logged into the site), some modules create blocks (e.g. Superfish\(^{108}\), which provides a block intended to be placed in region designed for a menu, and lets you display a menu in a drop-down way), you can create blocks using Views (see chapter 11), and you can create custom blocks with any arbitrary content. Block layout is on a theme-by-theme basis, because Drupal can't guarantee that the regions where you've put blocks will exist in the next theme you switch to, so if you change themes you'll need to reconfigure your block layout.

To manage blocks, go to Structure > Blocks or /admin/structure/block. What appears by default are the regions for your currently active theme (which appear in bold in the table), and all the blocks available to you, most of which will be under the header "Disabled" at the bottom. Other themes that are enabled but not set as default appear as tabbed options at the top of that interface.

11.2.1 “Demonstrate block regions” and positioning tricks

A link under the descriptive paragraph that says "Demonstrate block regions (Your Theme Name)" will show you a blank, generic page where the general location of each region will appear highlighted in yellow.

This doesn't always work well, particularly with groups of regions that form columns together-- for instance, a footer may have three different regions (left, center, right), but they'll only show up the way you expect if you first put a block in the left region, then the center region, then the right region. In the "demonstrate block regions" preview, they may appear stacked on top of each other. Putting a block in the right region without content in the center or left may make that block appear as if it were in the left region; an easy way around this, if you only want a block to appear on the right, is to create custom blocks with no content and use them to fill in the "slots" for the left and center regions.

In short, while "demonstrate block regions" can be a useful tool, the most reliable way to see what a block will look like in a region (and where exactly it will appear) is to assign it to that region, save, and take a look at the results.

11.2.2 Assigning blocks to regions

You can drag and drop blocks from "unassigned" into the region where you want them to appear. If you do this, be sure to hit "save" afterwards, before you go do any other configuration. If you drag and drop blocks to different regions, then hit the "configure" link next to one of the blocks, it will take you to the configuration page for that block without saving your changes, and you'll have to redo the drag and drop positioning.

\(^{108}\) https://www.drupal.org/project/superfish
11.2.3 Block configuration

The options that appear when you choose "configure" for a given block depend on the nature of the block. All blocks will have a "title" field and many will have a "description". The title will generally display in some stylized way above the block content; if you don't want a title, you have to enter `<none>`. Leaving the title blank will use whatever title the module or View that generates the block has indicated as the title. The description is what appears on the list of blocks on the block administration page ([Structure > Blocks](Structure > Blocks) or `/admin/structure/block`), to identify that particular block. For blocks created by Drupal core or many modules, you won't have the option of changing the description.

11.2.4 Region settings

The configuration page for a block will also list all enabled themes for your site, and in which region the block appears on each of them. You can use this interface to change the position of a block without having to go back to the drag and drop interface.

11.2.5 Visibility settings

The block configuration page provides multiple options for fine-grained visibility configurations. If you don't make any changes in this area, the block will always appear in the specified region, on every page. Making changes in the "visibility settings" area allows you to have blocks that only appear for users with certain roles, blocks that users can choose to turn off, blocks that appear on nodes of certain content types, and/or blocks that appear on pages with certain URLs.

These options are fairly self-explanatory, perhaps with the exception of the "Page" settings. To change these settings, you need to put down a series of paths, one per line. What exactly you put down will generally depend on what you've done for your Pathauto settings (see section 6.9). Maybe you want a block to appear on all user-related pages. In those cases, you would use whatever (fixed) text you put down in the pathauto configuration (e.g. "users"), then an asterisk, which matches any page (users/*). You can put things after the asterisk, too, if you want a block to appear only on the interface when a user edits their profile: user/*edit.

It's worth noting that you can't use tokens here the way you can in the pathauto configuration. You have to put down actual values, rather than placeholders (with the exception of the asterisk). If there's no fixed text that you use as part of the taxonomy path (e.g. if you use the default [term:vocabulary]/[term:name] for taxonomy terms), but you want a block to appear on the term pages for terms from all vocabularies, you'll have to list each vocabulary separately, by entering, for example, "tags/*", "another-vocabulary/*", "third-vocabulary/*" each on separate lines.
11.2.6 Custom blocks

The block administration page also has a link for "Add block", or you can go to Structure > Blocks > Add block (admin/structure/block/add). This is a way for you to create a block with any content--a copyright notice, an acknowledgement of a funding organization, a sponsor logo, etc. Don't forget to check that you have the right text format selected--depending on what you're using it for, you may need to make sure that the block body is set to "Full HTML". Programmers who have the "PHP filter" module enabled can even put PHP in a custom block. In general, though, the PHP filter module is not recommended if you're not fully comfortable with PHP and, ideally, the Drupal APIs.

11.3 Menus

Drupal allows you to create any number of menus (lists of links intended to be used for site navigation). Every menu is available as a block, and where the menu is displayed depends on where you put the block; how it's styled largely depends on your theme. Some modules provide additional menu display options, such as drop-down menus (i.e. menus where, when you hover over a menu item, any sub-items appear below or to the side. An "About" menu item on a center's website might have sub-items including "History", "Staff", etc.)

11.2.1 Default menus

A standard Drupal 7 installation provides three menus. You don't have to use any of them, but they might be a convenient starting point.

- Main menu: starts off with just a "home" link that will take you to whatever the front page of the site is (go to Configuration > Site information to specify what node, view, etc. should be the front page, if you want it to be something other than a blog-like display of the most recent nodes.) You can add other links to this menu and use it as the main navigation menu for your site.
- Navigation: this menu is often a grab-bag of different things. It starts off with menu items for adding content, and various modules add their own links here (e.g. the Biblio modules puts in menu items for the bibliography display page, and sub-items for "author", "keywords", etc.) Because it's not always clear what impact installing a new module will have on this menu, and unpredictability is not a desirable trait in a menu, it may be best to not use the Navigation menu.
- User menu: provides a "my account" menu item that takes an authenticated user to their profile, and a menu item for logging out. Potentially useful, though you might want to incorporate those links into your main menu.

11.3.2 Adding a menu

You may want to just add your own menu items to Drupal's default menus (particularly the "Main menu"), but if you need an additional menu, you can create it by going to Structure > Menus > Add menu (admin/structure/menu/add). Once you create it, it'll appear alongside the default menus on the menu configuration page (Structure > Menus or admin/structure/menu). You can delete any menus you've added, but you can't delete default menus.
Adding items to menus

There are two ways to add menu items: by editing the node or view-generated page that you want to provide access to through the menu, or by adding a link directly to the menu. This is one of the options for each menu on the menu configuration page, Structure > Menus (admin/structure/menu). If you want to add a taxonomy term, user, or something else that isn't a node or views-generated page to the menu, you have to use the "add link" option. "Add link" (which is also the interface if you edit an existing menu item) provides additional configuration options as well, such as "Show as expanded", which shows the links nested beneath the current link, even if the user is not on the page corresponding to the current link. You can also uncheck the "enabled" checkbox for a link to hide it in a menu, which can be useful if you're preparing pages for inclusion in a menu before you're done creating their content.

Editing a node

At the bottom of the node creation/editing form, there's an option labeled "Menu settings" with a checkbox, "Provide a menu link". If you check the box, a new set of options appear, including Menu link title-- the text that actually appears in the menu. This defaults to the title of the node, if the node has already been saved; otherwise, you have to enter the title manually.

Parent item is a drop-down set of options that shows all available menus and the menu items that already exist. If you want the node to be a top-level menu item, just choose the name of the menu (e.g. "<Main menu>"); if you want it to be a sub-item, choose the name of the parent item. By default, only "Main menu" is available, but you can change that by editing the content type that the node corresponds to (Structure > Content types > Content-type-name). Towards the bottom of the content type configuration page, there's an option for "menu settings" that lets you choose menus in addition to "Main menu" to make available when editing a node.

Once you've edited a node to make it a menu item, that menu item will show up when you "list links" for the menu (an option on Structure > Menus or admin/structure/menu). From there you can edit the menu item, e.g. by changing the link title.

Add link

From the menu configuration page (Structure > Menus), there's an "add link" option for each of the menus. Using that interface, you can put in the text that will display for the menu item, along with the path (could be a Drupal path like node/123, or you can use the more user-friendly pathauto alias like content/my-first-page; you can also put in external links here.) Like on the node editing page, you can select a parent item if you don't want the menu item you're creating to appear at the top level.

Arranging menu items

While careful use of the "weight" field (which allows items with a lower number to "float" to the top) can allow you to arrange your menu items in your preferred order, it's much easier to use the drag-and-drop interface. This interface also allows you to turn menu items into sub-items by moving them under a parent item. Click on "list links" from the menu configuration page (Structure > Menus), drag and drop items into the right order, then save. On this same screen, unchecking the "enable" box for any menu item makes it not display as part of the menu.
11.3.5 Making a menu available
Some themes include a configuration setting where you specify which menu will appear as the primary navigation. Other themes rely on blocks, and provide a specific region (see section 11.2.1) for the primary menu bar. Still other themes have no particular menu region, but are configured in such a way that one could reasonably add menu blocks to a sidebar or header region. You might also use a sidebar, header, or footer for additional menus, such as a menu for site editors that isn't visible to the general public. Some themes support the display of at least one level of menu sub-items (Bartik, the default theme, is one example), but in most cases, you'll need to use a module like Superfish or Nice Menus, which provide drop-down menus, to be able to see sub-items.

To make a menu visible, go to the Block configuration page (Structure > Blocks or admin/structure/block). If you want to use the default menu block, just look for a block with the same name as the menu (there's no label indicating specifically that it's a menu block), assign it to a region, and save.

While the first menu you think about may be the menu that appears towards the top of your site and provides the primary navigation, placing an additional menu in the sidebar can be useful particularly on complex sites with many nested sub-sections. Menus in sidebars -- or anywhere else where they aren’t stretched out horizontally -- display the menu hierarchy, and expand and contract sections as you navigate, so that you can easily see the “children” menu items of the page you’re currently viewing (i.e. those nested beneath the current page.) Links where you’ve checked “show as expanded” when adding or subsequently configuring the link will always appear expanded, even if you’re currently in another section of the site.

11.3.6 Dropdown menus
While some themes have a “Menu” region that displays at least one level of nested sub-menu items, in order to display complex menus, you generally need to install additional modules. For dropdown menus, Nice Menus[^109] and Superfish[^110] are the most widely used options. Both modules are widely used and comparable in functionality, but some themes are optimized to work with one or the other. For “mega menus” (complex, multi-column sub-menus that appear when you hover over a top-level menu item), the Superfish module can accommodate that arrangement, or you can use a dedicated mega menu module like TB Mega Menu[^111].

Superfish is the dropdown menu module recommended for the theme we will use for the example site, but Nice Menus works largely the same way. Download, install and enable the Superfish module. Download the Superfish library, linked from the module page and listed in the documentation. Unzip it, rename it “superfish”, and upload it to the sites/all/libraries folder (see section XX for more about adding libraries). If you want to add animation effects to your menu (e.g. having sub-menus “bounce” or “slide” open), you’ll also need to add the jQuery Easing library (also linked from the Superfish module page), as well as add the jQuery update module[^112], to update your Drupal site’s core version of the jQuery library to version 1.6.1 or higher. Download the jQuery Easing library, rename it jquery.easing.js, and put it in in sites/all/libraries/easing.

[^109]: https://www.drupal.org/project/nice_menus
[^110]: https://www.drupal.org/project/superfish
[^111]: https://www.drupal.org/project/tb_megamenu
[^112]: https://www.drupal.org/project/jquery_update
Once you’ve done this, go to the Modules page and enable Superfish, as well as jQuery Update if you want to use the animation effects. If you’ve enabled jQuery Update, first configure it by going to Configuration > Development > jQuery update (admin/config/development/jquery_update). Set the version to 1.8, if you’re using version 7.x-1.9 of the Superfish module.\footnote{For the Superfish module to work without needing any workarounds, either use a version of the module higher than 7.x-1.9 (or the development version), or upgrade jQuery to a version less than 1.9. Superfish 7.x-1.9 breaks with jQuery 1.9 or higher.}

Next, go to the Blocks page (Structure > Blocks). You should see four disabled blocks, “Superfish 1 (Superfish)” through “Superfish 4 (Superfish)”. Click the “configure” link for one of them.

On the configuration screen, provide a more informative block description than “Superfish [number]”, such as “Main menu (dropdown)”. It’s important to choose something other than just the menu name, because there’s already a block for each menu where the block description is the menu name (the default display of the menu, as described in section 11.3.5).

Under “Menu parent”, select the menu you want to display using Superfish. The default menu depth of -1 is generally the right choice, as is leaving “Take “Expanded” option into effect” unchecked.

Under “Superfish settings”, there are a number of configuration options:

- Menu type: horizontal (where sub-items open as a vertical drop-down column), vertical (where sub-items open as a vertical column to one side), or navbar (horizontal, but where sub-items open as a parallel horizontal row)
- Style: Superfish comes with a number of different styles, which provide different colors and spacing. Most are not very attractive, and odds are they will clash with your site’s theme. If your theme is designed to work with Superfish, the default “none” style is probably the best choice, since your site theme should take over with the styling. If you want to change the design of the Superfish menu, follow the “How to style” instructions in the README.txt file in the Superfish module folder, or in the Superfish documentation.\footnote{https://www.drupal.org/node/1125896}
- Animation speed: how long it takes for the animation transitions to take place; you can choose “slow”, “normal”, “fast” (the default), or enter a time in milliseconds.
- Mouse delay: how long, in milliseconds, the mouse can be outside a menu before the menu closes.
- Path class and path levels: leave these with their default values.
- Slide in effect: where the sub-menu items should come from. “Vertical” means they slide in from the top, “Horizontal” means they slide in from the left, “Diagonal” means they come in from the top left. If you added the jQuery Easing library, there’ll also be a long list of animations that begin “easeIn”. See the Drupal for Humanists website for demos of these animations.

Under “Superfish plugins”, you can choose to enable options to make the menus compatible with touchscreens and small screens (or small browser windows). After you’ve enabled these, the default configuration for them is generally fine.
Under “Multi-column sub-menus”, you can turn on Superfish’s mega menu functionality, and specify at what level of depth the multi-column sub-menus should begin, and how many levels should be included in the multi-column menu.

The Advanced HTML and Advanced CSS settings are probably only helpful if you’re doing custom theming for the menu.

12. Views

12.1 Overview

12.2 A very simple view using “add new view”
   12.2.1 Choosing what to display
   12.2.2 Displays
   12.2.3 Next steps for new views
   12.3 Displays
   12.3.1 Titles, and overriding and reverting display settings
   12.4 Format
      12.4.1 Grouping fields
      12.4.2 Show
   12.5 Fields
      12.5.1 Configuring date fields
      12.5.3 Configuring image fields
      12.5.4 Rearranging fields
      12.5.5 Style settings
      12.5.6 Inline fields and rewriting fields
         12.5.6.1 Using “Format” to display inline fields
         12.5.6.2 Excluding and rewriting fields
   12.6 Filter criteria
      12.6.1 And/or options
   12.7 Sort criteria
   12.3.7 Display-specific settings
      12.3.7.1 Page settings
      12.3.7.2 Feed settings
      12.3.7.3 Block settings
   12.3.8 Header & Footer
   12.3.9 Pager

12.1 Overview

The power of the Views module is one of the best reasons for developing well-structured content types for your site. Views allows you to select what data to display, from which content types, and how they should be displayed, filtered, and sorted. Views can do the same for data stored as part of user accounts, taxonomy terms, comments, files, and there may be many more options, depending on what other modules you have installed.

A full explication of all the settings and configuration options available in Views -- not to mention the many modules that extend it -- could easily fill an entire book. This chapter will provide a basic introduction to Views by walking through the case study of creating and modifying a
“People” page for the example site. This case study will familiarize you with the basic configuration options for Views, but the best way to develop proficiency with Views is through exploring and experimenting with it yourself.

Chapter 13 will build directly on chapter 12 by introducing the “advanced” Views configuration section, as well as creating different kinds of Views output, including tables, maps, timelines and slideshows.

12.2 A very simple view using “add new view”
The most basic Views configuration settings are available without even accessing the full Views interface. We will first use it to create a list of all the historical figures on the example site.

Go to Structure > Views > Add a new view (or admin/structure/views/add). Give the view a short but informative name, like “People list”. For views that will be a permanent part of your site, you should check the "Description" box and provide more information about what the view shows, or what function it plays on the site (e.g. "Displays all historical figures in the database.")

12.2.1 Choosing what to display
The next option allows you to choose what kind of data you want to display. The options in the dropdown menu include the following by default, though additional options may be available depending on what modules you have installed:

Content: this is the default option, and refers to content created using a content type form. The "of type" dropdown is set to All by default, which doesn't limit the content in the view to any particular content type. You might want to use this if you're creating a list of all recently updated content. The simple configuration screen only lets you choose "All" or a single content type; if you want to choose two or more content types (e.g. all new events and all new blog posts), you'll have to hit the "Continue & edit" button at the bottom of the screen and use the full Views interface to set this up.

In this case, we need to limit this to just the “Person” content type, so choose that from the drop-down list.

Users: this refers to user accounts on the site. If you want to display information that's stored in a user's profile, this is the right option.

Files: this refers to files uploaded to the site, or files from a third-party hosting provider embedded using the Media module. While you can also access uploaded files if you go to Content > File (if you have the Media module installed), creating a view gives you a lot more flexibility in what information is displayed about each file, as well as how many files are displayed per page.

Taxonomy terms: you can use this option to show taxonomy terms; you can also limit the list to one particular vocabulary. If you want to choose multiple vocabularies, you'll need to hit the "Continue & edit" button and use the full Views interface.

Comments: this option shows data stored as comments.
12.2.2 Displays
By default, the "Add new view" interface has enabled the option for creating a page to display the view contents. You might not want a page; some views, like ones that create a slideshow of content, only make sense as a block, and you'd never want them displayed independently on their own page. In such a case, you would uncheck the "Create a page" checkbox.

We do want our example View to be available as a page. Configure the following settings:

- **Page title**: this text appears in the same place, color, style, etc. as node titles on a node page. For the example view, use the page title People.
- **Path**: this is the URL (the part after your main site URL) where the page will appear. For the example view, use the path people.
- **Display format**: these options vary depending on what kind of data you're displaying. The options provided by the Views module are listed below; additional modules may provide other options. For the example view, use "Unformatted list".
  - Unformatted list - the default; it displays the content in a list, without bullet points or other adornment, and a blank space between each item
  - Grid - each item appears in an invisible box, and there are multiple rows and columns of these boxes. In the full Views interface (though not the "Add new view" page), you can specify how many columns the grid uses.
  - HTML list - an HTML bulleted list (using the <ul> element).
  - Jump menu - a jump menu is a dropdown menu where each of the dropdown items is a link; this is rarely used.
  - Table - a table of fields from the data type you specified under "show"; each field specified (using the full views interface) is a column, and each item is a row. If you choose this option, Drupal will automatically choose "fields" and you'll have to use the full views interface to configure it.
- **Display format options**: depending on the data type you've chosen, each of the display formats can be paired with one or more of the following options (using the dropdown next to the heading "of"). For the example view, choose “Titles (linked)”.
  - Comment - all the content stored in a comment
  - Fields - uses fields that you set up as part of content types
  - Full posts - the entire content of the node, including the values of all fields
  - Teasers - a shortened form of the node content; the exact nature of what the teaser contains can be configured by going to Structure > Content types > [select the content type] > Manage display > Teaser.
  - Titles - the Drupal title.
  - Titles (linked) - the Drupal title, linked to the node or comment
  - User - all the content in a user's profile
- **Items to display**: you can put a limit on the number of items that are displayed; the checkbox "use a pager" provides a row of numbers at the bottom of the view, each representing another page of results, along with options for next/previous and first/last. While it may seem counterintuitive, to display all items, enter the number 0. For the example view, enter 0 in order to display all people. Uncheck the "Use a pager" box; using a pager only makes sense if you are displaying fewer than all the results.
- **Create a menu link**: If you want the page generated by a view to appear in a menu, you can check this box, specify the menu, and what the title of the menu item will be. You can also add a view to a menu using the menu configuration interface (see section 11.2.3). For the example view, check the box for “Create a menu link”, put People in the link text field, and the menu should be the default "Main menu".
- **Include an RSS feed**: this will add an RSS icon to the bottom of the page, and will create an RSS feed corresponding to the content displayed in the view. For the example view,
check the box for "Include an RSS feed", set the path as people.xml, and the feed row style should be set to the default “Content”.

Check the box for “Create a block” as well. The title, like that of the Page display, should be “People”, and also like the page, it should display an unformatted list of linked titles. For items per page, enter 1, and leave “use a pager” unchecked.

12.2.3 Next steps for new views
The configuration options on the “Add new view” page may be sufficient, and you can always come back and edit the view you've just created using the full Views interface. There’s no way to return to the simple view-creation interface after you've saved a newly-created view. If there's nothing more you need to change now, you can hit the "Save & exit" button, which will save the view. The "Continue & edit" button takes you to the full views interface, where you have many more configuration options. Keep in mind, though, that your view is not saved when you hit "Continue & edit", and you have to be sure to save it on the next page.

For the example view, choose “Save & exit”, and use the new menu item to take a look at the new People page.

This view as we have created it has numerous shortcomings. The sort order may seem somewhat random. In fact, the default sort order displays the most recently entered item first. For this People view, the order in which Person nodes was created is an unimportant artifact of the data entry process, and we do not want to use it to determine the sort order. At the same time, the block display (which shows a single person and will appear on the front page of the site) shouldn't always display the alphabetically-first person in the database. We need to change the block -- but only for the block -- to display using an actually random order.

The display of the page could be more inviting. Instead of having a vertical text list, let’s change the view to use the “Grid” display format, in order to take advantage of the horizontal space on the page. Let’s also add an image thumbnail to each listing, as well as the birth and death dates of the person.

To make these changes, we need to edit the view we just created. If you're looking at the page that displays the view, the fastest way to edit the view is by hovering your mouse towards the upper right corner of the view, to the far right of the name of the view (here, “People”). A small cog with an arrow will appear, and a dotted line will appear around the view itself. Click on the cog, and select “Edit view”.

Alternately, you can go to Structure > Views > People (if it doesn't appear in the admin menu, clear the cache first), or go to Structure > Views and click on the “Edit” button next to the listing for “People”. This will take you to the full Views configuration interface.

12.3 Displays
Grouped at the top of the views configuration interface, in a gray box, are the different “displays” for your view. Displays are manifestations of the view (e.g. as a page, as a block, as an RSS feed), and a view can have one or more displays. Displays answer the question “In what form will this view appear on the site?” -- not in terms of what the view will look like, but the manner in which it appears.
The example view has three displays, as configured on the simple view-creation interface: Page, Feed, and Block.

By default, each display is given the name of the display type (e.g. Page, Feed, etc.) A single view may have multiple page or block manifestations, in which case it may be helpful to rename one or more of them to indicate what’s unique about each one. Immediately below the gray bar of displays, there is a section called “[Display type] details”. On the left side of the screen, you can click on the name of the display next to “Display name” to change the name of the display. This only appears within the Views interface, as well as other places that display the administrative name of a display (e.g. the blocks overview page), and won’t impact what’s shown to users. You might have displays with names like “Block-5” for a block that lists 5 people, or “Gallery” for a page that provides a gallery-like display.

On the right side of the screen within the “[Display type] details” section, there is a set of options for modifying the display you currently have selected. Some displays have a “View [display type]” option, which shows you how the output of the display will look on your site; this works better for some display types (e.g. Page) than others (e.g. Feed). If you hit the small arrow, you’ll see options like “Clone”, “Delete” and “Disable”. Clone is useful if you want to provide a variant of the display, while keeping most of the configuration the same (for example, if you have a block that displays an MLA-formatted bibliographic citation for works with two authors, and you want to also create one that displays the citation for works with three authors.) Disable is a way to “turn off” a display you think you won’t need, before fully committing to deleting it. Deleting a display removes it entirely, but only after you save the view. Before you save the view, the display appears with a line through it, and you can click on it to undo the deletion.

12.3.1 Titles, and overriding and reverting display settings

When you configure the various settings within Views, you can choose whether that configuration should apply to all displays, or whether it should only apply to one particular display, thereby overriding the default setting.

For the example view, the need to have different settings for different displays first occurs with the view title.

This title appears wherever a title would be appropriate for the display in question-- e.g. as the page title, as the block title, etc. By default, the title is the same as the name you created for the view.

Different displays can have different titles; in fact, this extends to all views settings. The ability to create different variants of the view in different displays is very useful, but it’s easy to make mistakes that can impact multiple displays and require a good deal of cleanup work. By default, when you make a change to a setting in Views (other than in the display-specific settings area; see section 12.x.x), it impacts all the displays where relevant (for instance, adding fields to a Page display won’t impact an RSS feed display that doesn’t use fields.) If you want to make a change that only impacts a single display, you need to use the drop-down option at the top of the configuration screen to change the setting from “All displays (except overridden)” to “This [display type] (override)”. This will change the “Apply (all displays)” button to “Apply (this display)”. If you make a change to a display and indicate it should apply only to that display, all other displays will retain the previous value.
Overriding happens at the section level: a display can have its title overridden, but its format, fields, filter criteria, etc. can remain unified with the other displays. The settings within a section that has been overridden, and/or the title of the section, will appear in italics. Overriding represents a fork in the road for the configuration of a given section. Once you've overridden a setting within a section for a particular display, that display is no longer impacted by changes that you make to displays that aren’t overridden within that section. For that reason, it’s best to do all configuration that should be shared by all displays before you start overriding displays to address unique configuration needs.

If you have overridden a display in a particular section and later realize you need to make a change across all your displays, there are two ways to approach the situation. You can either make the change separately for the overridden displays as well as the non-overridden ones (if the overridden displays have extensive changes), or you can revert the overridden displays to the default settings, make the changes across all displays, and then re-override the displays as needed. To revert an overridden display, click on any setting within the overridden section, and in the dropdown at the top, choose "Revert to default". This will change the “Apply (this display)” button into a “Revert to default” button. Click to restore the settings to those shared by other displays.

For the example view, go to the Feed display, and override the default title. The default title “People” makes sense for the page, but if you leave the Feed title as “People”, that won’t be very informative when a user adds the feed to an RSS feed reader. Change the title for the Feed display only to “CHAAMP: People”.

12.4 Format

This section refers to how the content is displayed. Clicking the name of a particular format ("unformatted list") allows you to choose a different option (such as "table"). Clicking on "settings" allows you to change the configuration for the currently-selected option (e.g. how to group the items in the unformatted list, which columns in a table should be sortable, etc.) The settings under “Show” refer to the format of the content that will be displayed in the table, unformatted list, etc.

For the example view, make sure you have selected the Page display, then click on “Unformatted list”. Change the display to “Grid”. The grid format is useful for a gallery-like display, filling up horizontal as well as vertical space. Making this change will impact both the page and the block displays, though there will be no visible difference in the block display, since it is only displaying a single Person node.

If you click on the “settings” next to “grid” once you have selected that display format, you can define the number of columns in the grid, and whether you want to use horizontal or vertical alignment. Horizontal alignment (where the first four results appear in the first row of the grid) generally leads to a more intuitive interface. The default number of columns, four, is fine in this case.

While we will not need this feature for our example view, the “grouping field” option available under “settings” here is extremely useful in many cases, and is worth discussing briefly.
12.4.1 Grouping fields
Under "settings" for any of the default Views formats (grid, HTML list, jump menu, table, unformatted list), you can choose one of the fields you've added (see section 12.3.5 or more about fields) as a "grouping field". This allows you to organize the nodes based on a particular parameter. For instance, if you added a "Profession" field to this view, you could use it here as the grouping field to create a display of people grouped by profession. Once you've added a grouping field and saved the setting, you can click "settings" again to add a second grouping field, which further groups the results within the first grouping using a different parameter. If you set "Profession" to be the first grouping field, and "Specialization" to be the second, Views would display all the physicians together (separate from the nurses and medical researchers), then, where applicable, it would group together all cardiology specialists, all neurology specialists, etc.

Any field can be used as a grouping field, but it's best to choose a field where multiple nodes will share a common value -- birth date fields, for instance, are not usually a good choice.

Fields that you wish to use as grouping fields are good candidates for modifying the field HTML (see section XX), by setting them to be a header (like <h2> or <h3>), and for excluding from display (see section YY). If you don't modify the field HTML, they will appear in a header-like way, but without the formatting that provides an additional visual clue to their function. If you don't exclude them from display, they will be repeated: once as the header, and once in every single record grouped under the header. Excluding them from display removes them from each record under the header.

12.4.2 Show
The "show" setting refers to the format of the content itself. For a view of nodes, you can choose between "fields" (where you can specify in the "Fields" configuration section which fields you want to display) and "content". If your view is based on users or files instead of nodes, there will be an option that corresponds to that entity type available in lieu of "content".

In most cases, you will use "fields", as our example view is currently configured. Choosing "content" or the equivalent is good in situations where you want to create a blog-like view, either of full posts or of teasers with links to read the full content.

12.5 Fields
In the "Fields" configuration section, our example view currently has one field, "Content: Title". This will display the Drupal title of the nodes that appear on our view. Because of the way we configured Drupal's title field for the Person content type, it displays the full name of all the people in the database.

We want this view to display an image of each person, along with their birth and death dates. To pull that information into the view, click the "Add" button to the right of the "Fields" label. This will open an overlay where you can check the boxes corresponding to each field you want to add. In most cases, you'll be adding fields from the "Content" section, so the "Filter" option isn't particularly useful. The search box, which automatically updates the list as you type, is much more helpful here. If you don't want to scroll through the whole list, type "birth" into the search field, and check the box for the "Content: Birth" field, then hit the "Apply (all displays)" button at the bottom of the screen.
12.5.1 Configuring date fields
Most of the parameters for configuring fields are the same regardless of the field type, but there are some important unique configuration settings for some specific fields.

We do not want a label for this field (i.e. we want to just display the date, not “Birth: November 22, 1913”), so uncheck the “create a label” box.

Under “Formatter”, the default “Date and time” is generally the right choice, though you can also choose “Time ago” or “Plain” (which stores the data as it is stored in the database, including “00:00:00” for the hour, minute, and second information that we are not capturing with the field.) Under “Choose how users view dates and times”, you can choose from the date formats that you previously configured in section 9.2.2.1. For this view, we just want to display the year, to prevent the birth and date displays for a person from getting too long. This means revising the date format configuration.

Click the “Apply (all displays)” button. The birth date of each person, using the default date format, will now appear under the person’s name. Save your view by clicking the “Save” button towards the upper right of the screen.

Now, go to Configuration > Regional and Language > Date and time > Formats > Add format and enter: Y which is the PHP date shorthand corresponding to “1872”. Then, go to the Date and Time settings (Configuration > Regional and Language > Date and time) and use the dropdown to change the “Short” date type to use that format. (This repeats the steps that you took in section 9.2.2.1 to configure the “Long” date format.)

Go back to your view by going to Structure > Views > People. Click on the “Content: Birth” field and select “Short” under “Choose how users view dates and times”. Click “Apply (all displays)”, and the view should update to just show the year of people’s birth.

Repeat this process for the “Death” field, adding “Content: Death”, removing the label, and setting it to only display the year. At this point, your view should show the name of each person, their birth year, and their death year, in a list.

12.5.3 Configuring image fields
Add the “Content: Image” field to your view. If you use the search box to find it, you’ll get numerous results in addition to the one you want. This is due to the fact that you have a content type on your site called “Image”, and the search box also searches the metadata of where each field appears, which is displayed under the field name.

This field does not need a label, either. The formatter should be the default “Image”, and the image style should be the “Medium square” (300 pixels, scaled and cropped) that you configured in section 9.2.5.1. Under “Link image to”, choose “Content”, so that when people click on the image, it will take them to the corresponding person’s profile. Hit the “Apply (all displays)” button to apply the changes you’ve made.

12.5.4 Rearranging fields
Now your view should display the name, birth date, death date, and image for each person, arranged in that order. Let’s rearrange the fields next. Click the little down arrow next to the
“Add” button for fields, and you can select “Rearrange”. Using the little + sign to the left of each field name as a handle, drag and drop the fields in the following order: 1) Content: Image, 2) Content: Title, 3) Content: Birth, 4) Content: Death.

If you want to remove fields, going into the “Rearrange” screen is the fastest way to do it, as you can simply click the “Remove” link, rather than editing each field individually and removing it from there.

12.5.5 Style settings
To clearly differentiate the names from the other text in this view, edit the “Content: Title” field by clicking on it. Toggle down the “Style settings” section. Check the box for “Customize field HTML”, and set the HTML element to H3 using the dropdown. Click “Apply (all displays)” and take a look at the view preview. It’s not a perfect representation of how your view will display once you save it, as it doesn’t use your site’s theme, but it gives you some sense for the effect you’ve created.

There are separate style settings for the field HTML and the label HTML in the “Style settings” section. If you ever have a view where you include a label as part of a grid, unformatted list, or HTML list display, by default the label will appear above (rather than next to) the field content. The easiest way to put the label and field content on the same line is to customize the HTML for the field and chose <span>.

12.5.6 Inline fields and rewriting fields
We don't want the birth and death fields to display on separate lines. Instead, we want them to display as follows: (1872 - 1935). There are two ways to accomplish this.

12.5.6.1 Using “Format” to display inline fields
To simply put fields on the same line and separate them using a hyphen, click on “settings” next to “fields” in the “Format” section of the Views configuration screen. Under “Inline fields”, check the boxes of the fields that should appear on the same line (Content: Birth and Content: Death), and put the hyphen in the separator field.

This almost has the result that we want, other than the parentheses. We can add the parentheses to the birth and death fields, respectively, by rewriting them.

Click on the “Content: Birth” field and toggle down the “Rewrite results” section. Check the box for “Rewrite the output of this field”. A text box will appear. Below it, toggle down the “Replacement patterns” section. The replacement patterns will show the syntax you can use to pull in the value for the current field, as well as any field that appears before it in the fields list. Sometimes there are multiple options for a single field, as in this case: [field_birth-value] will give you the “raw” value (the date as stored in the database, as if you’d chosen the “plain” formatter when you first set up the field), and [field_birth] will give you the value of the field, respecting the configuration options you’ve already selected (e.g. using the “date and time” formatter, displaying it as only a year, etc.) Type into the text box: ([field_birth] (i.e. an opening parenthesis and the replacement pattern for the value of the field. Then click “Apply (all displays)”. Configure the “Content: Death” field the same way, putting a closing parenthesis at the end.
12.5.6.2 Excluding and rewriting fields

An alternate approach is useful when you want different groups of inline fields to be separated in a different way (e.g. for birth and death dates to appear together separated by a hyphen, but for first and middle names to appear together separated by a space), or when you want to have separating punctuation appear immediately next to a field value, rather than with a separating space\textsuperscript{115}. It's also faster to configure than the approach described above.

Edit the “Content: Birth” field and check the “Exclude this field from display” box. It will disappear from your view as soon as you click “Apply (all displays). Next, edit the “Content: Death” field and toggle down the “Rewrite results” section, as well as the “Replacement patterns”. Even though “Content: Birth” isn't being displayed by your view, it's still available as a replacement pattern. In the text field, type: \texttt{([field\_birth] - [field\_death])} and click “Apply (all displays)”. The result will look the same as the result using the approach described above.

The Drupal for Humanists site includes an example view from the World Shakespeare Bibliography that extensively uses field rewriting. If you have a lot of fields that are excluded from display, and particularly if you have duplicates of individual fields that are rewritten or configured in different ways, you may want to toggle down the “More” setting at the bottom of the configuration screen for those fields, and give them a more descriptive name. For instance, if you had two birth fields, one rewritten using a parenthesis and displayed, and another excluded from display, you might name the latter “Birth - hidden, w/o parenthesis, for rewriting”.

12.6 Filter criteria

The filter criteria restrict what content is displayed. By default, even if you don’t limit your view to a particular content type, Drupal adds a filter to only display published content, “Content: Published (Yes)”. This is important because, by default, only site administrators and the person who drafted the unpublished content would be able to see it, and everyone else would see an error message if they tried to access the unpublished content.

On our example view there is an additional filter, “Content: Type (= Person)”. If you click on it, you could check the box for showing other content types as well, in contrast to the simple view creation interface which only allowed you to select one content type.

While we already configured the filters we need for the example view when we first created it, you could potentially add any number of other filters. A few examples:

- If you had a site with an “Event” content type, you could add a filter that limits the results to those where the event date is greater or equal to the current date, in order to only show events that have not yet taken place.
- If you wanted to create a gallery, you could add a filter to only show those nodes where a particular image field is not empty (using the File ID option, “Content: Image (field\_image:fid)”), in order to avoid gaps in the display.
- If you’re using the Flag module, you could add a filter in order to only show nodes that at least five people selected as a favorite.

The Drupal for Humanists website has many examples of how to use different filter criteria.

\textsuperscript{115} See the Drupal for Humanists website for a case study on using Views to create MLA bibliography listings for an example where this could be relevant
Filter criteria can be exposed, or made visible to users, allowing them to dynamically update the results list based on criteria they select. Chapter 13 will describe how to configure exposed filters.

If you need to create a one-off display of all the events in the life of a particular person (as a way to quickly gather material for a blog post about that person, for instance), you can set up a filter where you limit the results to event nodes that reference the person in question. In general, though, if you’re trying to create something like a block that you can use on an individual person’s profile, showing all the events in their life, filter criteria are not the right approach. Under the “Advanced” views settings, there is a section called “Contextual filters” that you can use to accomplish such a thing. Contextual filters are discussed in chapter 13.

12.6.1 And/or options
By default, all filters that you add to a view are connected by “and”: an item must meet all the criteria listed, or else it will be excluded. If you click on the small down arrow next to the “Add” button for filters, you can rearrange filters and create new groups of filters that can be separated by either “and” or “or”. For an example of how to use this, see chapter 13.

12.7 Sort criteria
By default, Views sorts nodes by the order they were published, with the most recent nodes appearing first, using “Content: Post date (desc)”. To define a different sort order, you'll need to add criteria, and rearrange them so they appear before the “Content: Post date” sort criterion, otherwise the nodes will sort first by post date, and then (if two nodes have the exact same post date) by the criteria you've specified. You could also remove “Content: Post date (desc)” if it isn't relevant for how you want to sort your content.

For the example view, we want different sort orders for different displays. Let’s start with the page display, and make its configuration the default. Remove “Content: Post date (desc)”. This will sort the nodes by their node ID, which is to say, by post date in the opposite order, with the nodes that were added first appearing first. Add “Content: Surname” as a sort criterion, and choose “ascending” as the direction. This will sort the people in alphabetical order, by last name, which is the correct order for the page display.

Next, click on the “Block” display at the top of the Views configuration screen. Once you have selected that display, click on the “Content: Surname (asc)” sort criterion. At the top of the overlay screen that appears, using the dropdown, choose “This block (override)” and then click the “Remove” button. The “Sort criteria” header should now appear in italics, without any criteria under it. If you click the “Page” display, the “Content: Surname (asc)” sort option should still be there. If you go to the “Page” display and “Content: Surname (asc)” is missing, add it to the page again, go back to the block, and make sure you’ve chosen “This block (override)” when removing the surname criterion.

For the “Block” display, add the sort criterion “Global: Random” and click “Apply (this display)”, then “Apply (this display)” again, as there are no configuration options for this sort criterion beyond exposing it to users, which wouldn't be useful since there's nothing they could configure either. A single person should appear in the view preview area. To see the effect of the random sort order, you can click the “Update preview” button above the preview area, again and again, and different people should appear below.
While it isn't relevant for the example view, the sort criteria-- like the filter criteria-- can be exposed for users to configure; in most cases, this amounts to selecting an ascending or descending sort for a given criterion.

12.3.7 Display-specific settings
At the top of the center column of the Views interface, there are display-specific settings (e.g. “Page settings”, “Block settings”, etc., depending on the display that you are looking at. Different displays have slightly sets of options, and we will walk through the options for each of the three displays of the example view.

12.3.7.1 Page settings

Path
The path is the URL where the Views-generated page will appear.

Menu
By default, Views-generated pages are not added to a menu. To add a page to a menu, click “No menu” and instead select “Normal menu entry”. Enter the title (the text that will appear in the menu), an optional description, and choose the menu from the drop-down. You can only select the top-level menu; if you want to make the Views-generated page a child page of another menu item, you need to first add the page to a menu, and rearrange it using the Menu configuration interface (Structure > Menus > [Menu name]).

Access
The default setting here limits access to the Views-generated page to any user who has the permission for “View published content”. By default, this includes anonymous (unauthenticated) users; see section 10.5 for more about how to define permissions. If this is a view of users rather than nodes, the permission setting will be “View user profiles”.

Instead of using a particular permission setting, you may want to limit access to a view to a particular role (like administrator). Click “Permission”, change it to “Role”, and then choose the role(s) that should be able to see the view.

Sometimes views need to be accessed by non-human agents, such as a Views Data Export CSV display that will be used by a Feeds importer on another site (see chapter 14 for more about Feeds). In such cases, you may need to change the Access setting to “None”, which bypasses the Drupal permissions system entirely.

12.3.7.2 Feed settings

Feeds displays have a “Path” and an “Access” setting, just like the Page display. There is an additional setting, “Attach to”, that applies to “Feeds”. This allows you to select other displays (such as a Page, and/or Block) that should display an RSS icon at the bottom, with a link to the RSS feed.

12.3.7.3 Block settings

Blocks have an “Access” setting, as well as a “Block name”. The block name is the text that will appear as the title of the block.
12.3.8 Header & Footer

The header and footer appear below the display-specific settings. You can add one or more text areas (“Global: Text area”; for writing custom text, like an explanation of what the view contains, or how to use exposed filters), or a View area (“Global: View area”, that will allow you to embed another View in the header/footer of the View), or a result summary (“Global: Result summary”, showing how many pages of results there are). You can also choose whether or not these items appear when a View has no results.

The example view doesn’t need a header or footer.

12.3.9 Pager

At the bottom of the center column of the Views interface, this section lets you specify the number of results that will be displayed, and how. By default, Drupal provides a "full pager" (that shows a number for each page of results, in addition to "next", "previous", "first" and "last" navigation options) and 10 results.

If you click on "Full" under "Use pager", you can choose to display a specified number of results, all results, or choose between the "full" and "mini" pager (which provides fewer navigation options). If you have enabled a pager (either full or mini), you can configure how many pages you want to offer, what the text of the pager should say (other than "first", "last", etc.), whether you want to provide an option for seeing all items, etc.

The example view displays all Person nodes, so a pager is not needed.

13. Advanced Views

This chapter is still in development as of March 2015. Please see the most recent version of the course pack for more information about advanced Views.

It will most likely cover:
- Advanced settings
  - Contextual filters
  - Relationships
  - No results behavior
  - Exposed form settings
- Creating and editing tables
- Slideshows, using Views Slideshow
- Exporting data, using Views Data Export and Views Datasource
- Map displays, using Leaflet, with Leaflet More Maps for additional options
- Timelines, using Simple Timeline, Timelinr, and/or Views TimelineJS.

---

116 https://www.drupal.org/project/views_slideshow
117 https://www.drupal.org/project/views_data_export
118 https://www.drupal.org/project/views_datasource
119 https://www.drupal.org/project/leaflet
120 https://www.drupal.org/project/leaflet_more_maps
121 https://www.drupal.org/project/simple_timeline
122 https://www.drupal.org/project/timelinr
• Editview\textsuperscript{124}, for easily editing nodes and adding new nodes
• Image galleries, using Colorbox\textsuperscript{125} and default Views configuration options
• Views Conditional\textsuperscript{126}, which allows you to use conditions within individual fields in a view

14. Importing data

14.1 Overview
If you already have data for your project in another format, you can import it into Drupal without manually re-entering everything using the Feeds module. Perhaps data entry was already underway in a spreadsheet before you started building a Drupal site, or maybe you’re migrating your project from different software, like FileMaker Pro. There is even a module that allows you to import data from XML, like TEI-encoded texts. This chapter will cover how to import data from a spreadsheet via a CSV (comma-separated values) file, using our example site. It will then address the use cases of pulling in items from RSS feeds, and importing data from a TEI file. It will conclude with a brief overview of other relevant modules that extend Feeds.

14.2 Essential modules for data import
The most widely used module for importing data is Feeds\textsuperscript{127}. Feeds allows you to specify the type of file you’re importing from, whether you want to upload it or point to a URL, how often you want to run the import, what kind of Drupal “thing” it should create (users, nodes, taxonomy terms, etc.), and how to map the source data into Drupal’s fields. To install Feeds, you must also install its prerequisite, the Job Scheduler module\textsuperscript{128}.

If you need to modify or clean up the source data before importing it into Drupal, you may be able to simply use the Feeds Tamper module\textsuperscript{129}. Feeds Tamper offers a wide range of transformations that can be applied to any source data that has been mapped to a Drupal field (see section 14.3.5 for field mapping, 14.3.6 for Feeds Tamper).

By default, Feeds supports CSV files and RSS feeds. If you want to import from XML or HTML files, you need to also install and enable Feeds Extensible Parsers\textsuperscript{130}.

A fairly comprehensive list of all plugin modules for Feeds is maintained on drupal.org\textsuperscript{131}.

If you install and enable any of these plugin modules for Feeds after you’ve already installed and enabled Feeds, be sure to clear the cache (by hovering over the house icon in the upper left of the administration menu, and choosing “clear all caches”), or the new options provided by the Feeds plugin module may not appear when you go to the Feeds interface.

\textsuperscript{123} https://www.drupal.org/project/views_timelinejs
\textsuperscript{124} https://github.com/agile-humanities/agile_editview
\textsuperscript{125} https://www.drupal.org/project/colorbox
\textsuperscript{126} https://www.drupal.org/project/views_conditional
\textsuperscript{127} https://www.drupal.org/project/feeds
\textsuperscript{128} https://www.drupal.org/project/job_scheduler
\textsuperscript{129} https://www.drupal.org/project/feeds_tamper
\textsuperscript{130} https://www.drupal.org/project/feeds_ex
\textsuperscript{131} https://www.drupal.org/node/856644
14.3 Overview of Feeds settings

Install the Feeds module, and enable Feeds and Feeds Admin UI, along with Feeds’ prerequisite, Job Scheduler. Go to Structure > Feeds importers > Add importer. Provide a name for the importer; this should usually include information about what kind of data the importer creates, and optionally, what kind of source data it draws from. For instance, “TEI to poem” for an importer that creates “Poem” nodes out of imported TEI files, or “User import”. You can include more details in the “Description” field.

The next page is the main configuration screen for Feeds, and is broken up into a number of sections.

14.3.1 Basic settings

This section allows you to change the name and description of the Feeds importer. For the “Attach to content type” setting, the default “Use standalone form” option is generally correct if you only have one source for your importer. For instance, if your importer is meant to pull in data from one particular CSV file even if you expect you may end up uploading multiple revisions of the CSV file -- the standalone form is the right choice. If, however, you want to import data from multiple different sources on an ongoing basis (like RSS feeds from different sites) using the same Feeds importer, to make this possible you should create a new content type (call it something like “RSS feed” if that’s your source data), and attach the importer to that content type. Don’t worry about adding any fields to the content type you create. You won’t store any data in it other than the name and the URL of the source.

If this is an import that should happen on an ongoing basis (e.g. importing from a file on a remote server that is frequently updated), you can specify how often the import should happen. If the importer you are creating is meant to be run once, just to get data on the site so that it can be managed there, you can set “Periodic import” to “off”.

You should choose between “Import on submission” and “Process in background”. For most imports, “Import on submission” is correct. That setting immediately carries out the import as soon as you hit “Submit” on the import page. If you choose “Process in background”, Drupal will import 50 records every time cron (a series of automated site actions, including indexing new content for the site search and importing data in the background) runs. By default, cron runs every 3 hours; you can modify this by going to Configuration > System > Cron (admin/config/system/cron). If you have an import of thousands and thousands of nodes, you may want to increase the number of records imported per cron run. There is no easy, UI-based way to configure it, but it can be done with a small tweak to the settings.php in your site’s code base, as discussed in this Feeds module issue.

14.3.2 Fetcher

The “Fetcher” is how Drupal acquires the source file. The default fetcher is the “HTTP Fetcher”, which allows you to put in a URL pointing to the file. If you click the “Change” link next to the “Fetcher” header, you can choose “File upload”, which allows you to upload a source file.

---

132 Note that you can also import data from an entire folder of source files, as mentioned in 14.3.2.2. This doesn’t change the applicability of the standalone form.

133 https://www.drupal.org/node/1551246
14.3.2.1 HTTP Fetcher
There are configuration options for attempting to detect an RSS feed if you submit an HTML page. This works, for instance, on most WordPress sites: if you were to put in the URL of a WordPress site to a Feeds importer where you've checked the “Auto detect feeds” box, Drupal should be able to identify the site’s RSS feed and import based on that.

By default, Drupal waits 30 seconds to get a response after sending a request to an external website. If you want it to wait longer (for instance, if you know that the server that hosts the source for the feed is slow to respond), you can indicate a large number of seconds in the “Request timeout” field.

14.3.2.2 File upload
You can add file extensions to the “Allowed file extensions” field, but the default set should cover most cases. The default upload directory is fine.

If you want to import a directory of files (for instance, if you have a directory of TEI-encoded poems uploaded to your site, and want to import content from each file into a “Poem” node), check the “Supply path to file or directory directly” box.

4.3.3 Parser
The “Parser” setting lets you specify the kind of source file you will be using.

14.3.3.1 Common syndication parser
This parser is used for importing RSS feeds, which are widely used to disseminate web-based content like blog posts, or notifications of updates to websites. There are no configuration settings.

14.3.3.2 CSV parser
This parser is used for importing CSV (comma-separated value) files. CSV files are used to store tabular (spreadsheet) data in a format that isn’t dependent on any piece of software -- unlike, for instance, XLS files from Microsoft Excel. You can export data from any spreadsheet software, and from individual tables in database software (like FileMaker Pro) as a CSV file. Be sure to choose “UTF-8” or “Unicode” encoding, if possible, to ensure that any non-Latin characters are stored correctly.

If you choose the CSV parser, there are a couple configuration options that you can almost always leave with the default settings. While this is quite rare, if your CSV file uses something other than a comma to separate the columns of the spreadsheet (such as a tab, or a semicolon), you can select it from the dropdown menu. If there is not a header row on your CSV that indicates what each column is (e.g. “Project name”, “Disciplinary field”, “Surname”, etc.), you can check a box for that, and use the number corresponding to each column instead of a label. If you do this, note that the first column is number “0”, the second column is “1”, etc. If your CSV file does not have headers when you export it from the source software, it may be easiest to open it in a plain-text editor (e.g. Notepad or TextWrangler, not Microsoft Word or any other editor that includes text formatting options) and add headers rather than dealing with the numbers.
14.3.3.3 HTML Xpath parser and XML Xpath parser
If you’ve installed the Feeds Extensible Parser module, this option will be available. The configuration for the parser allows you to define XPath expressions that will pull out the data that you want to map to Drupal fields. First, you must define the context. This is the expression relative to which all subsequent expressions will be evaluated. In XSLT, this is similar to the expression used in `<xsl:template match="some-expression-here">`. Then, you can add one new source at a time, give it a name (which will be used in the mapping interface), and an expression. After each new source, hit “save” at the bottom of the screen. You can choose to display errors and/or use the debugging mode if you’re testing the feed or if you experience problems.

14.3.4 Processor
The Feeds settings described so far deal with the data source. How often should it be imported? How can Drupal access it? What is its format? The Processor section, in contrast, deals directly with the connection between the source and what you want the source to become once it’s imported into Drupal. The default processor options are “Node”, “Taxonomy term” and “User”. Work is underway on the Feeds module to support a more generic processor that would make it possible to import content into the “File” entity created by the Media module; in the meantime, importing Files can be done as part of importing nodes, users or taxonomy terms (if there is a file field associated with one of those content types).

14.3.4.1 Node and taxonomy term processor
In the settings for the node processor, “Bundle” refers to the content type that should be created for each entry in the source file. For the taxonomy term processor, there is a more intuitively-labeled “Vocabulary” option. Note that you have to choose only one; if you want to import nodes from multiple different content types or terms from multiple vocabularies, you need to create multiple source files and Feeds importers.

You can specify what should happen if Drupal imports content that matches existing content. (In the “mapping” configuration section, you define how Drupal should find matches.) The options are fairly intuitive. If you choose “Update existing nodes/terms”, only the fields that match the mappings you have defined will be updated. If you’ve made other changes to the node/term (e.g. comments, or adding data to other fields that don’t match a mapping), those will remain untouched.

The text format should be “plain text” (the default), unless the source data itself includes HTML, in which case you should choose one of the text formats that accommodates HTML (“filtered text”, “full HTML”, or a text format you have defined; see section 8.2 for more on text formats).

The node author you set on this screen is the default author. On the mapping screen, you can indicate that some aspect of the source data indicates the author, and that will supersede the default setting. It’s best to assign imported nodes to an actual user (even if it’s the “admin” user). If you use the default “anonymous” value for the author, you should uncheck the default “authorize” box, because if you’ve configured permissions correctly (see section 10.5) anonymous users shouldn’t be able to create new nodes.

If you’re importing data that should be a permanent part of your site, you can use the default “Expire nodes: Never” setting. If you’re importing data that you want to display on your site for a short period of time (e.g. the latest RSS feed items from another site, which will quickly be
replaced by newer items), you can choose a shorter time period. For example, if you expect that new RSS feed items are published every five hours, you might want to expire nodes after 6 or 12 hours, so your site doesn’t get cluttered up with data that will never be displayed again.

### 14.3.4.2 User processor

The user processor has a similar setting for updating existing users. You can define which role the users should have; this is not configurable as a field mapping, so all imported users should have the same role, or you should add roles later.

Feeds does not send email notifications to users when their accounts are created using Feeds. One way to work around that is to import user accounts with the “Blocked” status. When you change their status from “Blocked” to “Active”, they will receive an email with their username, password and the login URL.

### 14.3.5 Mapping

Mapping is a setting under “Processor”, but it is nearly identical for nodes, taxonomy terms, and users, with the exception of the specific options available under “Target”. “Target” options are the Drupal fields (user-created, like “Given name” or “Surname” or system-internal, like “Created date”) into which the data will be imported. The mapping interface is where you indicate how different components of your source data should be mapped onto the different fields (targets) within Drupal.

The “Source” section is where you specify where Drupal should look in the source file for the data that will be imported into each field. Source options vary based on the parser you selected. If you are using the CSV parser, you will be presented with a text field where you can type in the label of the column in your source file corresponding to the data you want to import. You must type this in exactly as it exists in the source CSV file for it to match; it may be easiest to copy and paste it directly from the CSV. If you are using the HTML or XML parser, the source options will be a drop-down list drawn from the configuration you entered on the parser settings page (where you defined XPath expressions to find different pieces of data in your source file). If you are using the Common Syndication parser (for RSS feeds, for instance), the source options are drawn from the properties defined for RSS feeds (e.g. title, author name, publication date, etc.).

You can map a source to more than one target, but you can only create one Feeds Tamper configuration for each source. Mapping multiple sources to a single target depends on the nature of the target, but in most cases, Feeds Tamper will be required. See section 14.3.6 for how to address both of these scenarios.

#### 14.3.5.1 Unique targets

Some “Target” options can be “used as unique”. This means using that field as a way to identify unique nodes, in order to determine whether an existing node matches a node represented in the CSV. For instance, say your site has an existing node with the title “Crime and Punishment”, and your CSV has a row where the data in the “Title” column is “Crime and Punishment”. If you are using the node processor, enter “Title” in the source field, choose “Title” from the target dropdown, and indicate that it should be used as unique (by clicking the cog under “target configuration” and checking the “Use as unique” checkbox), when Drupal starts importing the “Crime and Punishment” row, it will recognize that a “Crime and Punishment” node already exists, and will proceed as you specified in section 14.3.4.1, by skipping, updating or replacing that node.
14.3.5.2 Term reference fields
When used as a target, term reference fields have target configuration options. The default setting searches taxonomy terms by term name (i.e. it matches a term in the source to a term stored in Drupal based on the term name, rather than a system-internal ID number), which is correct in almost all cases. If you are importing data that uses terms that don’t already exist on your Drupal site, and you don’t want to first run another feeds importer in order to pre-import the new terms, you can click the cog icon under “target configuration” for the term reference field in question, and check the box for “Auto create”\(^{134}\).

14.3.5.3 Node reference fields
There are numerous target options available for node reference fields, but the most common setting is “Node reference by node title”. If you are importing data into a node reference field that accepts multiple values, instead choose “Node reference by node title -- allow duplicate nodes”. For multi-valued node reference fields, you will need to add a Feeds Tamper plugin for it to work; see section 14.3.6.

14.3.5.4 Image and file targets
If you want to import images or files, the easiest thing to do is upload them directly to the sites/default/files directory (or some sub-directory within it), and then use the filename as part of your spreadsheet. You have to add public:// to the front of the actual filename (if you have uploaded the files to the public files directory), which you can do either in the spreadsheet itself or using Feeds Tamper’s “rewrite” plugin (see 14.3.6.4).

14.3.5.5 Date targets
Even if you did not configure it to have “start” and “end” values, every field created using the Date module has those two variants as part of the feed importer. Choose the “start” variant if your date field is configured to only take a single value. (Note: Partial date fields are handled a little differently; how Partial Date works with Feeds is evolving as of March 2015. Consult the latest version of the course pack for details.)

14.3.6 Feeds Tamper
After you have saved the mappings for your Feeds importer, you may need to use Feeds Tamper to manipulate your source data. You can access Feeds Tamper through the “Tamper” tab in the upper right of the Feeds configuration interface, or by using the administration menu: Structure > Feeds importers > [Feed importer name] > Tamper.

Feeds Tamper lists each source you defined in the Mapping interface, along with the target(s) it is mapped to. If a source is mapped to multiple targets, they all get grouped together. Under each source is a table with an option to “Add plugin”. Feeds Tamper comes with a set of plugins that can transform your data; there are many more besides the common ones discussed below. Be sure to explore the full list if you need to manipulate your data to see if there is an option that fits.

\(^{134}\) Due to a bug in the Feeds module (https://www.drupal.org/node/2026543), for this to work correctly and not create duplicate terms, you need to use Feeds Tamper and add the “Trim” plugin to the term reference field. Leave the field blank to trim whitespace, and choose “both”. Note that if your source file has more than one term to import into the term reference field, you should also add the “Explode” Feeds Tamper plugin, and run that before the “Trim” plugin.
14.3.6.1 Adding a plugin
To add a plugin to a source, click the “Add plugin” link. On the next screen, select the plugin from the dropdown list. If you are only going to use the plugin once for a given source, you can keep the default description and machine name. If you need to apply the same plugin more than once (for instance, you might use the “Find replace” plugin multiple times to modify the data in different ways), you will need to at least provide different machine names for each instance. The available configuration options vary from plugin to plugin.

14.3.6.2 Explode
If you are importing data from a CSV into a field that accepts multiple values, and your CSV spreadsheet has cells that contain more than one value (e.g. a person with an appointment in multiple departments might have a spreadsheet cell under “Department” that looks like “English, Interdisciplinary Studies”), you need to use the “Explode” plugin so that Drupal recognizes the values as distinct. The string separator is the character(s) used to delineate the different values in the source file. If you are importing data into a multi-valued taxonomy term field, node reference field, or user reference field, you should add the “Trim” plugin after the “Explode” plugin.

14.3.6.3 Trim
Importing into multi-valued fields that are supposed to match existing nodes, terms or users may malfunction if you don’t trim spaces before and after each value. Leave the “Characters to trim” field blank in order to trim spaces, and leave the “Side” setting on “Both”.

14.3.6.4 Rewrite
This plugin is useful if you want to take multiple pieces of source data, and map them to a single Drupal field-- for instance, if your source file has “given name” and “surname”, and you want to map them to the Drupal “title” field. If you also want to store each value separately in Drupal, you can simply map the “given name” source to a “Given name” field target, and the “surname” source to a “Surname” field target. To create the mapping to the title field, you can add a new, blank column to your CSV source, by putting something in the header row (for instance, “Title”, if that isn’t already used by another column.) Map the “title” source to the “Title” target, and add the “rewrite” plugin in Feeds Tamper. In the rewrite plugin configuration options, you can specify the “replacement patterns” (i.e. other imported source data) to use. Enter “[given name] [surname]” (or the equivalent, depending on your source data) and save.

If you want to import “given name” and “surname” into the title field, but you do not want to store the data separately, you can directly map one of the sources to the title (for instance, mapping the “given name” source to the “Title” target.) For “surname”, create a mapping to a “Temporary target”. This will temporarily store the data from the source, just long enough for you to use it in Feeds Tamper, before discarding it. On the mapping list, temporary targets will display as “Missing”, which is not a problem. In Feeds Tamper, go to the configuration for “given name”, and add the “rewrite” plugin. For the values, enter “[given name] [surname]” (or the equivalent) and save. This will put the values of the given name and surname in the title field, but they won’t be imported anywhere else.
14.3.6.5 Find replace / find replace REGEX
The “find replace” filter allows you to specify the text to search for in the source, along with the replacement text. There are checkboxes for making this case sensitive, respecting word boundaries, and matching whole words/phrases.

To do more complex find and replace queries, you can use the “find replace REGEX” plugin, which lets you use regular expressions, a particular kind of syntax for doing find and replace, which allows you select things like all dates within a text that are formatted a particular way, or all email addresses. Many programmers are familiar with at least the basics of regular expressions. If you need to do sophisticated find and replace work on your source data, it might be helpful to show your data to a programmer and explain what data you are trying to capture. Because the details of regular expression syntax vary by programming language, you may want to point the programmer to the documentation for the particular flavor of regular expressions used by Drupal¹³⁵.

14.3.6.6 Copy source value
This plugin can be used like “Rewrite”, insofar as it can copy data both to and from another field. Unlike “Rewrite”, which takes the value of the field after all its Tamper-based modifications have been done, you can add a “Copy source value” plugin at any point in the Feeds Tamper workflow, and it will copy the value at that point, before subsequent changes are made.

14.3.6.7 HTML entity decode
Depending on how your source file was created, you may notice that punctuation shows up strangely after you’ve imported the data. If you see things like “&amp;” and “&quot;” in your imported data, you should re-import it using the “HTML entity decode” Feeds Tamper plugin, which will restore the actual punctuation marks¹³⁶.

14.3.6.8 Convert case
If the text in your source uses, for instance, all capital letters, and you want to normalize that before the import, you can use the “convert case” plugin. This provides a choice of converting the text to all capital letters, all lower-case letters, “title case” (every word capitalized), or “sentence case” (the first word capitalized).

14.3.6.9 Strip tags
Particularly if you are using the HTML Xpath parser source, you may want to strip out the HTML tags embedded in your data, if you are importing the text into a plain-text field. (See section 8.2 for more on text formats.) You can specify which tags, if any, you should remain after running this plugin.

¹³⁶ Note: this problem may not always be caused by the source data. If you are using Automatic Nodetitles, using tokens provided by the Token module (i.e. tokens that use underscores in their names), you may see things like &amp; in the titles of nodes, even when the data in the fields that Automatic Nodetitles is pulling from has the correct punctuation. If that is happening, replace the tokens with the equivalent entity tokens (the variants with a hyphen in their name). See section 6.8.1 for more on entity tokens.
14.3.6.10 Keyword filter

If you import data from a spreadsheet that you have prepared specifically for your site, you can be confident that all the data in the source belongs on the site. The keyword filter plugin is useful if your source isn't specific to your import--for instance, if you are importing data from another site's RSS feed, but not all items they publish are relevant. You can define words and/or phrases that must occur in the source; if those words do not occur in the field where the feeds tamper plugin is assigned, the corresponding item won't be imported. You can also invert the filter to turn it into a blacklist, so that the terms in the list cause items to be rejected, rather than accepted.

If you want to require a source to have all of a certain set of words, include them all on a single line in the text box. If you want to require a source to have any of a certain set of words, include each on a separate line. See section 14.5.4 for an example.

14.3.7 Importing data

After you have gone through all these configuration settings, go to the “Import” page. This page is not part of the administration menu, and as such you have to type it into the URL bar yourself by going to [yoursite.university.edu]/import. (Note that Structure > Feeds importers > Import importer does not take you to the import page, but rather, to a different page where you can copy and paste the exported configuration settings for a feed importer.) Click on the name of the importer you want to run. Only importers that use the “use standalone form” configuration under “basic settings” appear on this list; for importers that are tied to a content type, go to Content > Add content > [content type in question] to do the import.

Once you’ve selected the importer to run, you will be presented with a field where you must specify the source, either by putting in a URL or by uploading a file, depending on your choice of fetcher in section 14.3.2. Once you’ve indicated the source, click the “Import” button to start the import.

14.3.8 Import page options

The import page for each Feeds importer has a set of tabs along the top that present additional options. If an import didn’t yield the results you wanted (for instance, if your Feed Tamper plugins didn’t work as expected), you might want to delete all the items you created in the import, in order to make subsequent imports go more quickly than if Drupal has to look for existing node matches in order to update or replace content. The “Delete items” tab allows you to do this. Note that it deletes all items that were ever imported using that particular Feeds importer, not just the items from the most recent time you ran the importer.

The “Log” tab shows you how many items were created each time the importer has been run, as well as how long the import took to run.

The “Unlock” tab can be useful if you are running a large import and it gets stuck, failing to make any more progress. In those cases, you can unlock the importer, delete the nodes that have been imported (or not, if you prefer), and restart the import.

14.4 Example: importing from a CSV

To illustrate importing data from a CSV, we will create a series of events on the example site, associated with Alexander Thomas Augusta, Mary Eliza Mahoney and Adah Belle Thoms.
Before getting started, be sure to create a Person node for each of these three people. You can leave everything blank except the given and surname fields, or you can use the data on the book website to fill in their biographical information.

14.4.1 Source data
Start by downloading the CSV from the website (we will provide this URL in class). If you want to take a look at it, open it in a plain text editor like TextWrangler on Mac, or Notepad on Windows. The header row includes the following fields:

- Event - a very brief description of the event
- Person - each event is associated with the life of one or more people.
- Date - the date of the event; the granularity varies, but each date includes at least a year
- Location - text describing the location where the event took place
- Description - a description of the event
- Event type - categorizes the type of event
- Institution - associates an institution with an event
- Main timeline - uses “yes” or “no” to indicate whether the event should appear on the main timeline
- Year - this column is empty other than the header; you’ll use it in combination with Feeds Tamper.

14.4.2 Creating the importer and basic configuration
Create a new feeds importer by going to Structure > Feeds importers > Add importer. For the name, call it “Event importer from DfH CSV”, with the description “Imports events using the CSV file from the DfH website.”

The default values under “Basic settings” are mostly fine, but change “Periodic import” to “Off” and save.

Set the fetcher to “File upload” and the parser to “CSV parser”. The default node processor setting is correct. In the settings for the node processor, choose “Event” under “Bundle”. Under “Update existing nodes”, select “Replace existing nodes”. For the author, choose your own account.

14.4.3 Mapping
Create the following mappings:

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Target configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Title</td>
<td>Used as unique</td>
</tr>
<tr>
<td>Person</td>
<td>Person (node reference by title) -- allow duplicate nodes</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>TBD, pending availability of Partial Date Feeds importer</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>Location</td>
<td></td>
</tr>
</tbody>
</table>
14.4.4 Feeds tamper

**Person**
For the Person field, add the “Explode” plugin. The default comma is the correct separator. Then, add the “Trim” plugin; the default settings for that are correct.

**Date**
TBD; we will discuss in class.

**Event type**
The Event type field only takes one value, so you don’t need to apply the “Explode” plugin.

**Institution**
For Institution, because you are allowing it to create new taxonomy entries, choose the “Trim” plugin to make sure that it doesn’t create duplicate new terms due to confusion over whitespace.

**Main timeline**
As discussed in section 7.2.2.2, the “Main timeline” field is a Boolean field that uses the value “1” for “on” and “0” for “off”. This differs from the data in the source CSV file, which is “yes” and “no”. You can use the plugin “Convert to boolean” to address this situation. Under “Truth value”, enter yes and under “False value” enter no and save.

**Year**
By default, this field is empty. First, we need to rewrite it using the value from the Date field. Then, we need to extract the year from the date. Add the Rewrite plugin, and enter `[date]`. Save.

Add the “Find replace REGEX” plugin. For “REGEX to find”, enter `/\d\d\d\d/` which looks for four digits together. For “Replacement pattern”, enter $1$ which replaces the value of the field with the four digits you just selected.

14.4.5 Importing
Go to the import page (/import; note that this page is not linked from any menu), and click on the “Event importer from DfH CSV” importer. Upload the CSV you downloaded from the website and click the “Import” button to import the events.
14.5 Importing from RSS feeds
Imagine building the website for a language center that is affiliated with multiple departments. Each department has its own news RSS feed that includes a mix of general announcements and event announcements. Different departments have different conventions for marking events. Some prefix event announcements with “Event”, some use “Events”, and some just mention the words “lecture” or “talk” in the title.

You would need to create two content types: “Departmental feed” and “Departmental event”. Leave “Departmental feed” with the default settings (Title and Body fields), and add a Link field to “Departmental event” called “Event link”. When configuring the link field, set it to have no title, and only one value.

14.5.1 Source data
Imagine the source data consists of RSS feeds from six different departmental sites. Each contains a mix of event and non-event announcements.

14.5.2 Creating the importer and basic configuration
You would create a new feeds importer by going to Structure > Feeds importers > Add importer, where you would give it a reasonable name and description.

Go to “Basic settings”. Under “Attach to content type”, choose “Departmental feed”. Set the periodic import to “Every 1 day”; a few hours’ delay between an event’s posting on the source departmental site and when it appears on the language center’s site isn’t a problem. Save.

The default fetcher (HTTP), parser (Common syndication parser) and processor (node) settings are correct for this example.

In the settings for the node processor, choose “Departmental event” under “Bundle”. Under “Update existing nodes”, choose “Do not update existing nodes”. Change the author to the site admin user. Change the “Expire nodes” setting to “6 months”, on the assumption that events are not announced more than six months in advance. After six months, the nodes created by this feed will be deleted.

14.5.3 Mapping
The common syndication parser, which is used for RSS feeds, has a limited number of predetermined fields as part of the source. Not all feeds will use all these fields, but all data is stored in one of these fields. Because of this, both the sources and the targets will be presented as drop-downs. Create the following mappings:

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Target configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Title</td>
<td>Used as unique</td>
</tr>
<tr>
<td>Item URL (link)</td>
<td>Event link: URL</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Body</td>
<td></td>
</tr>
</tbody>
</table>

14.5.4 Feeds tamper
Title
Add the “Keyword filter” plugin. Each department has different conventions for marking events, and some don’t even mark them explicitly, just mentioning “lecture” or “talk” in the title. Because each line of the text box operates as an “or” (i.e. the keyword filter looks for the word in line 1 of the text box, OR line 2), enter each of these words on its own line within the text box:

Event
Events
lecture
talk

14.5.5 Importing
Go to Content > Add content > Departmental feed. Attached the feeds importer to that content type added a new field to it, “URL” under the header “Feed”. Put in the name of one of the departments, and paste in the URL of the department’s RSS feed, and save. This will fetch the RSS feed, and will create as many new “Departmental event” nodes as there are items that match the keyword filter criteria.

14.6 Importing from XML

14.6.1 Source data
This example uses the TEI-encoded “Wizard of Oz” musical libretto from the New York Public Library as its source data. As in many TEI documents, the encoding is not entirely consistent, and so Feeds Tamper will be used to try to clean up some of the data. While this example uses an XML file, the same general process could be used with an HTML file.

This example illustrates how to extract songs from the libretto and save each as its own node.

Create a content type called “Song”, and add the following fields:
- “Act”, text field with one value
- “Scene”, text field with one value

If this were a real project, you would likely have “Scene” and “Act” content types, and “Scene” would be a node reference field pointing to the correct “Scene” node, and “Act” would be a node reference field within the “Scene” content type. Since the goal of this example is to illustrate extracting data from an XML file using Feeds, simply using text fields here will accomplish that goal, even if it’s unrealistic from the perspective of actual project implementation.

14.6.2 Creating the importer and basic configuration
Create a new feeds importer by going to Structure > Feeds importers > Add importer. For the name, call it “Song importer”, with the description “Imports songs from the Wizard of Oz TEI file.”

Under “Basic settings”, but change “Periodic import” to “Off” and save.

Set the fetcher to “File upload” and the parser to “XML XPath parser”. If you don’t see this option, make sure the Feeds extensible parser module is installed and enabled.

137 This TEI file is available on the Drupal for Humanists site, or at https://github.com/TEI-examples/tei-examples/blob/master/nypl.org/WizardOfOz.xml.
14.6.3 Settings for XML Xpath parser

The XML Xpath parser settings page includes a table similar to the “Mapping” configuration screen. First, you must put in an XPath expression for the “context”. The context tells Feeds how to find each “thing” in the XML document that you want to import as a node. From there, you can add multiple “sources” by putting in an XPath expression relative to the source. The data that can be found using the XPath expression for each source will be available on the “Mapping” configuration screen.

In this example, songs can be found at TEI/text/body/div/div/div[head] (which is to say, within the body of the TEI document, a <div> element inside an act <div> and scene <div>, where that <div> element has a <head> element within it). Inside the value field for Source, enter TEI/text/body/div/div/div[head]. Then, create the following sources with the following values:
- Song title, head
- Content, . (the value here should be a single period)
- Scene, ../@n
- Act, ../.@n

14.6.4 Processor settings, mapping and debugging

The default node processor setting is correct. In the settings for the node processor, choose “Song” under “Bundle”. Under “Update existing nodes”, select “Replace existing nodes”. For the author, choose your own account.

Create the following mappings:

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Target configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Song title</td>
<td>Title</td>
<td>Used as unique</td>
</tr>
<tr>
<td>Scene</td>
<td>Scene</td>
<td></td>
</tr>
<tr>
<td>Act</td>
<td>Act</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>Body</td>
<td></td>
</tr>
</tbody>
</table>

At this point, it’s worth running the import by going to the import page, choosing “Song importer”, uploading the Wizard of Oz XML file, and clicking “Import”. Once it runs, you should see a notification “Created 13 nodes”.

Click on “Content” in the administration menu and look at the nodes that have been imported.

The node titles need to be cleaned up: they are all caps, some include one or more hyphens, some are in quotes, and some end with a period. Also, some of the items imported don’t include the actual lyrics to the song in the body field. These should be removed.

For those nodes that do have the actual song lyrics, the XML should be stripped out; this is the default configuration. At the same time, the easiest way to identify whether the text that gets imported into the body field is song lyrics or other text is to look for <l> (verse line) elements. Go back to editing the parser settings and add a new source, “Content with markup”, with the same XPath value as “Content”. For this new field, however, check the “raw” checkbox.
On the “Mapping” page, add a mapping from “Content with markup” to “Temporary target”.

14.6.5 Feeds tamper
Song title --> Title
Add the “Find replace” plugin. Insert quotation marks in the “Text to find” field, and leave the second text field blank. Add the “Convert case” plugin and choose the default “Title Case” setting. Then, add the “Find replace” plugin, and change the machine name so it doesn’t conflict with the version of the same plugin you added earlier. Insert a hyphen in the “Text to find” field, and leave the second text field blank. Next, add another “Find replace” plugin, change the machine name, and put a period in the “Text to find” field, leaving the second text field blank. Finally, add the “Trim” filter. Leave the text field blank, and leave the default “Both” setting.

This will strip the quotation marks, then convert the titles in all caps to a more reasonable format, with only the first letter of each word capitalized. Then, it will strip out the hyphen characters, then periods. Last, it will remove excess whitespace. If you don’t strip the quotation marks first, the “Convert case” plugin will treat the quotation marks as the first “letter” of the first word, and will fail to capitalize the actual first letter of the word.

If you rerun the import at this point, you’ll notice that it creates numerous new nodes. This is because you’ve set the title field to be unique, and after you’ve stripped out the punctuation marks, the reformatted title no longer matches what you previously imported, and so Drupal won’t update the existing node.

Content with markup --> Temporary target one
Add the plugin “Keyword filter”. Under “Words or phrases to filter on”, enter `<l>` and save. This will exclude from the import any song that does not include line elements.

14.6.6 Importing
If you have already done test imports, you should delete all the previously imported items. Since you added a filter to weed out the songs without line elements, the previously-imported versions of those songs will linger on your site if you don’t delete them. The easiest way to delete them is to just delete all previously imported items by clicking the “Delete items” tab on the import screen, and choosing “Delete”. This time when you run the import, it should only create 7 nodes.

15. Search
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information on this topic. Sections will likely include:

- Drupal’s internal search
- Ways to tweak Drupal’s internal search
- Using Views with exposed filters instead of Drupal’s internal search

16. Taxonomy modules
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information on this topic. Sections will likely include:
• Taxonomy manager module\(^ {138}\) for managing large taxonomies
• Term merge module\(^ {139}\) for merging duplicate terms
• Hierarchical select module\(^ {140}\) for selecting terms using defined taxonomy hierarchies
• Term reference tree module\(^ {141}\) for automatically selecting parent terms when a child term is selected

17. Themes
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information. In summary:
• Prioritize regions (places to put blocks) over color/design when choosing a theme; it’s easier to change the look of a theme than to add a sidebar if your theme doesn’t have one
• If you don’t want to write CSS yourself, but still want to configure your theme, choose one that has multiple configuration options available through the UI
  ○ The AdaptiveTheme\(^ {142}\) family of modules (Sky\(^ {143}\), Corolla\(^ {144}\), Pixture Reloaded\(^ {145}\), AT Commerce\(^ {146}\), and “Footheme”\(^ {147}\) which can be used to modify any of these) are recommended for this purpose. In addition to having many exposed configuration options, they are also responsive (mobile-friendly) themes.
  ○ The list of Drupal 7 themes, sorted by “most installed”, is potentially useful\(^ {148}\). Keep in mind, though, that many of the most installed themes are ones that individual sites heavily modify through custom CSS.

18. Other useful modules
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information. A few likely sections:
• Biblio module\(^ {149}\): lots of good features for managing bibliographies, including Zotero compatibility and ability to display properly-formatted bibliographic listings. Downsides: not extensible (it’s hard to add new fields), doesn’t use Feeds for importing (you have to have BibTex, Pubmed, or other bibliographic formats that are uncommon in the humanities to do an import.)
• Flag\(^ {150}\): useful for creating things like per-user “bookmarks”, “favorites”, etc.

\(^{138}\) https://www.drupal.org/project/taxonomy_manager
\(^{139}\) https://www.drupal.org/project/term_merge
\(^{140}\) https://www.drupal.org/project/hierarchical_select
\(^{141}\) https://www.drupal.org/project/term_reference_tree
\(^{142}\) https://www.drupal.org/project/adaptivetheme
\(^{143}\) https://www.drupal.org/project/sky
\(^{144}\) https://www.drupal.org/project/corolla
\(^{145}\) https://www.drupal.org/project/pixture_reloaded
\(^{146}\) https://www.drupal.org/project/at_commerce
\(^{147}\) https://www.drupal.org/project/footheme
\(^{148}\) https://www.drupal.org/project/project_theme?f%5B0%5D=&f%5B1%5D=&f%5B2%5D=drupal_core%3A1\n03&f%5B3%5D=sm_field_project_type%3Afull&text=&solrsort=iss_project_release_usage+desc&op=Sea\nrch
\(^{149}\) https://www.drupal.org/project/biblio
\(^{150}\) https://www.drupal.org/project/flag
- Book (core module): useful for creating hierarchies of nodes
- Webform\textsuperscript{151}: for creating forms / surveys
- Rabbit hole\textsuperscript{152}: if you have nodes that should never be viewed on their own, but only as part of a view, this module can make that possible
- Path redirect\textsuperscript{153}: if your nodes change paths (e.g. if your Pathauto configuration involves the node title, and nodes may change titles), this module automatically creates a redirect from the old URL to the new one. Can also be used when migrating a site into Drupal (for instance, from HTML) so that old URLs that include ".html" still work.

19. Running and maintaining a Drupal site
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information. Sections will probably include:
- Updates
  - Keeping tabs on module updates
  - Performing module updates
  - Performing core updates
  - Major version upgrades
- Caching
  - Hugely helpful for improving site performance, especially when you have lots of nodes
  - Don’t forget to turn it on for each view (with some exceptions, like Views Bulk Operations views)
  - Views Content Cache (\url{https://www.drupal.org/project/views_content_cache}) can be useful here
- Features
  - What features are
  - How to build a feature
  - How to install a feature
- Backups
  - Backup & migrate module
- Site migration
  - Overview of process
  - Appendix will include full step-by-step instruction
- Multi-site setups
  - Benefits of multi-site installations
  - How to convert stand-alone sites to a multi-site

20. Debugging Drupal
This chapter is still in development as of March 2015. Check the latest version of the course pack for more information. Sections will probably include:
- How to determine whether to use a module
- When -dev versions of modules are a good idea
- How to search the forums
- How to file bug reports
- “Keep calm and clear the cache”

\textsuperscript{151} \url{https://www.drupal.org/project/webform}
\textsuperscript{152} \url{https://www.drupal.org/project/rabbit_hole}
\textsuperscript{153} \url{https://www.drupal.org/project/path_redirect}
• White screen of death
• Contributing to the Drupal community

21. Writing documentation
• What modules are you using, and for what?
• What are your content types, fields, taxonomies for?
• What do your Views do?
• Any major tweaks built into theme code

22. Appendix A: SFTP
Drupal’s interface allows you to upload certain kinds of files — modules, themes, and images / media / documents that have content for your site — to the correct place within the filesystem. For modules that have external dependencies (such as Javascript libraries), there is no way to put those dependencies in the correct place without accessing the filesystem directly. Similarly, while you can update modules and themes through the Drupal interface, you have to update Drupal core by accessing the filesystem.

22.1 Installing software
Unless your Drupal site is hosted on your own computer, you will need to install and use an SFTP client to access the filesystem. There are a number of options for free and open source options, including Filezilla\textsuperscript{154} (Windows, Mac and Linux) and WinSCP\textsuperscript{155} (Windows), Cyberduck\textsuperscript{156} (Mac and Windows) is an attractive modern option.

The Cyberduck website includes a download link for Windows and for Mac, as well as a link to download it through the Mac App Store if you prefer. Download and install the software as usual for your operating system.

22.2 Connection information
Before you use Cyberduck to connect to the server, you need the following information:
• Server name (may also be called “host” or “host name”): the name of the server that is hosting your site. If you’re using inexpensive shared commercial hosting, you may be able to use your site URL for this (e.g. myproject.org). If you’re using a university hosting service, Pantheon, or some other commercial hosting specifically for Drupal (e.g. Acquia hosting), this is likely to be something different than your site URL.
• Username: the username you use to connect to the server; this will probably be different from the username you use to log into Drupal via the web-based interface. If you’re using university hosting, odds are good that you’ll use your university authentication credentials (username and password)
• Password: the password that matches your username when connecting to the server; this will probably be different from the password you use to log into Drupal via the web-based interface.

\textsuperscript{154} https://filezilla-project.org/
\textsuperscript{155} http://winscp.net/eng/index.php
\textsuperscript{156} https://cyberduck.io/?l=en
● (optional) Path: the place on the server you should immediately be taken to once you log in.
● (optional) Port: some hosting systems require that you use a port different than the default to make the connection.

Regardless of whether you have “Path” information (in many cases you won’t), you should find out where the Drupal installation is on the server.

If you’re using university hosting, ask the IT or library staff who helped you get Drupal set up on the server for this information.

If you’re using Pantheon, when you’re looking at the Pantheon dashboard for your site (after you’ve logged into Pantheon, once you’ve clicked on the screenshot thumbnail for your site), there’s a button towards the top labeled “Connection info”. Click on it, and a little box appears with this information; the SFTP information is towards the bottom.
22.3 Connecting

Open Cyberduck, and click on the “Open Connection” button in the upper left:

![Launch Cyberduck](image1)

Figure 22.2. Launching Cyberduck

By default, the connection type is “FTP”. While this may work in some cases (many commercial hosting providers will allow you to connect via FTP), most universities and Pantheon require you to connect via SFTP for increased security. Click and hold the small arrow to the right of “FTP” to see a list of other options, and select “SFTP (SSH File Transfer Protocol)”.

![SFTP connection](image2)
Figure 22.3. Click and hold here to see a list of other connection options, and choose “SFTP (SSH File Transfer Protocol)”. Fill in the fields using the information that you’ve already gathered. In the server field, put in the server name (may also be called host name). Put in the username and password in those fields. By default, the port for SFTP is 22; only change this if your hosting provider requires it. In most cases, leave the fields under “More options” empty; these include the Path field, so if you do have that information from the server administrators, toggle down “More Options” and include it there.

Once you’ve filled in the fields, click “connect”.

Figure 22.4. Filling in the connection fields

After you hit the “Connect” button, if this is your first time connecting to the server, an “Unknown fingerprint” dialog box may pop up:

Figure 22.5. Unknown fingerprint box

Choose “Allow”, and if you don’t want to receive the same notification next time, check the “Always” box first.
22.4 Troubleshooting connection problems
If you see the following error, you probably mistyped the username or password:

![Login failed error](image)

This error indicates that you probably made some error in typing in the server/host name:

![Connection failed error](image)

22.5 Navigating to the correct place
Once you’ve successfully connected to the server, you need to find the directory with your Drupal installation. This will vary depending on your hosting arrangement. If you are using university hosting, ask the system administrator where the Drupal directory is. Their answer will probably take the form of a string of words (names of folders) separated by slashes; this indicates the nested relationship between the folders: each folder is contained within the folder before it in the sequence.

It’s also important to note whether their answer starts with a slash or not. If they tell you Drupal is in “htdocs/drupal”, you can expect to find an “htdocs” folder in the directory that immediately appears when you connect to the server. Double-click it to open it, and you should see a “drupal” folder. Double-click that, and you’ll be in the folder with your Drupal installation. However, if they tell you Drupal is in /var/www/localhost/htdocs/drupal, you’ll first need to navigate to the “root” (top-level) directory of the server before you start looking for the “var” folder. Towards the top of the Cyberduck interface, there’s a dropdown menu that shows your current location on the server. Activate the dropdown menu and select the option (at the bottom) that is labeled with “/”: 
Once you’re in the root directory, you should see a “var” folder, and be able to navigate to your Drupal installation from there.

22.5.1 Where to upload files
In most cases, modules should be uploaded in the sites/all/modules folder within your Drupal installation, **not** the modules folder. Themes, similarly, should be uploaded in sites/all/themes.

22.6 Uploading files
To upload files, drag and drop files or folders from your own computer into the folder where they belong in the SFTP window. Select the file(s) / folder(s) in Finder (on a Mac) or Windows Explorer, and drag and drop them into a blank space (figure X) or into an area that has individual files, rather than one that has folders (figure Y). If you drag and drop files and folders in an area with other folders, if you’re not very careful, you may accidentally upload the data **into** one of the folders, rather than where you want it to be.
Figure 22.10. It’s safe to drag and drop files and folders in an area with individual folders. Avoid dragging and dropping files and folders in an area with other folders.

It can be a little tricky to upload new modules via SFTP, because the modules folder will probably be full of folders, without any individual files or blank space for dragging and dropping new modules. For that reason, it may be easier to navigate to the parent folder for the modules folder (in most cases, sites/all within your Drupal installation) and drag and drop new modules on top of the “modules” folder, so they upload into the “modules” folder.

Figure 22.11. Drag and drop new modules on top of the “modules” folder to upload them within that folder.

22.6.3 Updating Drupal core, modules, themes, etc.
To update Drupal core, modules, and themes via SFTP, download the new version from drupal.org (i.e. from the page for an individual module or theme, or from https://www.drupal.org/project/drupal for Drupal core) and unzip it. For modules and themes, take the resulting folder and drag and drop it into the sites/all/modules or sites/all/themes folder, as if you were uploading it for the first time. Cyberduck will pop up a window asking if you want to overwrite what’s already there:
Click the “Continue” button, and the new version will be uploaded.

To update Drupal core, delete the “sites” folder from the new version of Drupal you have downloaded, then select all files and folders and drag and drop them on top of the files in the folder with your Drupal installation.