const c={light:"#ffffff",dark:"#1b1b1b"};if(window&&document.documentElement){const e=window.localStorage.getItem("theme");e&&(document.documentElement.className=e,document.documentElement.style.backgroundColor=c[e])}

* [Skip to main content](#content)
* [Skip to search](#top-nav-search-input)
* [Skip to select language](#languages-switcher-button)[MDN Web Docs.logo-m{fill:var(--text-link)}](/en-US/)Open main menu
* References[References](/en-US/docs/Web)
* [Overview / Web Technology](/en-US/docs/Web)

[Web technology reference for developers](/en-US/docs/Web)

* [HTML](/en-US/docs/Web/HTML)

[Structure of content on the web](/en-US/docs/Web/HTML)

* [CSS](/en-US/docs/Web/CSS)

[Code used to describe document style](/en-US/docs/Web/CSS)

* [JavaScript](/en-US/docs/Web/JavaScript)

[General-purpose scripting language](/en-US/docs/Web/JavaScript)

* [HTTP](/en-US/docs/Web/HTTP)

[Protocol for transmitting web resources](/en-US/docs/Web/HTTP)

* [Web APIs](/en-US/docs/Web/API)

[Interfaces for building web applications](/en-US/docs/Web/API)

* [Web Technology](/en-US/docs/Web)

[Web technology reference for developers](/en-US/docs/Web)

* Guides[Guides](/en-US/docs/Learn)
* [Overview / MDN Learning Area](/en-US/docs/Learn)

[Learn web development](/en-US/docs/Learn)

* [MDN Learning Area](/en-US/docs/Learn)

[Learn web development](/en-US/docs/Learn)

* [HTML](/en-US/docs/Learn/HTML)

[Learn to structure web content with HTML](/en-US/docs/Learn/HTML)

* [CSS](/en-US/docs/Learn/CSS)

[Learn to style content using CSS](/en-US/docs/Learn/CSS)

* [JavaScript](/en-US/docs/Learn/JavaScript)

[Learn to run scripts in the browser](/en-US/docs/Learn/JavaScript)

* [Accessibility](/en-US/docs/Web/Accessibility)

[Learn to make the web accessible to all](/en-US/docs/Web/Accessibility)

* MDN Plus[MDN Plus](/en-US/plus)
* [Overview](/en-US/plus)

[A customized MDN experience](/en-US/plus)

* [Documentation](/en-US/plus/docs/features/overview)

[Learn how to use MDN Plus](/en-US/plus/docs/features/overview)

* [FAQ](/en-US/plus/docs/faq)

[Frequently asked questions about MDN Plus](/en-US/plus/docs/faq)Search MDNClear search inputSearchTheme

* [Already a subscriber?](/users/fxa/login/authenticate/?next=%2Fen-US%2Fdocs%2FLearn%2FServer-side%2FDjango%2FModels)
* [Get MDN Plus](/en-US/plus)

1. [Guides](/en-US/docs/Learn)
2. [Server-side website programming](/en-US/docs/Learn/Server-side)
3. [Django Web Framework (Python)](/en-US/docs/Learn/Server-side/Django)
4. [Django Tutorial Part 3: Using models](/en-US/docs/Learn/Server-side/Django/Models)Article Actions

* English (US)

## In this article

* [Overview](#overview)
* [Designing the LocalLibrary models](#designing_the_locallibrary_models)
* [Model primer](#model_primer)
* [Defining the LocalLibrary Models](#defining_the_locallibrary_models)
* [Re-run the database migrations](#re-run_the_database_migrations)
* [Language model — challenge](#language_model_—_challenge)
* [Summary](#summary)
* [See also](#see_also)
* [In this module](#in_this_module)

#### Related Topics

1. [Complete beginners start here!](/en-US/docs/Learn/Getting_started_with_the_web)
2. Getting started with the web
3. [Getting started with the web overview](/en-US/docs/Learn/Getting_started_with_the_web)
4. [Installing basic software](/en-US/docs/Learn/Getting_started_with_the_web/Installing_basic_software)
5. [What will your website look like?](/en-US/docs/Learn/Getting_started_with_the_web/What_will_your_website_look_like)
6. [Dealing with files](/en-US/docs/Learn/Getting_started_with_the_web/Dealing_with_files)
7. [HTML basics](/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)
8. [CSS basics](/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)
9. [JavaScript basics](/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics)
10. [Publishing your website](/en-US/docs/Learn/Getting_started_with_the_web/Publishing_your_website)
11. [How the web works](/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works)
12. [HTML — Structuring the web](/en-US/docs/Learn/HTML)
13. Introduction to HTML
14. [Introduction to HTML overview](/en-US/docs/Learn/HTML/Introduction_to_HTML)
15. [Getting started with HTML](/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started)
16. [What's in the head? Metadata in HTML](/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML)
17. [HTML text fundamentals](/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals)
18. [Creating hyperlinks](/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks)
19. [Advanced text formatting](/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting)
20. [Document and website structure](/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure)
21. [Debugging HTML](/en-US/docs/Learn/HTML/Introduction_to_HTML/Debugging_HTML)
22. [Assessment: Marking up a letter](/en-US/docs/Learn/HTML/Introduction_to_HTML/Marking_up_a_letter)
23. [Assessment: Structuring a page of content](/en-US/docs/Learn/HTML/Introduction_to_HTML/Structuring_a_page_of_content)
24. Multimedia and embedding
25. [Multimedia and embedding overview](/en-US/docs/Learn/HTML/Multimedia_and_embedding)
26. [Images in HTML](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML)
27. [Video and audio content](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content)
28. [From object to iframe — other embedding technologies](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Other_embedding_technologies)
29. [Adding vector graphics to the web](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Adding_vector_graphics_to_the_Web)
30. [Responsive images](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images)
31. [Assessment: Mozilla splash page](/en-US/docs/Learn/HTML/Multimedia_and_embedding/Mozilla_splash_page)
32. HTML tables
33. [HTML tables overview](/en-US/docs/Learn/HTML/Tables)
34. [HTML table basics](/en-US/docs/Learn/HTML/Tables/Basics)
35. [HTML table advanced features and accessibility](/en-US/docs/Learn/HTML/Tables/Advanced)
36. [Assessment: Structuring planet data](/en-US/docs/Learn/HTML/Tables/Structuring_planet_data)
37. [CSS — Styling the web](/en-US/docs/Learn/CSS)
38. CSS first steps
39. [CSS first steps overview](/en-US/docs/Learn/CSS/First_steps)
40. [What is CSS?](/en-US/docs/Learn/CSS/First_steps/What_is_CSS)
41. [Getting started with CSS](/en-US/docs/Learn/CSS/First_steps/Getting_started)
42. [How CSS is structured](/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured)
43. [How CSS works](/en-US/docs/Learn/CSS/First_steps/How_CSS_works)
44. [Assessment: Styling a biography page](/en-US/docs/Learn/CSS/First_steps/Styling_a_biography_page)
45. CSS building blocks
46. [CSS building blocks overview](/en-US/docs/Learn/CSS/Building_blocks)
47. [Cascade and inheritance](/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance)
48. [CSS selectors](/en-US/docs/Learn/CSS/Building_blocks/Selectors)
49. [The box model](/en-US/docs/Learn/CSS/Building_blocks/The_box_model)
50. [Backgrounds and borders](/en-US/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders)
51. [Handling different text directions](/en-US/docs/Learn/CSS/Building_blocks/Handling_different_text_directions)
52. [Overflowing content](/en-US/docs/Learn/CSS/Building_blocks/Overflowing_content)
53. [CSS values and units](/en-US/docs/Learn/CSS/Building_blocks/Values_and_units)
54. [Sizing items in CSS](/en-US/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS)
55. [Images, media, and form elements](/en-US/docs/Learn/CSS/Building_blocks/Images_media_form_elements)
56. [Styling tables](/en-US/docs/Learn/CSS/Building_blocks/Styling_tables)
57. [Debugging CSS](/en-US/docs/Learn/CSS/Building_blocks/Debugging_CSS)
58. [Organizing your CSS](/en-US/docs/Learn/CSS/Building_blocks/Organizing)
59. [Assessment: Fundamental CSS comprehension](/en-US/docs/Learn/CSS/Building_blocks/Fundamental_CSS_comprehension)
60. [Assessment: Creating fancy letterheaded paper](/en-US/docs/Learn/CSS/Building_blocks/Creating_fancy_letterheaded_paper)
61. [Assessment: A cool-looking box](/en-US/docs/Learn/CSS/Building_blocks/A_cool_looking_box)
62. Styling text
63. [Styling text overview](/en-US/docs/Learn/CSS/Styling_text)
64. [Fundamental text and font styling](/en-US/docs/Learn/CSS/Styling_text/Fundamentals)
65. [Styling lists](/en-US/docs/Learn/CSS/Styling_text/Styling_lists)
66. [Styling links](/en-US/docs/Learn/CSS/Styling_text/Styling_links)
67. [Web fonts](/en-US/docs/Learn/CSS/Styling_text/Web_fonts)
68. [Assessment: Typesetting a community school homepage](/en-US/docs/Learn/CSS/Styling_text/Typesetting_a_homepage)
69. CSS layout
70. [CSS layout overview](/en-US/docs/Learn/CSS/CSS_layout)
71. [Introduction to CSS layout](/en-US/docs/Learn/CSS/CSS_layout/Introduction)
72. [Normal Flow](/en-US/docs/Learn/CSS/CSS_layout/Normal_Flow)
73. [Flexbox](/en-US/docs/Learn/CSS/CSS_layout/Flexbox)
74. [Grids](/en-US/docs/Learn/CSS/CSS_layout/Grids)
75. [Floats](/en-US/docs/Learn/CSS/CSS_layout/Floats)
76. [Positioning](/en-US/docs/Learn/CSS/CSS_layout/Positioning)
77. [Multiple-column layout](/en-US/docs/Learn/CSS/CSS_layout/Multiple-column_Layout)
78. [Responsive design](/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design)
79. [Beginner's guide to media queries](/en-US/docs/Learn/CSS/CSS_layout/Media_queries)
80. [Legacy layout methods](/en-US/docs/Learn/CSS/CSS_layout/Legacy_Layout_Methods)
81. [Supporting older browsers](/en-US/docs/Learn/CSS/CSS_layout/Supporting_Older_Browsers)
82. [Assessment: Fundamental layout comprehension](/en-US/docs/Learn/CSS/CSS_layout/Fundamental_Layout_Comprehension)
83. [JavaScript — Dynamic client-side scripting](/en-US/docs/Learn/JavaScript)
84. JavaScript first steps
85. [JavaScript first steps overview](/en-US/docs/Learn/JavaScript/First_steps)
86. [What is JavaScript?](/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript)
87. [A first splash into JavaScript](/en-US/docs/Learn/JavaScript/First_steps/A_first_splash)
88. [What went wrong? Troubleshooting JavaScript](/en-US/docs/Learn/JavaScript/First_steps/What_went_wrong)
89. [Storing the information you need — Variables](/en-US/docs/Learn/JavaScript/First_steps/Variables)
90. [Basic math in JavaScript — Numbers and operators](/en-US/docs/Learn/JavaScript/First_steps/Math)
91. [Handling text — Strings in JavaScript](/en-US/docs/Learn/JavaScript/First_steps/Strings)
92. [Useful string methods](/en-US/docs/Learn/JavaScript/First_steps/Useful_string_methods)
93. [Arrays](/en-US/docs/Learn/JavaScript/First_steps/Arrays)
94. [Assessment: Silly story generator](/en-US/docs/Learn/JavaScript/First_steps/Silly_story_generator)
95. JavaScript building blocks
96. [JavaScript building blocks overview](/en-US/docs/Learn/JavaScript/Building_blocks)
97. [Making decisions in your code — Conditionals](/en-US/docs/Learn/JavaScript/Building_blocks/conditionals)
98. [Looping code](/en-US/docs/Learn/JavaScript/Building_blocks/Looping_code)
99. [Functions — Reusable blocks of code](/en-US/docs/Learn/JavaScript/Building_blocks/Functions)
100. [Build your own function](/en-US/docs/Learn/JavaScript/Building_blocks/Build_your_own_function)
101. [Function return values](/en-US/docs/Learn/JavaScript/Building_blocks/Return_values)
102. [Introduction to events](/en-US/docs/Learn/JavaScript/Building_blocks/Events)
103. [Assessment: Image gallery](/en-US/docs/Learn/JavaScript/Building_blocks/Image_gallery)
104. Introducing JavaScript objects
105. [Introducing JavaScript objects overview](/en-US/docs/Learn/JavaScript/Objects)
106. [Object basics](/en-US/docs/Learn/JavaScript/Objects/Basics)
107. [Object prototypes](/en-US/docs/Learn/JavaScript/Objects/Object_prototypes)
108. [Object-oriented programming concepts](/en-US/docs/Learn/JavaScript/Objects/Object-oriented_programming)
109. [Classes in JavaScript](/en-US/docs/Learn/JavaScript/Objects/Classes_in_JavaScript)
110. [Working with JSON data](/en-US/docs/Learn/JavaScript/Objects/JSON)
111. [Object building practice](/en-US/docs/Learn/JavaScript/Objects/Object_building_practice)
112. [Assessment: Adding features to our bouncing balls demo](/en-US/docs/Learn/JavaScript/Objects/Adding_bouncing_balls_features)
113. Asynchronous JavaScript
114. [Asynchronous JavaScript overview](/en-US/docs/Learn/JavaScript/Asynchronous)
115. [Introducing asynchronous JavaScript](/en-US/docs/Learn/JavaScript/Asynchronous/Introducing)
116. [How to use promises](/en-US/docs/Learn/JavaScript/Asynchronous/Promises)
117. [Implementing a promise-based API](/en-US/docs/Learn/JavaScript/Asynchronous/Implementing_a_promise-based_API)
118. [Introducing workers](/en-US/docs/Learn/JavaScript/Asynchronous/Introducing_workers)
119. [Assessment: sequencing animations](/en-US/docs/Learn/JavaScript/Asynchronous/Sequencing_animations)
120. Client-side web APIs
121. [Client-side web APIs](/en-US/docs/Learn/JavaScript/Client-side_web_APIs)
122. [Introduction to web APIs](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction)
123. [Manipulating documents](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents)
124. [Fetching data from the server](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data)
125. [Third party APIs](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Third_party_APIs)
126. [Drawing graphics](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Drawing_graphics)
127. [Video and audio APIs](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Video_and_audio_APIs)
128. [Client-side storage](/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Client-side_storage)
129. [Web forms — Working with user data](/en-US/docs/Learn/Forms)
130. Core forms learning pathway
131. [Web forms overview](/en-US/docs/Learn/Forms)
132. [Your first form](/en-US/docs/Learn/Forms/Your_first_form)
133. [How to structure a web form](/en-US/docs/Learn/Forms/How_to_structure_a_web_form)
134. [Basic native form controls](/en-US/docs/Learn/Forms/Basic_native_form_controls)
135. [The HTML5 input types](/en-US/docs/Learn/Forms/HTML5_input_types)
136. [Other form controls](/en-US/docs/Learn/Forms/Other_form_controls)
137. [Styling web forms](/en-US/docs/Learn/Forms/Styling_web_forms)
138. [Advanced form styling](/en-US/docs/Learn/Forms/Advanced_form_styling)
139. [UI pseudo-classes](/en-US/docs/Learn/Forms/UI_pseudo-classes)
140. [Client-side form validation](/en-US/docs/Learn/Forms/Form_validation)
141. [Sending form data](/en-US/docs/Learn/Forms/Sending_and_retrieving_form_data)
142. Advanced forms articles
143. [How to build custom form controls](/en-US/docs/Learn/Forms/How_to_build_custom_form_controls)
144. [Sending forms through JavaScript](/en-US/docs/Learn/Forms/Sending_forms_through_JavaScript)
145. [CSS property compatibility table for form controls](/en-US/docs/Learn/Forms/Property_compatibility_table_for_form_controls)
146. [Accessibility — Make the web usable by everyone](/en-US/docs/Learn/Accessibility)
147. Accessibility guides
148. [Accessibility overview](/en-US/docs/Learn/Accessibility)
149. [What is accessibility?](/en-US/docs/Learn/Accessibility/What_is_accessibility)
150. [HTML: A good basis for accessibility](/en-US/docs/Learn/Accessibility/HTML)
151. [CSS and JavaScript accessibility best practices](/en-US/docs/Learn/Accessibility/CSS_and_JavaScript)
152. [WAI-ARIA basics](/en-US/docs/Learn/Accessibility/WAI-ARIA_basics)
153. [Accessible multimedia](/en-US/docs/Learn/Accessibility/Multimedia)
154. [Mobile accessibility](/en-US/docs/Learn/Accessibility/Mobile)
155. Accessibility assessment
156. [Assessment: Accessibility troubleshooting](/en-US/docs/Learn/Accessibility/Accessibility_troubleshooting)
157. [Tools and testing](/en-US/docs/Learn/Tools_and_testing)
158. Client-side web development tools
159. [Client-side web development tools index](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools)
160. [Client-side tooling overview](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Overview)
161. [Command line crash course](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Command_line)
162. [Package management basics](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Package_management)
163. [Introducing a complete toolchain](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Introducing_complete_toolchain)
164. [Deploying our app](/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Deployment)
165. Introduction to client-side frameworks
166. [Client-side frameworks overview](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction)
167. [Framework main features](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Main_features)
168. React
169. [Getting started with React](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started)
170. [Beginning our React todo list](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_todo_list_beginning)
171. [Componentizing our React app](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_components)
172. [React interactivity: Events and state](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_interactivity_events_state)
173. [React interactivity: Editing, filtering, conditional rendering](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_interactivity_filtering_conditional_rendering)
174. [Accessibility in React](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_accessibility)
175. [React resources](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_resources)
176. Ember
177. [Getting started with Ember](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_getting_started)
178. [Ember app structure and componentization](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_structure_componentization)
179. [Ember interactivity: Events, classes and state](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_interactivity_events_state)
180. [Ember Interactivity: Footer functionality, conditional rendering](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_conditional_footer)
181. [Routing in Ember](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_routing)
182. [Ember resources and troubleshooting](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Ember_resources)
183. Vue
184. [Getting started with Vue](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_getting_started)
185. [Creating our first Vue component](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_first_component)
186. [Rendering a list of Vue components](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_rendering_lists)
187. [Adding a new todo form: Vue events, methods, and models](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_methods_events_models)
188. [Styling Vue components with CSS](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_styling)
189. [Using Vue computed properties](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_computed_properties)
190. [Vue conditional rendering: editing existing todos](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_conditional_rendering  )
191. [Focus management with Vue refs](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_refs_focus_management)
192. [Vue resources](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_resources)
193. Svelte
194. [Getting started with Svelte](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_getting_started)
195. [Starting our Svelte Todo list app](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_Todo_list_beginning)
196. [Dynamic behavior in Svelte: working with variables and props](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_variables_props)
197. [Componentizing our Svelte app](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_components)
198. [Advanced Svelte: Reactivity, lifecycle, accessibility](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_reactivity_lifecycle_accessibility)
199. [Working with Svelte stores](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_stores)
200. [TypeScript support in Svelte](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_TypeScript  )
201. [Deployment and next steps](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_deployment_next)
202. Angular
203. [Getting started with Angular](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_getting_started)
204. [Beginning our Angular todo list app](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_todo_list_beginning)
205. [Styling our Angular app](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_styling)
206. [Creating an item component](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_item_component)
207. [Filtering our to-do items](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_filtering)
208. [Building Angular applications and further resources](/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_building)
209. Git and GitHub
210. [Git and GitHub overview](/en-US/docs/Learn/Tools_and_testing/GitHub)
211. [Hello World](https://guides.github.com/activities/hello-world/)
212. [Git Handbook](https://guides.github.com/introduction/git-handbook/)
213. [Forking Projects](https://guides.github.com/activities/forking/)
214. [About pull requests](https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests)
215. [Mastering Issues](https://guides.github.com/features/issues/)
216. Cross browser testing
217. [Cross browser testing overview](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing)
218. [Introduction to cross browser testing](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Introduction)
219. [Strategies for carrying out testing](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Testing_strategies)
220. [Handling common HTML and CSS problems](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/HTML_and_CSS)
221. [Handling common JavaScript problems](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/JavaScript)
222. [Handling common accessibility problems](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Accessibility)
223. [Implementing feature detection](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Feature_detection)
224. [Introduction to automated testing](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Automated_testing)
225. [Setting up your own test automation environment](/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Your_own_automation_environment)
226. [Server-side website programming](/en-US/docs/Learn/Server-side)
227. First steps
228. [First steps overview](/en-US/docs/Learn/Server-side/First_steps)
229. [Introduction to the server-side](/en-US/docs/Learn/Server-side/First_steps/Introduction)
230. [Client-Server overview](/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview)
231. [Server-side web frameworks](/en-US/docs/Learn/Server-side/First_steps/Web_frameworks)
232. [Website security](/en-US/docs/Learn/Server-side/First_steps/Website_security)
233. Django web framework (Python)
234. [Django web framework (Python) overview](/en-US/docs/Learn/Server-side/Django)
235. [Introduction](/en-US/docs/Learn/Server-side/Django/Introduction)
236. [Setting up a development environment](/en-US/docs/Learn/Server-side/Django/development_environment)
237. [Tutorial: The Local Library website](/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website)
238. [Tutorial Part 2: Creating a skeleton website](/en-US/docs/Learn/Server-side/Django/skeleton_website)
239. [Tutorial Part 3: Using models](/en-US/docs/Learn/Server-side/Django/Models)
240. [Tutorial Part 4: Django admin site](/en-US/docs/Learn/Server-side/Django/Admin_site)
241. [Tutorial Part 5: Creating our home page](/en-US/docs/Learn/Server-side/Django/Home_page)
242. [Tutorial Part 6: Generic list and detail views](/en-US/docs/Learn/Server-side/Django/Generic_views)
243. [Tutorial Part 7: Sessions framework](/en-US/docs/Learn/Server-side/Django/Sessions)
244. [Tutorial Part 8: User authentication and permissions](/en-US/docs/Learn/Server-side/Django/Authentication)
245. [Tutorial Part 9: Working with forms](/en-US/docs/Learn/Server-side/Django/Forms)
246. [Tutorial Part 10: Testing a Django web application](/en-US/docs/Learn/Server-side/Django/Testing)
247. [Tutorial Part 11: Deploying Django to production](/en-US/docs/Learn/Server-side/Django/Deployment)
248. [Web application security](/en-US/docs/Learn/Server-side/Django/web_application_security)
249. [Assessment: DIY mini blog](/en-US/docs/Learn/Server-side/Django/django_assessment_blog)
250. Express Web Framework (node.js/JavaScript)
251. [Express Web Framework (Node.js/JavaScript) overview](/en-US/docs/Learn/Server-side/Express_Nodejs)
252. [Express/Node introduction](/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction)
253. [Setting up a Node (Express) development environment](/en-US/docs/Learn/Server-side/Express_Nodejs/development_environment)
254. [Express tutorial: The Local Library website](/en-US/docs/Learn/Server-side/Express_Nodejs/Tutorial_local_library_website)
255. [Express Tutorial Part 2: Creating a skeleton website](/en-US/docs/Learn/Server-side/Express_Nodejs/skeleton_website)
256. [Express Tutorial Part 3: Using a database (with Mongoose)](/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose)
257. [Express Tutorial Part 4: Routes and controllers](/en-US/docs/Learn/Server-side/Express_Nodejs/routes)
258. [Express Tutorial Part 5: Displaying library data](/en-US/docs/Learn/Server-side/Express_Nodejs/Displaying_data)
259. [Express Tutorial Part 6: Working with forms](/en-US/docs/Learn/Server-side/Express_Nodejs/forms)
260. [Express Tutorial Part 7: Deploying to production](/en-US/docs/Learn/Server-side/Express_Nodejs/deployment)
261. [Further resources](#)
262. Common questions
263. [HTML questions](/en-US/docs/Learn/HTML/Howto)
264. [CSS questions](/en-US/docs/Learn/CSS/Howto)
265. [JavaScript questions](/en-US/docs/Learn/JavaScript/Howto)
266. [Web mechanics](/en-US/docs/Learn/Common_questions#Web_mechanics)
267. [Tools and setup](/en-US/docs/Learn/Common_questions#Tools_and_setup)
268. [Design and accessibility](/en-US/docs/Learn/Common_questions#Design_and_accessibility)

## In this article

* [Overview](#overview)
* [Designing the LocalLibrary models](#designing_the_locallibrary_models)
* [Model primer](#model_primer)
* [Defining the LocalLibrary Models](#defining_the_locallibrary_models)
* [Re-run the database migrations](#re-run_the_database_migrations)
* [Language model — challenge](#language_model_—_challenge)
* [Summary](#summary)
* [See also](#see_also)
* [In this module](#in_this_module)

# Django Tutorial Part 3: Using models

* [Previous](/en-US/docs/Learn/Server-side/Django/skeleton_website)
* [Overview: Django](/en-US/docs/Learn/Server-side/Django)
* [Next](/en-US/docs/Learn/Server-side/Django/Admin_site)

This article shows how to define models for the LocalLibrary website. It explains what a model is, how it is declared, and some of the main field types. It also briefly shows a few of the main ways you can access model data.

|  |  |
| --- | --- |
| **Prerequisites:** | [Django Tutorial Part 2: Creating a skeleton website](/en-US/docs/Learn/Server-side/Django/skeleton_website). |
| **Objective:** | To be able to design and create your own models, choosing fields appropriately. |

## [Overview](#overview)

Django web applications access and manage data through Python objects referred to as models. Models define the*structure*of stored data, including the field*types*and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. The definition of the model is independent of the underlying database — you can choose one of several as part of your project settings. Once you've chosen what database you want to use, you don't need to talk to it directly at all — you just write your model structure and other code, and Django handles all the dirty work of communicating with the database for you.

This tutorial shows how to define and access the models for the[LocalLibrary website](/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website)example.

## [Designing the LocalLibrary models](#designing_the_locallibrary_models)

Before you jump in and start coding the models, it's worth taking a few minutes to think about what data we need to store and the relationships between the different objects.

We know that we need to store information about books (title, summary, author, written language, category, ISBN) and that we might have multiple copies available (with globally unique id, availability status, etc.). We might need to store more information about the author than just their name, and there might be multiple authors with the same or similar names. We want to be able to sort information based on book title, author, written language, and category.

When designing your models, it makes sense to have separate models for every "object" (a group of related information). In this case, the obvious objects are books, book instances, and authors.

You might also want to use models to represent selection-list options (e.g. like a drop down list of choices), rather than hard coding the choices into the website itself — this is recommended when all the options aren't known up front or may change. Obvious candidates for models, in this case, include the book genre (e.g. Science Fiction, French Poetry, etc.) and language (English, French, Japanese).

Once we've decided on our models and field, we need to think about the relationships. Django allows you to define relationships that are one to one (OneToOneField), one to many (ForeignKey) and many to many (ManyToManyField).

With that in mind, the UML association diagram below shows the models we'll define in this case (as boxes).

<image: local\_library\_model\_uml.svg>

We've created models for the book (the generic details of the book), book instance (status of specific physical copies of the book available in the system), and author. We have also decided to have a model for the genre so that values can be created/selected through the admin interface. We've decided not to have a model for theBookInstance:status— we've hardcoded the values (LOAN\_STATUS) because we don't expect these to change. Within each of the boxes, you can see the model name, the field names, and types, and also the methods and their return types.

The diagram also shows the relationships between the models, including their*multiplicities*. The multiplicities are the numbers on the diagram showing the numbers (maximum and minimum) of each model that may be present in the relationship. For example, the connecting line between the boxes shows that Book and a Genre are related. The numbers close to the Genre model show that a book must have one or more Genres (as many as you like), while the numbers on the other end of the line next to the Book model show that a Genre can have zero or many associated books.

**Note:**The next section provides a basic primer explaining how models are defined and used. As you read it, consider how we will construct each of the models in the diagram above.

## [Model primer](#model_primer)

This section provides a brief overview of how a model is defined and some of the more important fields and field arguments.

### [Model definition](#model_definition)

Models are usually defined in an application's**models.py**file. They are implemented as subclasses ofdjango.db.models.Model, and can include fields, methods and metadata. The code fragment below shows a "typical" model, namedMyModelName:

from django.db import models  
from django.urls import reverse  
  
class MyModelName(models.Model):  
 """A typical class defining a model, derived from the Model class."""  
  
 # Fields  
 my\_field\_name = models.CharField(max\_length=20, help\_text='Enter field documentation')  
 # …  
  
 # Metadata  
 class Meta:  
 ordering = ['-my\_field\_name']  
  
 # Methods  
 def get\_absolute\_url(self):  
 """Returns the URL to access a particular instance of MyModelName."""  
 return reverse('model-detail-view', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 """String for representing the MyModelName object (in Admin site etc.)."""  
 return self.my\_field\_name

In the below sections we'll explore each of the features inside the model in detail:

#### Fields

A model can have an arbitrary number of fields, of any type — each one represents a column of data that we want to store in one of our database tables. Each database record (row) will consist of one of each field value. Let's look at the example seen below:

my\_field\_name = models.CharField(max\_length=20, help\_text='Enter field documentation')

Our above example has a single field calledmy\_field\_name, of typemodels.CharField— which means that this field will contain strings of alphanumeric characters. The field types are assigned using specific classes, which determine the type of record that is used to store the data in the database, along with validation criteria to be used when values are received from an HTML form (i.e. what constitutes a valid value). The field types can also take arguments that further specify how the field is stored or can be used. In this case we are giving our field two arguments:

* max\_length=20— States that the maximum length of a value in this field is 20 characters.
* help\_text='Enter field documentation'— helpful text that may be displayed in a form to help users understand how the field is used.

The field name is used to refer to it in queries and templates. Fields also have a label, which is specified using theverbose\_nameargument (with a default value ofNone). Ifverbose\_nameis not set, the label is created from the field name by replacing any underscores with a space, and capitalizing the first letter (for example, the fieldmy\_field\_namewould have a default label of*My field name*when used in forms).

The order that fields are declared will affect their default order if a model is rendered in a form (e.g. in the Admin site), though this may be overridden.

##### Common field arguments

The following common arguments can be used when declaring many/most of the different field types:

* [help\_text](https://docs.djangoproject.com/en/4.0/ref/models/fields/#help-text): Provides a text label for HTML forms (e.g. in the admin site), as described above.
* [verbose\_name](https://docs.djangoproject.com/en/4.0/ref/models/fields/#verbose-name): A human-readable name for the field used in field labels. If not specified, Django will infer the default verbose name from the field name.
* [default](https://docs.djangoproject.com/en/4.0/ref/models/fields/#default): The default value for the field. This can be a value or a callable object, in which case the object will be called every time a new record is created.
* [null](https://docs.djangoproject.com/en/4.0/ref/models/fields/#null): IfTrue, Django will store blank values asNULLin the database for fields where this is appropriate (aCharFieldwill instead store an empty string). The default isFalse.
* [blank](https://docs.djangoproject.com/en/4.0/ref/models/fields/#blank): IfTrue, the field is allowed to be blank in your forms. The default isFalse, which means that Django's form validation will force you to enter a value. This is often used withnull=True, because if you're going to allow blank values, you also want the database to be able to represent them appropriately.
* [choices](https://docs.djangoproject.com/en/4.0/ref/models/fields/#choices): A group of choices for this field. If this is provided, the default corresponding form widget will be a select box with these choices instead of the standard text field.
* [primary\_key](https://docs.djangoproject.com/en/4.0/ref/models/fields/#primary-key): IfTrue, sets the current field as the primary key for the model (A primary key is a special database column designated to uniquely identify all the different table records). If no field is specified as the primary key, Django will automatically add a field for this purpose. The type of auto-created primary key fields can be specified for each app in[AppConfig.default\_auto\_field](https://docs.djangoproject.com/en/4.0/ref/applications/#django.apps.AppConfig.default_auto_field)or globally in the[DEFAULT\_AUTO\_FIELD](https://docs.djangoproject.com/en/4.0/ref/settings/#std:setting-DEFAULT_AUTO_FIELD)setting.

**Note:**Apps created using**manage.py**set the type of the primary key to a[BigAutoField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#bigautofield). You can see this in the local library**catalog/apps.py**file:

class CatalogConfig(AppConfig):  
 default\_auto\_field = 'django.db.models.BigAutoField'

There are many other options — you can view the[full list of field options here](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-options).

##### Common field types

The following list describes some of the more commonly used types of fields.

* [CharField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.CharField)is used to define short-to-mid sized fixed-length strings. You must specify themax\_lengthof the data to be stored.
* [TextField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.TextField)is used for large arbitrary-length strings. You may specify amax\_lengthfor the field, but this is used only when the field is displayed in forms (it is not enforced at the database level).
* [IntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.IntegerField)is a field for storing integer (whole number) values, and for validating entered values as integers in forms.
* [DateField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#datefield)and[DateTimeField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#datetimefield)are used for storing/representing dates and date/time information (as Pythondatetime.dateanddatetime.datetimeobjects, respectively). These fields can additionally declare the (mutually exclusive) parametersauto\_now=True(to set the field to the current date every time the model is saved),auto\_now\_add(to only set the date when the model is first created) , anddefault(to set a default date that can be overridden by the user).
* [EmailField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#emailfield)is used to store and validate email addresses.
* [FileField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#filefield)and[ImageField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#imagefield)are used to upload files and images respectively (theImageFieldadds additional validation that the uploaded file is an image). These have parameters to define how and where the uploaded files are stored.
* [AutoField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#autofield)is a special type ofIntegerFieldthat automatically increments. A primary key of this type is automatically added to your model if you don't explicitly specify one.
* [ForeignKey](https://docs.djangoproject.com/en/4.0/ref/models/fields/#foreignkey)is used to specify a one-to-many relationship to another database model (e.g. a car has one manufacturer, but a manufacturer can make many cars). The "one" side of the relationship is the model that contains the "key" (models containing a "foreign key" referring to that "key", are on the "many" side of such a relationship).
* [ManyToManyField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#manytomanyfield)is used to specify a many-to-many relationship (e.g. a book can have several genres, and each genre can contain several books). In our library app we will use these very similarly toForeignKeys, but they can be used in more complicated ways to describe the relationships between groups. These have the parameteron\_deleteto define what happens when the associated record is deleted (e.g. a value ofmodels.SET\_NULLwould set the value toNULL).

There are many other types of fields, including fields for different types of numbers (big integers, small integers, floats), booleans, URLs, slugs, unique ids, and other "time-related" information (duration, time, etc.). You can view the[full list here](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-types).

#### Metadata

You can declare model-level metadata for your Model by declaringclass Meta, as shown.

class Meta:  
 ordering = ['-my\_field\_name']

One of the most useful features of this metadata is to control the*default ordering*of records returned when you query the model type. You do this by specifying the match order in a list of field names to theorderingattribute, as shown above. The ordering will depend on the type of field (character fields are sorted alphabetically, while date fields are sorted in chronological order). As shown above, you can prefix the field name with a minus symbol (-) to reverse the sorting order.

So as an example, if we chose to sort books like this by default:

ordering = ['title', '-pubdate']

the books would be sorted alphabetically by title, from A-Z, and then by publication date inside each title, from newest to oldest.

Another common attribute isverbose\_name, a verbose name for the class in singular and plural form:

verbose\_name = 'BetterName'

Other useful attributes allow you to create and apply new "access permissions" for the model (default permissions are applied automatically), allow ordering based on another field, or to declare that the class is "abstract" (a base class that you cannot create records for, and will instead be derived from to create other models).

Many of the other metadata options control what database must be used for the model and how the data is stored (these are really only useful if you need to map a model to an existing database).

The full list of metadata options are available here:[Model metadata options](https://docs.djangoproject.com/en/4.0/ref/models/options/)(Django docs).

#### Methods

A model can also have methods.

**Minimally, in every model you should define the standard Python class method\_\_str\_\_()to return a human-readable string for each object.**This string is used to represent individual records in the administration site (and anywhere else you need to refer to a model instance). Often this will return a title or name field from the model.

def \_\_str\_\_(self):  
 return self.field\_name

Another common method to include in Django models isget\_absolute\_url(), which returns a URL for displaying individual model records on the website (if you define this method then Django will automatically add a "View on Site" button to the model's record editing screens in the Admin site). A typical pattern forget\_absolute\_url()is shown below.

def get\_absolute\_url(self):  
 """Returns the URL to access a particular instance of the model."""  
 return reverse('model-detail-view', args=[str(self.id)])

**Note:**Assuming you will use URLs like/myapplication/mymodelname/2to display individual records for your model (where "2" is theidfor a particular record), you will need to create a URL mapper to pass the response and id to a "model detail view" (which will do the work required to display the record). Thereverse()function above is able to "reverse" your URL mapper (in the above case named*'model-detail-view'*) in order to create a URL of the right format.

Of course to make this work you still have to write the URL mapping, view, and template!

You can also define any other methods you like, and call them from your code or templates (provided that they don't take any parameters).

### [Model management](#model_management)

Once you've defined your model classes you can use them to create, update, or delete records, and to run queries to get all records or particular subsets of records. We'll show you how to do that in the tutorial when we define our views, but here is a brief summary.

#### Creating and modifying records

To create a record you can define an instance of the model and then callsave().

# Create a new record using the model's constructor.  
record = MyModelName(my\_field\_name="Instance #1")  
  
# Save the object into the database.  
record.save()

**Note:**If you haven't declared any field as aprimary\_key, the new record will be given one automatically, with the field nameid. You could query this field after saving the above record, and it would have a value of 1.

You can access the fields in this new record using the dot syntax, and change the values. You have to callsave()to store modified values to the database.

# Access model field values using Python attributes.  
print(record.id) # should return 1 for the first record.  
print(record.my\_field\_name) # should print 'Instance #1'  
  
# Change record by modifying the fields, then calling save().  
record.my\_field\_name = "New Instance Name"  
record.save()

#### Searching for records

You can search for records that match certain criteria using the model'sobjectsattribute (provided by the base class).

**Note:**Explaining how to search for records using "abstract" model and field names can be a little confusing. In the discussion below, we'll refer to aBookmodel withtitleandgenrefields, where genre is also a model with a single fieldname.

We can get all records for a model as aQuerySet, usingobjects.all(). TheQuerySetis an iterable object, meaning that it contains a number of objects that we can iterate/loop through.

all\_books = Book.objects.all()

Django'sfilter()method allows us to filter the returnedQuerySetto match a specified**text**or**numeric**field against particular criteria. For example, to filter for books that contain "wild" in the title and then count them, we could do the following.

wild\_books = Book.objects.filter(title\_\_contains='wild')  
number\_wild\_books = wild\_books.count()

The fields to match and the type of match are defined in the filter parameter name, using the format:field\_name\_\_match\_type(note the*double underscore*betweentitleandcontainsabove). Above we're filteringtitlewith a case-sensitive match. There are many other types of matches you can do:icontains(case insensitive),iexact(case-insensitive exact match),exact(case-sensitive exact match) andin,gt(greater than),startswith, etc. The[full list is here](https://docs.djangoproject.com/en/4.0/ref/models/querysets/#field-lookups).

In some cases, you'll need to filter on a field that defines a one-to-many relationship to another model (e.g. aForeignKey). In this case, you can "index" to fields within the related model with additional double underscores. So for example to filter for books with a specific genre pattern, you will have to index to thenamethrough thegenrefield, as shown below:

# Will match on: Fiction, Science fiction, non-fiction etc.  
books\_containing\_genre = Book.objects.filter(genre\_\_name\_\_icontains='fiction')

**Note:**You can use underscores (\_\_) to navigate as many levels of relationships (ForeignKey/ManyToManyField) as you like. For example, aBookthat had different types, defined using a further "cover" relationship might have a parameter name:type\_\_cover\_\_name\_\_exact='hard'.

There is a lot more you can do with queries, including backwards searches from related models, chaining filters, returning a smaller set of values, etc. For more information, see[Making queries](https://docs.djangoproject.com/en/4.0/topics/db/queries/)(Django Docs).

## [Defining the LocalLibrary Models](#defining_the_locallibrary_models)

In this section we will start defining the models for the library. Openmodels.py(in /locallibrary/catalog/). The boilerplate at the top of the page imports the*models*module, which contains the model base classmodels.Modelthat our models will inherit from.

from django.db import models  
  
# Create your models here.

### [Genre model](#genre_model)

Copy theGenremodel code shown below and paste it into the bottom of yourmodels.pyfile. This model is used to store information about the book category — for example whether it is fiction or non-fiction, romance or military history, etc. As mentioned above, we've created the genre as a model rather than as free text or a selection list so that the possible values can be managed through the database rather than being hard coded.

class Genre(models.Model):  
 """Model representing a book genre."""  
 name = models.CharField(max\_length=200, help\_text='Enter a book genre (e.g. Science Fiction)')  
  
 def \_\_str\_\_(self):  
 """String for representing the Model object."""  
 return self.name

The model has a singleCharFieldfield (name), which is used to describe the genre (this is limited to 200 characters and has somehelp\_text). At the end of the model, we declare a\_\_str\_\_()method, which returns the name of the genre defined by a particular record. No verbose name has been defined, so the field will be calledNamein forms.

### [Book model](#book_model)

Copy theBookmodel below and again paste it into the bottom of your file. TheBookmodel represents all information about an available book in a general sense, but not a particular physical "instance" or "copy" available for loan. The model uses aCharFieldto represent the book'stitleandisbn. Forisbn, note how the first unnamed parameter explicitly sets the label as "ISBN" (otherwise, it would default to "Isbn"). We also set the parameteruniqueastrueto ensure all books have a unique ISBN (the unique parameter makes the field value globally unique in a table). The model usesTextFieldfor thesummary, because this text may need to be quite long.

from django.urls import reverse # Used to generate URLs by reversing the URL patterns  
  
class Book(models.Model):  
 """Model representing a book (but not a specific copy of a book)."""  
 title = models.CharField(max\_length=200)  
  
 # Foreign Key used because book can only have one author, but authors can have multiple books  
 # Author is a string rather than an object because it hasn't been declared yet in the file  
 author = models.ForeignKey('Author', on\_delete=models.SET\_NULL, null=True)  
  
 summary = models.TextField(max\_length=1000, help\_text='Enter a brief description of the book')  
 isbn = models.CharField('ISBN', max\_length=13, unique=True,  
 help\_text='13 Character <a href="https://www.isbn-international.org/content/what-isbn">ISBN number</a>')  
  
 # ManyToManyField used because genre can contain many books. Books can cover many genres.  
 # Genre class has already been defined so we can specify the object above.  
 genre = models.ManyToManyField(Genre, help\_text='Select a genre for this book')  
  
 def \_\_str\_\_(self):  
 """String for representing the Model object."""  
 return self.title  
  
 def get\_absolute\_url(self):  
 """Returns the URL to access a detail record for this book."""  
 return reverse('book-detail', args=[str(self.id)])

The genre is aManyToManyField, so that a book can have multiple genres and a genre can have many books. The author is declared asForeignKey, so each book will only have one author, but an author may have many books (in practice a book might have multiple authors, but not in this implementation!)

In both field types the related model class is declared as the first unnamed parameter using either the model class or a string containing the name of the related model. You must use the name of the model as a string if the associated class has not yet been defined in this file before it is referenced! The other parameters of interest in theauthorfield arenull=True, which allows the database to store aNullvalue if no author is selected, andon\_delete=models.SET\_NULL, which will set the value of the book's author field toNullif the associated author record is deleted.

**Warning:**By defaulton\_delete=models.CASCADE, which means that if the author was deleted, this book would be deleted too! We useSET\_NULLhere, but we could also usePROTECTorRESTRICTto prevent the author being deleted while any book uses it.

The model also defines\_\_str\_\_(), using the book'stitlefield to represent aBookrecord. The final method,get\_absolute\_url()returns a URL that can be used to access a detail record for this model (for this to work, we will have to define a URL mapping that has the namebook-detail, and define an associated view and template).

### [BookInstance model](#bookinstance_model)

Next, copy theBookInstancemodel (shown below) under the other models. TheBookInstancerepresents a specific copy of a book that someone might borrow, and includes information about whether the copy is available or on what date it is expected back, "imprint" or version details, and a unique id for the book in the library.

Some of the fields and methods will now be familiar. The model uses:

* ForeignKeyto identify the associatedBook(each book can have many copies, but a copy can only have oneBook). The key specifieson\_delete=models.RESTRICTto ensure that theBookcannot be deleted while referenced by aBookInstance.
* CharFieldto represent the imprint (specific release) of the book.

import uuid # Required for unique book instances  
  
class BookInstance(models.Model):  
 """Model representing a specific copy of a book (i.e. that can be borrowed from the library)."""  
 id = models.UUIDField(primary\_key=True, default=uuid.uuid4, help\_text='Unique ID for this particular book across whole library')  
 book = models.ForeignKey('Book', on\_delete=models.RESTRICT, null=True)  
 imprint = models.CharField(max\_length=200)  
 due\_back = models.DateField(null=True, blank=True)  
  
 LOAN\_STATUS = (  
 ('m', 'Maintenance'),  
 ('o', 'On loan'),  
 ('a', 'Available'),  
 ('r', 'Reserved'),  
 )  
  
 status = models.CharField(  
 max\_length=1,  
 choices=LOAN\_STATUS,  
 blank=True,  
 default='m',  
 help\_text='Book availability',  
 )  
  
 class Meta:  
 ordering = ['due\_back']  
  
 def \_\_str\_\_(self):  
 """String for representing the Model object."""  
 return f'{self.id} ({self.book.title})'

We additionally declare a few new types of field:

* UUIDFieldis used for theidfield to set it as theprimary\_keyfor this model. This type of field allocates a globally unique value for each instance (one for every book you can find in the library).
* DateFieldis used for thedue\_backdate (at which the book is expected to become available after being borrowed or in maintenance). This value can beblankornull(needed for when the book is available). The model metadata (Class Meta) uses this field to order records when they are returned in a query.
* statusis aCharFieldthat defines a choice/selection list. As you can see, we define a tuple containing tuples of key-value pairs and pass it to the choices argument. The value in a key/value pair is a display value that a user can select, while the keys are the values that are actually saved if the option is selected. We've also set a default value of 'm' (maintenance) as books will initially be created unavailable before they are stocked on the shelves.

The method\_\_str\_\_()represents theBookInstanceobject using a combination of its unique id and the associatedBook's title.

**Note:**A little Python:

* Starting with Python 3.6, you can use the string interpolation syntax (also known as f-strings):f'{self.id} ({self.book.title})'.
* In older versions of this tutorial, we were using a[formatted string](https://peps.python.org/pep-3101/)syntax, which is also a valid way of formatting strings in Python (e.g.'{0} ({1})'.format(self.id,self.book.title)).

### [Author model](#author_model)

Copy theAuthormodel (shown below) underneath the existing code in**models.py**.

class Author(models.Model):  
 """Model representing an author."""  
 first\_name = models.CharField(max\_length=100)  
 last\_name = models.CharField(max\_length=100)  
 date\_of\_birth = models.DateField(null=True, blank=True)  
 date\_of\_death = models.DateField('Died', null=True, blank=True)  
  
 class Meta:  
 ordering = ['last\_name', 'first\_name']  
  
 def get\_absolute\_url(self):  
 """Returns the URL to access a particular author instance."""  
 return reverse('author-detail', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 """String for representing the Model object."""  
 return f'{self.last\_name}, {self.first\_name}'

All of the fields/methods should now be familiar. The model defines an author as having a first name, last name, and dates of birth and death (both optional). It specifies that by default the\_\_str\_\_()returns the name in*last name*,*firstname*order. Theget\_absolute\_url()method reverses theauthor-detailURL mapping to get the URL for displaying an individual author.

## [Re-run the database migrations](#re-run_the_database_migrations)

All your models have now been created. Now re-run your database migrations to add them to your database.

python3 manage.py makemigrations  
python3 manage.py migrate

## [Language model — challenge](#language_model_—_challenge)

Imagine a local benefactor donates a number of new books written in another language (say, Farsi). The challenge is to work out how these would be best represented in our library website, and then to add them to the models.

Some things to consider:

* Should "language" be associated with aBook,BookInstance, or some other object?
* Should the different languages be represented using model, a free text field, or a hard-coded selection list?

After you've decided, add the field. You can see what we decided on GitHub[here](https://github.com/mdn/django-locallibrary-tutorial/blob/master/catalog/models.py).

Don't forget that after a change to your model, you should again re-run your database migrations to add the changes.

python3 manage.py makemigrations  
python3 manage.py migrate

## [Summary](#summary)

In this article we've learned how models are defined, and then used this information to design and implement appropriate models for the*LocalLibrary*website.

At this point we'll divert briefly from creating the site, and check out the*Django Administration site*. This site will allow us to add some data to the library, which we can then display using our (yet to be created) views and templates.

## [See also](#see_also)

* [Writing your first Django app, part 2](https://docs.djangoproject.com/en/4.0/intro/tutorial02/)(Django docs)
* [Making queries](https://docs.djangoproject.com/en/4.0/topics/db/queries/)(Django Docs)
* [QuerySet API Reference](https://docs.djangoproject.com/en/4.0/ref/models/querysets/)(Django Docs)
* [Previous](/en-US/docs/Learn/Server-side/Django/skeleton_website)
* [Overview: Django](/en-US/docs/Learn/Server-side/Django)
* [Next](/en-US/docs/Learn/Server-side/Django/Admin_site)

## [In this module](#in_this_module)

* [Django introduction](/en-US/docs/Learn/Server-side/Django/Introduction)
* [Setting up a Django development environment](/en-US/docs/Learn/Server-side/Django/development_environment)
* [Django Tutorial: The Local Library website](/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website)
* [Django Tutorial Part 2: Creating a skeleton website](/en-US/docs/Learn/Server-side/Django/skeleton_website)
* **Django Tutorial Part 3: Using models**
* [Django Tutorial Part 4: Django admin site](/en-US/docs/Learn/Server-side/Django/Admin_site)
* [Django Tutorial Part 5: Creating our home page](/en-US/docs/Learn/Server-side/Django/Home_page)
* [Django Tutorial Part 6: Generic list and detail views](/en-US/docs/Learn/Server-side/Django/Generic_views)
* [Django Tutorial Part 7: Sessions framework](/en-US/docs/Learn/Server-side/Django/Sessions)
* [Django Tutorial Part 8: User authentication and permissions](/en-US/docs/Learn/Server-side/Django/Authentication)
* [Django Tutorial Part 9: Working with forms](/en-US/docs/Learn/Server-side/Django/Forms)
* [Django Tutorial Part 10: Testing a Django web application](/en-US/docs/Learn/Server-side/Django/Testing)
* [Django Tutorial Part 11: Deploying Django to production](/en-US/docs/Learn/Server-side/Django/Deployment)
* [Django web application security](/en-US/docs/Learn/Server-side/Django/web_application_security)
* [DIY Django mini blog](/en-US/docs/Learn/Server-side/Django/django_assessment_blog)

### Found a problem with this page?

* [Edit onGitHub](https://github.com/mdn/content/edit/main/files/en-us/learn/server-side/django/models/index.md)
* [Source onGitHub](https://github.com/mdn/content/blob/main/files/en-us/learn/server-side/django/models/index.md?plain=1)
* [Report a problem with this content onGitHub](https://github.com/mdn/content/issues/new?template=page-report.yml&mdn-url=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FLearn%2FServer-side%2FDjango%2FModels&metadata=%3C%21--+Do+not+make+changes+below+this+line+--%3E%0A%3Cdetails%3E%0A%3Csummary%3EPage+report+details%3C%2Fsummary%3E%0A%0A*+Folder%3A+%60en-us%2Flearn%2Fserver-side%2Fdjango%2Fmodels%60%0A*+MDN+URL%3A+https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FLearn%2FServer-side%2FDjango%2FModels%0A*+GitHub+URL%3A+https%3A%2F%2Fgithub.com%2Fmdn%2Fcontent%2Fblob%2Fmain%2Ffiles%2Fen-us%2Flearn%2Fserver-side%2Fdjango%2Fmodels%2Findex.md%0A*+Last+commit%3A+https%3A%2F%2Fgithub.com%2Fmdn%2Fcontent%2Fcommit%2F4d2655895e92601ca5292d5e4ebdf961055137f4%0A*+Document+last+modified%3A+2022-08-17T07%3A28%3A09.000Z%0A%0A%3C%2Fdetails%3E)
* Want to fix the problem yourself? See[our Contribution guide](https://github.com/mdn/content/blob/main/README.md).

**Last modified:**Aug 17, 2022,[by MDN contributors](/en-US/docs/Learn/Server-side/Django/Models/contributors.txt)[MDN logo](/)

Your blueprint for a better internet.

* [MDN on Twitter](https://twitter.com/mozdevnet)
* [MDN on GitHub](https://github.com/mdn/)

## MDN

* [About](/en-US/about)
* [Hacks Blog](https://hacks.mozilla.org/)
* [Careers](https://www.mozilla.org/en-US/careers/listings/?team=Marketing)

## Support

* [Product help](https://support.mozilla.org/products/mdn-plus)
* [Report a page issue](/en-US/docs/MDN/Contribute/Feedback#documentation_issues)
* [Report a site issue](/en-US/docs/MDN/Contribute/Feedback#site_issues)

## Our communities

* [MDN Community](/en-US/community)
* [MDN Forum](https://discourse.mozilla.org/c/mdn/236)
* [MDN Chat](https://wiki.mozilla.org/Matrix)

## Developers

* [Web Technologies](/en-US/docs/Web)
* [Learn Web Development](/en-US/docs/Learn)
* [MDN Plus](/en-US/plus)[Mozilla logo](https://www.mozilla.org/)
* [Website Privacy Notice](https://www.mozilla.org/privacy/websites/)
* [Cookies](https://www.mozilla.org/privacy/websites/#cookies)
* [Legal](https://www.mozilla.org/about/legal/terms/mozilla)
* [Community Participation Guidelines](https://www.mozilla.org/about/governance/policies/participation/)

Visit[Mozilla Corporation’s](https://www.mozilla.org)not-for-profit parent, the[Mozilla Foundation](https://foundation.mozilla.org/).  
Portions of this content are ©1998–2022by individual mozilla.org contributors. Content available under[a Creative Commons license](/docs/MDN/About#Copyrights_and_licenses).{"doc":{"isMarkdown":true,"isTranslated":false,"isActive":true,"flaws":{},"title":"Django Tutorial Part 3: Using models","mdn\_url":"/en-US/docs/Learn/Server-side/Django/Models","locale":"en-US","native":"English (US)","sidebarHTML":"\n\n<ol>\n <li data-default-state=\"\"><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web\"><strong>Complete beginners start here!</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Getting started with the web</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web\">Getting started with the web overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/Installing\_basic\_software\">Installing basic software</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/What\_will\_your\_website\_look\_like\">What will your website look like?</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/Dealing\_with\_files\">Dealing with files</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics\">HTML basics</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics\">CSS basics</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics\">JavaScript basics</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/Publishing\_your\_website\">Publishing your website</a></li>\n <li><a href=\"/en-US/docs/Learn/Getting\_started\_with\_the\_web/How\_the\_Web\_works\">How the web works</a></li>\n </ol>\n </details>\n </li>\n <li data-default-state=\"\"><a href=\"/en-US/docs/Learn/HTML\"><strong>HTML — Structuring the web</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Introduction to HTML</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML\">Introduction to HTML overview</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Getting\_started\">Getting started with HTML</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/The\_head\_metadata\_in\_HTML\">What's in the head? Metadata in HTML</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/HTML\_text\_fundamentals\">HTML text fundamentals</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Creating\_hyperlinks\">Creating hyperlinks</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Advanced\_text\_formatting\">Advanced text formatting</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Document\_and\_website\_structure\">Document and website structure</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Debugging\_HTML\">Debugging HTML</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Marking\_up\_a\_letter\">Assessment: Marking up a letter</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Structuring\_a\_page\_of\_content\">Assessment: Structuring a page of content</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Multimedia and embedding</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding\">Multimedia and embedding overview</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Images\_in\_HTML\">Images in HTML</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Video\_and\_audio\_content\">Video and audio content</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Other\_embedding\_technologies\">From object to iframe — other embedding technologies</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Adding\_vector\_graphics\_to\_the\_Web\">Adding vector graphics to the web</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Responsive\_images\">Responsive images</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Multimedia\_and\_embedding/Mozilla\_splash\_page\">Assessment: Mozilla splash page</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>HTML tables</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/HTML/Tables\">HTML tables overview</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Tables/Basics\">HTML table basics</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Tables/Advanced\">HTML table advanced features and accessibility</a></li>\n <li><a href=\"/en-US/docs/Learn/HTML/Tables/Structuring\_planet\_data\">Assessment: Structuring planet data</a></li>\n </ol>\n </details>\n </li>\n <li data-default-state=\"\"><a href=\"/en-US/docs/Learn/CSS\"><strong>CSS — Styling the web</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>CSS first steps</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps\">CSS first steps overview</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps/What\_is\_CSS\">What is CSS?</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps/Getting\_started\">Getting started with CSS</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps/How\_CSS\_is\_structured\">How CSS is structured</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps/How\_CSS\_works\">How CSS works</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/First\_steps/Styling\_a\_biography\_page\">Assessment: Styling a biography page</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>CSS building blocks</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks\">CSS building blocks overview</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Cascade\_and\_inheritance\">Cascade and inheritance</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Selectors\">CSS selectors</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/The\_box\_model\">The box model</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Backgrounds\_and\_borders\">Backgrounds and borders</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Handling\_different\_text\_directions\">Handling different text directions</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Overflowing\_content\">Overflowing content</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Values\_and\_units\">CSS values and units</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Sizing\_items\_in\_CSS\">Sizing items in CSS</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Images\_media\_form\_elements\">Images, media, and form elements</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Styling\_tables\">Styling tables</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Debugging\_CSS\">Debugging CSS</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Organizing\">Organizing your CSS</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Fundamental\_CSS\_comprehension\">Assessment: Fundamental CSS comprehension</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/Creating\_fancy\_letterheaded\_paper\">Assessment: Creating fancy letterheaded paper</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Building\_blocks/A\_cool\_looking\_box\">Assessment: A cool-looking box</a></li>\n </ol></details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Styling text</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text\">Styling text overview</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text/Fundamentals\">Fundamental text and font styling</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text/Styling\_lists\">Styling lists</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text/Styling\_links\">Styling links</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text/Web\_fonts\">Web fonts</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Styling\_text/Typesetting\_a\_homepage\">Assessment: Typesetting a community school homepage</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>CSS layout</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout\">CSS layout overview</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Introduction\">Introduction to CSS layout</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Normal\_Flow\">Normal Flow</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Flexbox\">Flexbox</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Grids\">Grids</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Floats\">Floats</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Positioning\">Positioning</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Multiple-column\_Layout\">Multiple-column layout</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Responsive\_Design\">Responsive design</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Media\_queries\">Beginner's guide to media queries</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Legacy\_Layout\_Methods\">Legacy layout methods</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Supporting\_Older\_Browsers\">Supporting older browsers</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/CSS\_layout/Fundamental\_Layout\_Comprehension\">Assessment: Fundamental layout comprehension</a></li>\n </ol>\n </details>\n </li>\n <li data-default-state=\"\"><a href=\"/en-US/docs/Learn/JavaScript\"><strong>JavaScript — Dynamic client-side scripting</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>JavaScript first steps</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps\">JavaScript first steps overview</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript\">What is JavaScript?</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/A\_first\_splash\">A first splash into JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/What\_went\_wrong\">What went wrong? Troubleshooting JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Variables\">Storing the information you need — Variables</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Math\">Basic math in JavaScript — Numbers and operators</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Strings\">Handling text — Strings in JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Useful\_string\_methods\">Useful string methods</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Arrays\">Arrays</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/First\_steps/Silly\_story\_generator\">Assessment: Silly story generator</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>JavaScript building blocks</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks\">JavaScript building blocks overview</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/conditionals\">Making decisions in your code — Conditionals</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Looping\_code\">Looping code</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Functions\">Functions — Reusable blocks of code</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Build\_your\_own\_function\">Build your own function</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Return\_values\">Function return values</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Events\">Introduction to events</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Building\_blocks/Image\_gallery\">Assessment: Image gallery</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Introducing JavaScript objects</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects\">Introducing JavaScript objects overview</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Basics\">Object basics</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Object\_prototypes\">Object prototypes</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Object-oriented\_programming\">Object-oriented programming concepts</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Classes\_in\_JavaScript\">Classes in JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/JSON\">Working with JSON data</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Object\_building\_practice\">Object building practice</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Objects/Adding\_bouncing\_balls\_features\">Assessment: Adding features to our bouncing balls demo</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Asynchronous JavaScript</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous\">Asynchronous JavaScript overview</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous/Introducing\">Introducing asynchronous JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous/Promises\">How to use promises</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous/Implementing\_a\_promise-based\_API\">Implementing a promise-based API</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous/Introducing\_workers\">Introducing workers</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Asynchronous/Sequencing\_animations\">Assessment: sequencing animations</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Client-side web APIs</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs\">Client-side web APIs</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Introduction\">Introduction to web APIs</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Manipulating\_documents\">Manipulating documents</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Fetching\_data\">Fetching data from the server</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Third\_party\_APIs\">Third party APIs</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Drawing\_graphics\">Drawing graphics</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Video\_and\_audio\_APIs\">Video and audio APIs</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Client-side\_storage\">Client-side storage</a></li>\n </ol>\n </details>\n </li>\n <li><a href=\"/en-US/docs/Learn/Forms\"><strong>Web forms — Working with user data</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Core forms learning pathway</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Forms\">Web forms overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Your\_first\_form\">Your first form</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/How\_to\_structure\_a\_web\_form\">How to structure a web form</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Basic\_native\_form\_controls\">Basic native form controls</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/HTML5\_input\_types\">The HTML5 input types</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Other\_form\_controls\">Other form controls</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Styling\_web\_forms\">Styling web forms</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Advanced\_form\_styling\">Advanced form styling</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/UI\_pseudo-classes\">UI pseudo-classes</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Form\_validation\">Client-side form validation</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Sending\_and\_retrieving\_form\_data\">Sending form data</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Advanced forms articles</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Forms/How\_to\_build\_custom\_form\_controls\">How to build custom form controls</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Sending\_forms\_through\_JavaScript\">Sending forms through JavaScript</a></li>\n <li><a href=\"/en-US/docs/Learn/Forms/Property\_compatibility\_table\_for\_form\_controls\">CSS property compatibility table for form controls</a></li>\n </ol>\n </details>\n </li>\n <li><a href=\"/en-US/docs/Learn/Accessibility\"><strong>Accessibility — Make the web usable by everyone</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Accessibility guides</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Accessibility\">Accessibility overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/What\_is\_accessibility\">What is accessibility?</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/HTML\">HTML: A good basis for accessibility</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/CSS\_and\_JavaScript\">CSS and JavaScript accessibility best practices</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/WAI-ARIA\_basics\">WAI-ARIA basics</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/Multimedia\">Accessible multimedia</a></li>\n <li><a href=\"/en-US/docs/Learn/Accessibility/Mobile\">Mobile accessibility</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Accessibility assessment</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Accessibility/Accessibility\_troubleshooting\">Assessment: Accessibility troubleshooting</a></li>\n </ol>\n </details>\n </li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing\"><strong>Tools and testing</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Client-side web development tools</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools\">Client-side web development tools index</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools/Overview\">Client-side tooling overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools/Command\_line\">Command line crash course</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools/Package\_management\">Package management basics</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools/Introducing\_complete\_toolchain\">Introducing a complete toolchain</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Understanding\_client-side\_tools/Deployment\">Deploying our app</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Introduction to client-side frameworks</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Introduction\">Client-side frameworks overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Main\_features\">Framework main features</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>React</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_getting\_started\">Getting started with React</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_todo\_list\_beginning\">Beginning our React todo list</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_components\">Componentizing our React app</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_interactivity\_events\_state\">React interactivity: Events and state</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_interactivity\_filtering\_conditional\_rendering\">React interactivity: Editing, filtering, conditional rendering</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_accessibility\">Accessibility in React</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_resources\">React resources</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Ember</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_getting\_started\">Getting started with Ember</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_structure\_componentization\">Ember app structure and componentization</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_interactivity\_events\_state\">Ember interactivity: Events, classes and state</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_conditional\_footer\">Ember Interactivity: Footer functionality, conditional rendering</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_routing\">Routing in Ember</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Ember\_resources\">Ember resources and troubleshooting</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Vue</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_getting\_started\">Getting started with Vue</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_first\_component\">Creating our first Vue component</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_rendering\_lists\">Rendering a list of Vue components</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_methods\_events\_models\">Adding a new todo form: Vue events, methods, and models</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_styling\">Styling Vue components with CSS</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_computed\_properties\">Using Vue computed properties</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_conditional\_rendering \">Vue conditional rendering: editing existing todos</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_refs\_focus\_management\">Focus management with Vue refs</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Vue\_resources\">Vue resources</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Svelte</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_getting\_started\">Getting started with Svelte</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_Todo\_list\_beginning\">Starting our Svelte Todo list app</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_variables\_props\">Dynamic behavior in Svelte: working with variables and props</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_components\">Componentizing our Svelte app</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_reactivity\_lifecycle\_accessibility\">Advanced Svelte: Reactivity, lifecycle, accessibility</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_stores\">Working with Svelte stores</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_TypeScript \">TypeScript support in Svelte</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Svelte\_deployment\_next\">Deployment and next steps</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Angular</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_getting\_started\">Getting started with Angular</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_todo\_list\_beginning\">Beginning our Angular todo list app</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_styling\">Styling our Angular app</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_item\_component\">Creating an item component</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_filtering\">Filtering our to-do items</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/Angular\_building\">Building Angular applications and further resources</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Git and GitHub</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/GitHub\">Git and GitHub overview</a></li>\n <li><a href=\"https://guides.github.com/activities/hello-world/\">Hello World</a></li>\n <li><a href=\"https://guides.github.com/introduction/git-handbook/\">Git Handbook</a></li>\n <li><a href=\"https://guides.github.com/activities/forking/\">Forking Projects</a></li>\n <li><a href=\"https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests\">About pull requests</a></li>\n <li><a href=\"https://guides.github.com/features/issues/\">Mastering Issues</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Cross browser testing</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing\">Cross browser testing overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Introduction\">Introduction to cross browser testing</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Testing\_strategies\">Strategies for carrying out testing</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/HTML\_and\_CSS\">Handling common HTML and CSS problems</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/JavaScript\">Handling common JavaScript problems</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Accessibility\">Handling common accessibility problems</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Feature\_detection\">Implementing feature detection</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Automated\_testing\">Introduction to automated testing</a></li>\n <li><a href=\"/en-US/docs/Learn/Tools\_and\_testing/Cross\_browser\_testing/Your\_own\_automation\_environment\">Setting up your own test automation environment</a></li>\n </ol>\n </details>\n </li>\n <li data-default-state=\"open\"><a href=\"/en-US/docs/Learn/Server-side\"><strong>Server-side website programming</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>First steps</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Server-side/First\_steps\">First steps overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/First\_steps/Introduction\">Introduction to the server-side</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/First\_steps/Client-Server\_overview\">Client-Server overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/First\_steps/Web\_frameworks\">Server-side web frameworks</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/First\_steps/Website\_security\">Website security</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details open=\"\">\n <summary>Django web framework (Python)</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django\">Django web framework (Python) overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Introduction\">Introduction</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/development\_environment\">Setting up a development environment</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Tutorial\_local\_library\_website\">Tutorial: The Local Library website</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/skeleton\_website\">Tutorial Part 2: Creating a skeleton website</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Models\">Tutorial Part 3: Using models</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Admin\_site\">Tutorial Part 4: Django admin site</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Home\_page\">Tutorial Part 5: Creating our home page</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Generic\_views\">Tutorial Part 6: Generic list and detail views</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Sessions\">Tutorial Part 7: Sessions framework</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Authentication\">Tutorial Part 8: User authentication and permissions</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Forms\">Tutorial Part 9: Working with forms</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Testing\">Tutorial Part 10: Testing a Django web application</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Deployment\">Tutorial Part 11: Deploying Django to production</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/web\_application\_security\">Web application security</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/django\_assessment\_blog\">Assessment: DIY mini blog</a></li>\n </ol>\n </details>\n </li>\n <li class=\"toggle\">\n <details>\n <summary>Express Web Framework (node.js/JavaScript)</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs\">Express Web Framework (Node.js/JavaScript) overview</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/Introduction\">Express/Node introduction</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/development\_environment\">Setting up a Node (Express) development environment</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/Tutorial\_local\_library\_website\">Express tutorial: The Local Library website</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/skeleton\_website\">Express Tutorial Part 2: Creating a skeleton website</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/mongoose\">Express Tutorial Part 3: Using a database (with Mongoose)</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/routes\">Express Tutorial Part 4: Routes and controllers</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/Displaying\_data\">Express Tutorial Part 5: Displaying library data</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/forms\">Express Tutorial Part 6: Working with forms</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Express\_Nodejs/deployment\">Express Tutorial Part 7: Deploying to production</a></li>\n </ol>\n </details>\n </li>\n <li><a href=\"#\"><strong>Further resources</strong></a></li>\n <li class=\"toggle\">\n <details>\n <summary>Common questions</summary>\n <ol>\n <li><a href=\"/en-US/docs/Learn/HTML/Howto\">HTML questions</a></li>\n <li><a href=\"/en-US/docs/Learn/CSS/Howto\">CSS questions</a></li>\n <li><a href=\"/en-US/docs/Learn/JavaScript/Howto\">JavaScript questions</a></li>\n <li><a href=\"/en-US/docs/Learn/Common\_questions#Web\_mechanics\">Web mechanics</a></li>\n <li><a href=\"/en-US/docs/Learn/Common\_questions#Tools\_and\_setup\">Tools and setup</a></li>\n <li><a href=\"/en-US/docs/Learn/Common\_questions#Design\_and\_accessibility\">Design and accessibility</a></li>\n </ol>\n </details>\n </li>\n</ol>\n\n","body":[{"type":"prose","value":{"id":null,"title":null,"isH3":false,"content":"<ul class=\"prev-next\">\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django/skeleton\_website\"><span class=\"button-wrap\"> Previous </span></a></li>\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django\"><span class=\"button-wrap\"> Overview: Django</span></a></li>\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django/Admin\_site\"><span class=\"button-wrap\"> Next </span></a></li>\n</ul>\n<p>This article shows how to define models for the LocalLibrary website. It explains what a model is, how it is declared, and some of the main field types. It also briefly shows a few of the main ways you can access model data.</p>\n<div class=\"table-scroll\"><table>\n <tbody>\n <tr>\n <th scope=\"row\">Prerequisites:</th>\n <td><a href=\"/en-US/docs/Learn/Server-side/Django/skeleton\_website\">Django Tutorial Part 2: Creating a skeleton website</a>.</td>\n </tr>\n <tr>\n <th scope=\"row\">Objective:</th>\n <td>\n <p>To be able to design and create your own models, choosing fields appropriately.</p>\n </td>\n </tr>\n </tbody>\n</table></div>"}},{"type":"prose","value":{"id":"overview","title":"Overview","isH3":false,"content":"<p>Django web applications access and manage data through Python objects referred to as models. Models define the <em>structure</em> of stored data, including the field <em>types</em> and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. The definition of the model is independent of the underlying database — you can choose one of several as part of your project settings. Once you've chosen what database you want to use, you don't need to talk to it directly at all — you just write your model structure and other code, and Django handles all the dirty work of communicating with the database for you.</p>\n<p>This tutorial shows how to define and access the models for the <a href=\"/en-US/docs/Learn/Server-side/Django/Tutorial\_local\_library\_website\">LocalLibrary website</a> example.</p>"}},{"type":"prose","value":{"id":"designing\_the\_locallibrary\_models","title":"Designing the LocalLibrary models","isH3":false,"content":"<p>Before you jump in and start coding the models, it's worth taking a few minutes to think about what data we need to store and the relationships between the different objects.</p>\n<p>We know that we need to store information about books (title, summary, author, written language, category, ISBN) and that we might have multiple copies available (with globally unique id, availability status, etc.). We might need to store more information about the author than just their name, and there might be multiple authors with the same or similar names. We want to be able to sort information based on book title, author, written language, and category.</p>\n<p>When designing your models, it makes sense to have separate models for every \"object\" (a group of related information). In this case, the obvious objects are books, book instances, and authors.</p>\n<p>You might also want to use models to represent selection-list options (e.g. like a drop down list of choices), rather than hard coding the choices into the website itself — this is recommended when all the options aren't known up front or may change. Obvious candidates for models, in this case, include the book genre (e.g. Science Fiction, French Poetry, etc.) and language (English, French, Japanese).</p>\n<p>Once we've decided on our models and field, we need to think about the relationships. Django allows you to define relationships that are one to one (<code>OneToOneField</code>), one to many (<code>ForeignKey</code>) and many to many (<code>ManyToManyField</code>).</p>\n<p>With that in mind, the UML association diagram below shows the models we'll define in this case (as boxes).</p>\n<p>\n <img src=\"/en-US/docs/Learn/Server-side/Django/Models/local\_library\_model\_uml.svg\" alt=\"LocalLibrary Model UML with fixed Author multiplicity inside the Book class\" width=\"1140\" height=\"724\" loading=\"lazy\">\n</p>\n<p>We've created models for the book (the generic details of the book), book instance (status of specific physical copies of the book available in the system), and author. We have also decided to have a model for the genre so that values can be created/selected through the admin interface. We've decided not to have a model for the <code>BookInstance:status</code> — we've hardcoded the values (<code>LOAN\_STATUS</code>) because we don't expect these to change. Within each of the boxes, you can see the model name, the field names, and types, and also the methods and their return types.</p>\n<p>The diagram also shows the relationships between the models, including their <em>multiplicities</em>. The multiplicities are the numbers on the diagram showing the numbers (maximum and minimum) of each model that may be present in the relationship. For example, the connecting line between the boxes shows that Book and a Genre are related. The numbers close to the Genre model show that a book must have one or more Genres (as many as you like), while the numbers on the other end of the line next to the Book model show that a Genre can have zero or many associated books.</p>\n<div class=\"notecard note\" id=\"sect1\">\n <p><strong>Note:</strong> The next section provides a basic primer explaining how models are defined and used. As you read it, consider how we will construct each of the models in the diagram above.</p>\n</div>"}},{"type":"prose","value":{"id":"model\_primer","title":"Model primer","isH3":false,"content":"<p>This section provides a brief overview of how a model is defined and some of the more important fields and field arguments.</p>"}},{"type":"prose","value":{"id":"model\_definition","title":"Model definition","isH3":true,"content":"<p>Models are usually defined in an application's <strong>models.py</strong> file. They are implemented as subclasses of <code>django.db.models.Model</code>, and can include fields, methods and metadata. The code fragment below shows a \"typical\" model, named <code>MyModelName</code>:</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">from</span> django<span class=\"token punctuation\">.</span>db <span class=\"token keyword\">import</span> models\n<span class=\"token keyword\">from</span> django<span class=\"token punctuation\">.</span>urls <span class=\"token keyword\">import</span> reverse\n\n<span class=\"token keyword\">class</span> <span class=\"token class-name\">MyModelName</span><span class=\"token punctuation\">(</span>models<span class=\"token punctuation\">.</span>Model<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"A typical class defining a model, derived from the Model class.\"\"\"</span>\n\n <span class=\"token comment\"># Fields</span>\n my\_field\_name <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">20</span><span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Enter field documentation'</span><span class=\"token punctuation\">)</span>\n <span class=\"token comment\"># …</span>\n\n <span class=\"token comment\"># Metadata</span>\n <span class=\"token keyword\">class</span> <span class=\"token class-name\">Meta</span><span class=\"token punctuation\">:</span>\n ordering <span class=\"token operator\">=</span> <span class=\"token punctuation\">[</span><span class=\"token string\">'-my\_field\_name'</span><span class=\"token punctuation\">]</span>\n\n <span class=\"token comment\"># Methods</span>\n <span class=\"token keyword\">def</span> <span class=\"token function\">get\_absolute\_url</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Returns the URL to access a particular instance of MyModelName.\"\"\"</span>\n <span class=\"token keyword\">return</span> reverse<span class=\"token punctuation\">(</span><span class=\"token string\">'model-detail-view'</span><span class=\"token punctuation\">,</span> args<span class=\"token operator\">=</span><span class=\"token punctuation\">[</span><span class=\"token builtin\">str</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">]</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"String for representing the MyModelName object (in Admin site etc.).\"\"\"</span>\n <span class=\"token keyword\">return</span> self<span class=\"token punctuation\">.</span>my\_field\_name\n</code></pre></div>\n<p>In the below sections we'll explore each of the features inside the model in detail:</p>\n<h4 id=\"fields\">Fields</h4>\n<p>A model can have an arbitrary number of fields, of any type — each one represents a column of data that we want to store in one of our database tables. Each database record (row) will consist of one of each field value. Let's look at the example seen below:</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code>my\_field\_name <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">20</span><span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Enter field documentation'</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<p>Our above example has a single field called <code>my\_field\_name</code>, of type <code>models.CharField</code> — which means that this field will contain strings of alphanumeric characters. The field types are assigned using specific classes, which determine the type of record that is used to store the data in the database, along with validation criteria to be used when values are received from an HTML form (i.e. what constitutes a valid value). The field types can also take arguments that further specify how the field is stored or can be used. In this case we are giving our field two arguments:</p>\n<ul>\n <li><code>max\_length=20</code> — States that the maximum length of a value in this field is 20 characters.</li>\n <li><code>help\_text='Enter field documentation'</code> — helpful text that may be displayed in a form to help users understand how the field is used.</li>\n</ul>\n<p>\n The field name is used to refer to it in queries and templates.\n Fields also have a label, which is specified using the <code>verbose\_name</code> argument (with a default value of <code>None</code>).\n If <code>verbose\_name</code> is not set, the label is created from the field name by replacing any underscores with a space, and capitalizing the first letter (for example, the field <code>my\_field\_name</code> would have a default label of <em>My field name</em> when used in forms).\n</p>\n<p>The order that fields are declared will affect their default order if a model is rendered in a form (e.g. in the Admin site), though this may be overridden.</p>\n<h5 id=\"common\_field\_arguments\">Common field arguments</h5>\n<p>The following common arguments can be used when declaring many/most of the different field types:</p>\n<ul>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#help-text\" class=\"external\" rel=\" noopener\">help\_text</a>: Provides a text label for HTML forms (e.g. in the admin site), as described above.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#verbose-name\" class=\"external\" rel=\" noopener\">verbose\_name</a>: A human-readable name for the field used in field labels. If not specified, Django will infer the default verbose name from the field name.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#default\" class=\"external\" rel=\" noopener\">default</a>: The default value for the field. This can be a value or a callable object, in which case the object will be called every time a new record is created.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#null\" class=\"external\" rel=\" noopener\">null</a>: If <code>True</code>, Django will store blank values as <code>NULL</code> in the database for fields where this is appropriate (a <code>CharField</code> will instead store an empty string). The default is <code>False</code>.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#blank\" class=\"external\" rel=\" noopener\">blank</a>: If <code>True</code>, the field is allowed to be blank in your forms. The default is <code>False</code>, which means that Django's form validation will force you to enter a value. This is often used with <code>null=True</code> , because if you're going to allow blank values, you also want the database to be able to represent them appropriately.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#choices\" class=\"external\" rel=\" noopener\">choices</a>: A group of choices for this field. If this is provided, the default corresponding form widget will be a select box with these choices instead of the standard text field.</li>\n <li>\n <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#primary-key\" class=\"external\" rel=\" noopener\">primary\_key</a>:\n If <code>True</code>, sets the current field as the primary key for the model (A primary key is a special database column designated to uniquely identify all the different table records).\n If no field is specified as the primary key, Django will automatically add a field for this purpose.\n The type of auto-created primary key fields can be specified for each app in <a href=\"https://docs.djangoproject.com/en/4.0/ref/applications/#django.apps.AppConfig.default\_auto\_field\" class=\"external\" rel=\" noopener\"><code>AppConfig.default\_auto\_field</code></a> or globally in the <a href=\"https://docs.djangoproject.com/en/4.0/ref/settings/#std:setting-DEFAULT\_AUTO\_FIELD\" class=\"external\" rel=\" noopener\"><code>DEFAULT\_AUTO\_FIELD</code></a> setting.\n <div class=\"notecard note\" id=\"sect2\">\n <p>\n <strong>Note:</strong> Apps created using <strong>manage.py</strong> set the type of the primary key to a <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#bigautofield\" class=\"external\" rel=\" noopener\">BigAutoField</a>.\n You can see this in the local library <strong>catalog/apps.py</strong> file:\n </p>\n <div class=\"code-example\"><pre class=\"brush: py notranslate\"><code><span class=\"token keyword\">class</span> <span class=\"token class-name\">CatalogConfig</span><span class=\"token punctuation\">(</span>AppConfig<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n default\_auto\_field <span class=\"token operator\">=</span> <span class=\"token string\">'django.db.models.BigAutoField'</span>\n</code></pre></div>\n </div>\n </li>\n</ul>\n<p>There are many other options — you can view the <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-options\" class=\"external\" rel=\" noopener\">full list of field options here</a>.</p>\n<h5 id=\"common\_field\_types\">Common field types</h5>\n<p>The following list describes some of the more commonly used types of fields.</p>\n<ul>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.CharField\" class=\"external\" rel=\" noopener\">CharField</a> is used to define short-to-mid sized fixed-length strings. You must specify the <code>max\_length</code> of the data to be stored.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.TextField\" class=\"external\" rel=\" noopener\">TextField</a> is used for large arbitrary-length strings. You may specify a <code>max\_length</code> for the field, but this is used only when the field is displayed in forms (it is not enforced at the database level).</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.IntegerField\" class=\"external\" rel=\" noopener\">IntegerField</a> is a field for storing integer (whole number) values, and for validating entered values as integers in forms.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#datefield\" class=\"external\" rel=\" noopener\">DateField</a> and <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#datetimefield\" class=\"external\" rel=\" noopener\">DateTimeField</a> are used for storing/representing dates and date/time information (as Python <code>datetime.date</code> and <code>datetime.datetime</code> objects, respectively). These fields can additionally declare the (mutually exclusive) parameters <code>auto\_now=True</code> (to set the field to the current date every time the model is saved), <code>auto\_now\_add</code> (to only set the date when the model is first created) , and <code>default</code> (to set a default date that can be overridden by the user).</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#emailfield\" class=\"external\" rel=\" noopener\">EmailField</a> is used to store and validate email addresses.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#filefield\" class=\"external\" rel=\" noopener\">FileField</a> and <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#imagefield\" class=\"external\" rel=\" noopener\">ImageField</a> are used to upload files and images respectively (the <code>ImageField</code> adds additional validation that the uploaded file is an image). These have parameters to define how and where the uploaded files are stored.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#autofield\" class=\"external\" rel=\" noopener\">AutoField</a> is a special type of <code>IntegerField</code> that automatically increments. A primary key of this type is automatically added to your model if you don't explicitly specify one.</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#foreignkey\" class=\"external\" rel=\" noopener\">ForeignKey</a> is used to specify a one-to-many relationship to another database model (e.g. a car has one manufacturer, but a manufacturer can make many cars). The \"one\" side of the relationship is the model that contains the \"key\" (models containing a \"foreign key\" referring to that \"key\", are on the \"many\" side of such a relationship).</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#manytomanyfield\" class=\"external\" rel=\" noopener\">ManyToManyField</a> is used to specify a many-to-many relationship (e.g. a book can have several genres, and each genre can contain several books). In our library app we will use these very similarly to <code>ForeignKeys</code>, but they can be used in more complicated ways to describe the relationships between groups. These have the parameter <code>on\_delete</code> to define what happens when the associated record is deleted (e.g. a value of <code>models.SET\_NULL</code> would set the value to <code>NULL</code>).</li>\n</ul>\n<p>There are many other types of fields, including fields for different types of numbers (big integers, small integers, floats), booleans, URLs, slugs, unique ids, and other \"time-related\" information (duration, time, etc.). You can view the <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-types\" class=\"external\" rel=\" noopener\">full list here</a>.</p>\n<h4 id=\"metadata\">Metadata</h4>\n<p>You can declare model-level metadata for your Model by declaring <code>class Meta</code>, as shown.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">class</span> <span class=\"token class-name\">Meta</span><span class=\"token punctuation\">:</span>\n ordering <span class=\"token operator\">=</span> <span class=\"token punctuation\">[</span><span class=\"token string\">'-my\_field\_name'</span><span class=\"token punctuation\">]</span>\n</code></pre></div>\n<p>One of the most useful features of this metadata is to control the <em>default ordering</em> of records returned when you query the model type. You do this by specifying the match order in a list of field names to the <code>ordering</code> attribute, as shown above. The ordering will depend on the type of field (character fields are sorted alphabetically, while date fields are sorted in chronological order). As shown above, you can prefix the field name with a minus symbol (-) to reverse the sorting order.</p>\n<p>So as an example, if we chose to sort books like this by default:</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code>ordering <span class=\"token operator\">=</span> <span class=\"token punctuation\">[</span><span class=\"token string\">'title'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'-pubdate'</span><span class=\"token punctuation\">]</span>\n</code></pre></div>\n<p>the books would be sorted alphabetically by title, from A-Z, and then by publication date inside each title, from newest to oldest.</p>\n<p>Another common attribute is <code>verbose\_name</code>, a verbose name for the class in singular and plural form:</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code>verbose\_name <span class=\"token operator\">=</span> <span class=\"token string\">'BetterName'</span>\n</code></pre></div>\n<p>Other useful attributes allow you to create and apply new \"access permissions\" for the model (default permissions are applied automatically), allow ordering based on another field, or to declare that the class is \"abstract\" (a base class that you cannot create records for, and will instead be derived from to create other models).</p>\n<p>Many of the other metadata options control what database must be used for the model and how the data is stored (these are really only useful if you need to map a model to an existing database).</p>\n<p>The full list of metadata options are available here: <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/options/\" class=\"external\" rel=\" noopener\">Model metadata options</a> (Django docs).</p>\n<h4 id=\"methods\">Methods</h4>\n<p>A model can also have methods.</p>\n<p><strong>Minimally, in every model you should define the standard Python class method <code>\_\_str\_\_()</code> to return a human-readable string for each object.</strong> This string is used to represent individual records in the administration site (and anywhere else you need to refer to a model instance). Often this will return a title or name field from the model.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token keyword\">return</span> self<span class=\"token punctuation\">.</span>field\_name\n</code></pre></div>\n<p>Another common method to include in Django models is <code>get\_absolute\_url()</code>, which returns a URL for displaying individual model records on the website (if you define this method then Django will automatically add a \"View on Site\" button to the model's record editing screens in the Admin site). A typical pattern for <code>get\_absolute\_url()</code> is shown below.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">def</span> <span class=\"token function\">get\_absolute\_url</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Returns the URL to access a particular instance of the model.\"\"\"</span>\n <span class=\"token keyword\">return</span> reverse<span class=\"token punctuation\">(</span><span class=\"token string\">'model-detail-view'</span><span class=\"token punctuation\">,</span> args<span class=\"token operator\">=</span><span class=\"token punctuation\">[</span><span class=\"token builtin\">str</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">]</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<div class=\"notecard note\" id=\"sect3\">\n <p><strong>Note:</strong> Assuming you will use URLs like <code>/myapplication/mymodelname/2</code> to display individual records for your model (where \"2\" is the <code>id</code> for a particular record), you will need to create a URL mapper to pass the response and id to a \"model detail view\" (which will do the work required to display the record). The <code>reverse()</code> function above is able to \"reverse\" your URL mapper (in the above case named <em>'model-detail-view'</em>) in order to create a URL of the right format.</p>\n <p>Of course to make this work you still have to write the URL mapping, view, and template!</p>\n</div>\n<p>You can also define any other methods you like, and call them from your code or templates (provided that they don't take any parameters).</p>"}},{"type":"prose","value":{"id":"model\_management","title":"Model management","isH3":true,"content":"<p>Once you've defined your model classes you can use them to create, update, or delete records, and to run queries to get all records or particular subsets of records. We'll show you how to do that in the tutorial when we define our views, but here is a brief summary.</p>\n<h4 id=\"creating\_and\_modifying\_records\">Creating and modifying records</h4>\n<p>To create a record you can define an instance of the model and then call <code>save()</code>.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token comment\"># Create a new record using the model's constructor.</span>\nrecord <span class=\"token operator\">=</span> MyModelName<span class=\"token punctuation\">(</span>my\_field\_name<span class=\"token operator\">=</span><span class=\"token string\">\"Instance #1\"</span><span class=\"token punctuation\">)</span>\n\n<span class=\"token comment\"># Save the object into the database.</span>\nrecord<span class=\"token punctuation\">.</span>save<span class=\"token punctuation\">(</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<div class=\"notecard note\" id=\"sect4\">\n <p><strong>Note:</strong> If you haven't declared any field as a <code>primary\_key</code>, the new record will be given one automatically, with the field name <code>id</code>. You could query this field after saving the above record, and it would have a value of 1.</p>\n</div>\n<p>You can access the fields in this new record using the dot syntax, and change the values. You have to call <code>save()</code> to store modified values to the database.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token comment\"># Access model field values using Python attributes.</span>\n<span class=\"token keyword\">print</span><span class=\"token punctuation\">(</span>record<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">)</span> <span class=\"token comment\"># should return 1 for the first record.</span>\n<span class=\"token keyword\">print</span><span class=\"token punctuation\">(</span>record<span class=\"token punctuation\">.</span>my\_field\_name<span class=\"token punctuation\">)</span> <span class=\"token comment\"># should print 'Instance #1'</span>\n\n<span class=\"token comment\"># Change record by modifying the fields, then calling save().</span>\nrecord<span class=\"token punctuation\">.</span>my\_field\_name <span class=\"token operator\">=</span> <span class=\"token string\">\"New Instance Name\"</span>\nrecord<span class=\"token punctuation\">.</span>save<span class=\"token punctuation\">(</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<h4 id=\"searching\_for\_records\">Searching for records</h4>\n<p>You can search for records that match certain criteria using the model's <code>objects</code> attribute (provided by the base class).</p>\n<div class=\"notecard note\" id=\"sect5\">\n <p><strong>Note:</strong> Explaining how to search for records using \"abstract\" model and field names can be a little confusing. In the discussion below, we'll refer to a <code>Book</code> model with <code>title</code> and <code>genre</code> fields, where genre is also a model with a single field <code>name</code>.</p>\n</div>\n<p>We can get all records for a model as a <code>QuerySet</code>, using <code>objects.all()</code>. The <code>QuerySet</code> is an iterable object, meaning that it contains a number of objects that we can iterate/loop through.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code>all\_books <span class=\"token operator\">=</span> Book<span class=\"token punctuation\">.</span>objects<span class=\"token punctuation\">.</span><span class=\"token builtin\">all</span><span class=\"token punctuation\">(</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<p>Django's <code>filter()</code> method allows us to filter the returned <code>QuerySet</code> to match a specified <strong>text</strong> or <strong>numeric</strong> field against particular criteria. For example, to filter for books that contain \"wild\" in the title and then count them, we could do the following.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code>wild\_books <span class=\"token operator\">=</span> Book<span class=\"token punctuation\">.</span>objects<span class=\"token punctuation\">.</span><span class=\"token builtin\">filter</span><span class=\"token punctuation\">(</span>title\_\_contains<span class=\"token operator\">=</span><span class=\"token string\">'wild'</span><span class=\"token punctuation\">)</span>\nnumber\_wild\_books <span class=\"token operator\">=</span> wild\_books<span class=\"token punctuation\">.</span>count<span class=\"token punctuation\">(</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<p>The fields to match and the type of match are defined in the filter parameter name, using the format: <code>field\_name\_\_match\_type</code> (note the <em>double underscore</em> between <code>title</code> and <code>contains</code> above). Above we're filtering <code>title</code> with a case-sensitive match. There are many other types of matches you can do: <code>icontains</code> (case insensitive), <code>iexact</code> (case-insensitive exact match), <code>exact</code> (case-sensitive exact match) and <code>in</code>, <code>gt</code> (greater than), <code>startswith</code>, etc. The <a href=\"https://docs.djangoproject.com/en/4.0/ref/models/querysets/#field-lookups\" class=\"external\" rel=\" noopener\">full list is here</a>.</p>\n<p>\n In some cases, you'll need to filter on a field that defines a one-to-many relationship to another model (e.g. a <code>ForeignKey</code>). In this case, you can \"index\" to fields within the related model with additional double underscores.\n So for example to filter for books with a specific genre pattern, you will have to index to the <code>name</code> through the <code>genre</code> field, as shown below:\n</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token comment\"># Will match on: Fiction, Science fiction, non-fiction etc.</span>\nbooks\_containing\_genre <span class=\"token operator\">=</span> Book<span class=\"token punctuation\">.</span>objects<span class=\"token punctuation\">.</span><span class=\"token builtin\">filter</span><span class=\"token punctuation\">(</span>genre\_\_name\_\_icontains<span class=\"token operator\">=</span><span class=\"token string\">'fiction'</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<div class=\"notecard note\" id=\"sect6\">\n <p>\n <strong>Note:</strong> You can use underscores (<code>\_\_</code>) to navigate as many levels of relationships (<code>ForeignKey</code>/<code>ManyToManyField</code>) as you like.\n For example, a <code>Book</code> that had different types, defined using a further \"cover\" relationship might have a parameter name: <code>type\_\_cover\_\_name\_\_exact='hard'.</code>\n </p>\n</div>\n<p>There is a lot more you can do with queries, including backwards searches from related models, chaining filters, returning a smaller set of values, etc. For more information, see <a href=\"https://docs.djangoproject.com/en/4.0/topics/db/queries/\" class=\"external\" rel=\" noopener\">Making queries</a> (Django Docs).</p>"}},{"type":"prose","value":{"id":"defining\_the\_locallibrary\_models","title":"Defining the LocalLibrary Models","isH3":false,"content":"<p>In this section we will start defining the models for the library. Open <code>models.py</code> (in /locallibrary/catalog/). The boilerplate at the top of the page imports the <em>models</em> module, which contains the model base class <code>models.Model</code> that our models will inherit from.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">from</span> django<span class=\"token punctuation\">.</span>db <span class=\"token keyword\">import</span> models\n\n<span class=\"token comment\"># Create your models here.</span>\n</code></pre></div>"}},{"type":"prose","value":{"id":"genre\_model","title":"Genre model","isH3":true,"content":"<p>\n Copy the <code>Genre</code> model code shown below and paste it into the bottom of your <code>models.py</code> file. This model is used to store information about the book category — for example whether it is fiction or non-fiction, romance or military history, etc.\n As mentioned above, we've created the genre as a model rather than as free text or a selection list so that the possible values can be managed through the database rather than being hard coded.\n</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">class</span> <span class=\"token class-name\">Genre</span><span class=\"token punctuation\">(</span>models<span class=\"token punctuation\">.</span>Model<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Model representing a book genre.\"\"\"</span>\n name <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">200</span><span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Enter a book genre (e.g. Science Fiction)'</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"String for representing the Model object.\"\"\"</span>\n <span class=\"token keyword\">return</span> self<span class=\"token punctuation\">.</span>name\n</code></pre></div>\n<p>The model has a single <code>CharField</code> field (<code>name</code>), which is used to describe the genre (this is limited to 200 characters and has some <code>help\_text</code>). At the end of the model, we declare a <code>\_\_str\_\_()</code> method, which returns the name of the genre defined by a particular record. No verbose name has been defined, so the field will be called <code>Name</code> in forms.</p>"}},{"type":"prose","value":{"id":"book\_model","title":"Book model","isH3":true,"content":"<p>Copy the <code>Book</code> model below and again paste it into the bottom of your file. The <code>Book</code> model represents all information about an available book in a general sense, but not a particular physical \"instance\" or \"copy\" available for loan. The model uses a <code>CharField</code> to represent the book's <code>title</code> and <code>isbn</code>. For <code>isbn</code>, note how the first unnamed parameter explicitly sets the label as \"ISBN\" (otherwise, it would default to \"Isbn\"). We also set the parameter <code>unique</code> as <code>true</code> to ensure all books have a unique ISBN (the unique parameter makes the field value globally unique in a table). The model uses <code>TextField</code> for the <code>summary</code>, because this text may need to be quite long.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">from</span> django<span class=\"token punctuation\">.</span>urls <span class=\"token keyword\">import</span> reverse <span class=\"token comment\"># Used to generate URLs by reversing the URL patterns</span>\n\n<span class=\"token keyword\">class</span> <span class=\"token class-name\">Book</span><span class=\"token punctuation\">(</span>models<span class=\"token punctuation\">.</span>Model<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Model representing a book (but not a specific copy of a book).\"\"\"</span>\n title <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">200</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token comment\"># Foreign Key used because book can only have one author, but authors can have multiple books</span>\n <span class=\"token comment\"># Author is a string rather than an object because it hasn't been declared yet in the file</span>\n author <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>ForeignKey<span class=\"token punctuation\">(</span><span class=\"token string\">'Author'</span><span class=\"token punctuation\">,</span> on\_delete<span class=\"token operator\">=</span>models<span class=\"token punctuation\">.</span>SET\_NULL<span class=\"token punctuation\">,</span> null<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">)</span>\n\n summary <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>TextField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">1000</span><span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Enter a brief description of the book'</span><span class=\"token punctuation\">)</span>\n isbn <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span><span class=\"token string\">'ISBN'</span><span class=\"token punctuation\">,</span> max\_length<span class=\"token operator\">=</span><span class=\"token number\">13</span><span class=\"token punctuation\">,</span> unique<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span>\n help\_text<span class=\"token operator\">=</span><span class=\"token string\">'13 Character &lt;a href=\"https://www.isbn-international.org/content/what-isbn\"&gt;ISBN number&lt;/a&gt;'</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token comment\"># ManyToManyField used because genre can contain many books. Books can cover many genres.</span>\n <span class=\"token comment\"># Genre class has already been defined so we can specify the object above.</span>\n genre <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>ManyToManyField<span class=\"token punctuation\">(</span>Genre<span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Select a genre for this book'</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"String for representing the Model object.\"\"\"</span>\n <span class=\"token keyword\">return</span> self<span class=\"token punctuation\">.</span>title\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">get\_absolute\_url</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Returns the URL to access a detail record for this book.\"\"\"</span>\n <span class=\"token keyword\">return</span> reverse<span class=\"token punctuation\">(</span><span class=\"token string\">'book-detail'</span><span class=\"token punctuation\">,</span> args<span class=\"token operator\">=</span><span class=\"token punctuation\">[</span><span class=\"token builtin\">str</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">]</span><span class=\"token punctuation\">)</span>\n</code></pre></div>\n<p>The genre is a <code>ManyToManyField</code>, so that a book can have multiple genres and a genre can have many books. The author is declared as <code>ForeignKey</code>, so each book will only have one author, but an author may have many books (in practice a book might have multiple authors, but not in this implementation!)</p>\n<p>In both field types the related model class is declared as the first unnamed parameter using either the model class or a string containing the name of the related model. You must use the name of the model as a string if the associated class has not yet been defined in this file before it is referenced! The other parameters of interest in the <code>author</code> field are <code>null=True</code>, which allows the database to store a <code>Null</code> value if no author is selected, and <code>on\_delete=models.SET\_NULL</code>, which will set the value of the book's author field to <code>Null</code> if the associated author record is deleted.</p>\n<div class=\"notecard warning\" id=\"sect7\">\n <p><strong>Warning:</strong> By default <code>on\_delete=models.CASCADE</code>, which means that if the author was deleted, this book would be deleted too! We use <code>SET\_NULL</code> here, but we could also use <code>PROTECT</code> or <code>RESTRICT</code> to prevent the author being deleted while any book uses it.</p>\n</div>\n<p>The model also defines <code>\_\_str\_\_()</code>, using the book's <code>title</code> field to represent a <code>Book</code> record. The final method, <code>get\_absolute\_url()</code> returns a URL that can be used to access a detail record for this model (for this to work, we will have to define a URL mapping that has the name <code>book-detail</code>, and define an associated view and template).</p>"}},{"type":"prose","value":{"id":"bookinstance\_model","title":"BookInstance model","isH3":true,"content":"<p>Next, copy the <code>BookInstance</code> model (shown below) under the other models. The <code>BookInstance</code> represents a specific copy of a book that someone might borrow, and includes information about whether the copy is available or on what date it is expected back, \"imprint\" or version details, and a unique id for the book in the library.</p>\n<p>Some of the fields and methods will now be familiar. The model uses:</p>\n<ul>\n <li><code>ForeignKey</code> to identify the associated <code>Book</code> (each book can have many copies, but a copy can only have one <code>Book</code>). The key specifies <code>on\_delete=models.RESTRICT</code> to ensure that the <code>Book</code> cannot be deleted while referenced by a <code>BookInstance</code>.</li>\n <li><code>CharField</code> to represent the imprint (specific release) of the book.</li>\n</ul>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">import</span> uuid <span class=\"token comment\"># Required for unique book instances</span>\n\n<span class=\"token keyword\">class</span> <span class=\"token class-name\">BookInstance</span><span class=\"token punctuation\">(</span>models<span class=\"token punctuation\">.</span>Model<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Model representing a specific copy of a book (i.e. that can be borrowed from the library).\"\"\"</span>\n <span class=\"token builtin\">id</span> <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>UUIDField<span class=\"token punctuation\">(</span>primary\_key<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span> default<span class=\"token operator\">=</span>uuid<span class=\"token punctuation\">.</span>uuid4<span class=\"token punctuation\">,</span> help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Unique ID for this particular book across whole library'</span><span class=\"token punctuation\">)</span>\n book <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>ForeignKey<span class=\"token punctuation\">(</span><span class=\"token string\">'Book'</span><span class=\"token punctuation\">,</span> on\_delete<span class=\"token operator\">=</span>models<span class=\"token punctuation\">.</span>RESTRICT<span class=\"token punctuation\">,</span> null<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">)</span>\n imprint <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">200</span><span class=\"token punctuation\">)</span>\n due\_back <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>DateField<span class=\"token punctuation\">(</span>null<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span> blank<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">)</span>\n\n LOAN\_STATUS <span class=\"token operator\">=</span> <span class=\"token punctuation\">(</span>\n <span class=\"token punctuation\">(</span><span class=\"token string\">'m'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'Maintenance'</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">,</span>\n <span class=\"token punctuation\">(</span><span class=\"token string\">'o'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'On loan'</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">,</span>\n <span class=\"token punctuation\">(</span><span class=\"token string\">'a'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'Available'</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">,</span>\n <span class=\"token punctuation\">(</span><span class=\"token string\">'r'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'Reserved'</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">,</span>\n <span class=\"token punctuation\">)</span>\n\n status <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>\n max\_length<span class=\"token operator\">=</span><span class=\"token number\">1</span><span class=\"token punctuation\">,</span>\n choices<span class=\"token operator\">=</span>LOAN\_STATUS<span class=\"token punctuation\">,</span>\n blank<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span>\n default<span class=\"token operator\">=</span><span class=\"token string\">'m'</span><span class=\"token punctuation\">,</span>\n help\_text<span class=\"token operator\">=</span><span class=\"token string\">'Book availability'</span><span class=\"token punctuation\">,</span>\n <span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">class</span> <span class=\"token class-name\">Meta</span><span class=\"token punctuation\">:</span>\n ordering <span class=\"token operator\">=</span> <span class=\"token punctuation\">[</span><span class=\"token string\">'due\_back'</span><span class=\"token punctuation\">]</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"String for representing the Model object.\"\"\"</span>\n <span class=\"token keyword\">return</span> <span class=\"token string-interpolation\"><span class=\"token string\">f'</span><span class=\"token interpolation\"><span class=\"token punctuation\">{</span>self<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">}</span></span><span class=\"token string\"> (</span><span class=\"token interpolation\"><span class=\"token punctuation\">{</span>self<span class=\"token punctuation\">.</span>book<span class=\"token punctuation\">.</span>title<span class=\"token punctuation\">}</span></span><span class=\"token string\">)'</span></span>\n</code></pre></div>\n<p>We additionally declare a few new types of field:</p>\n<ul>\n <li><code>UUIDField</code> is used for the <code>id</code> field to set it as the <code>primary\_key</code> for this model. This type of field allocates a globally unique value for each instance (one for every book you can find in the library).</li>\n <li><code>DateField</code> is used for the <code>due\_back</code> date (at which the book is expected to become available after being borrowed or in maintenance). This value can be <code>blank</code> or <code>null</code> (needed for when the book is available). The model metadata (<code>Class Meta</code>) uses this field to order records when they are returned in a query.</li>\n <li><code>status</code> is a <code>CharField</code> that defines a choice/selection list. As you can see, we define a tuple containing tuples of key-value pairs and pass it to the choices argument. The value in a key/value pair is a display value that a user can select, while the keys are the values that are actually saved if the option is selected. We've also set a default value of 'm' (maintenance) as books will initially be created unavailable before they are stocked on the shelves.</li>\n</ul>\n<p>The method <code>\_\_str\_\_()</code> represents the <code>BookInstance</code> object using a combination of its unique id and the associated <code>Book</code>'s title.</p>\n<div class=\"notecard note\" id=\"sect8\">\n <p><strong>Note:</strong> A little Python:</p>\n <ul>\n <li>Starting with Python 3.6, you can use the string interpolation syntax (also known as f-strings): <code>f'{self.id} ({self.book.title})'</code>.</li>\n <li>In older versions of this tutorial, we were using a <a href=\"https://peps.python.org/pep-3101/\" class=\"external\" rel=\" noopener\">formatted string</a> syntax, which is also a valid way of formatting strings in Python (e.g. <code>'{0} ({1})'.format(self.id,self.book.title)</code>).</li>\n </ul>\n</div>"}},{"type":"prose","value":{"id":"author\_model","title":"Author model","isH3":true,"content":"<p>Copy the <code>Author</code> model (shown below) underneath the existing code in <strong>models.py</strong>.</p>\n<div class=\"code-example\"><pre class=\"brush: python notranslate\"><code><span class=\"token keyword\">class</span> <span class=\"token class-name\">Author</span><span class=\"token punctuation\">(</span>models<span class=\"token punctuation\">.</span>Model<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Model representing an author.\"\"\"</span>\n first\_name <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">100</span><span class=\"token punctuation\">)</span>\n last\_name <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>CharField<span class=\"token punctuation\">(</span>max\_length<span class=\"token operator\">=</span><span class=\"token number\">100</span><span class=\"token punctuation\">)</span>\n date\_of\_birth <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>DateField<span class=\"token punctuation\">(</span>null<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span> blank<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">)</span>\n date\_of\_death <span class=\"token operator\">=</span> models<span class=\"token punctuation\">.</span>DateField<span class=\"token punctuation\">(</span><span class=\"token string\">'Died'</span><span class=\"token punctuation\">,</span> null<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">,</span> blank<span class=\"token operator\">=</span><span class=\"token boolean\">True</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">class</span> <span class=\"token class-name\">Meta</span><span class=\"token punctuation\">:</span>\n ordering <span class=\"token operator\">=</span> <span class=\"token punctuation\">[</span><span class=\"token string\">'last\_name'</span><span class=\"token punctuation\">,</span> <span class=\"token string\">'first\_name'</span><span class=\"token punctuation\">]</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">get\_absolute\_url</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"Returns the URL to access a particular author instance.\"\"\"</span>\n <span class=\"token keyword\">return</span> reverse<span class=\"token punctuation\">(</span><span class=\"token string\">'author-detail'</span><span class=\"token punctuation\">,</span> args<span class=\"token operator\">=</span><span class=\"token punctuation\">[</span><span class=\"token builtin\">str</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">.</span><span class=\"token builtin\">id</span><span class=\"token punctuation\">)</span><span class=\"token punctuation\">]</span><span class=\"token punctuation\">)</span>\n\n <span class=\"token keyword\">def</span> <span class=\"token function\">\_\_str\_\_</span><span class=\"token punctuation\">(</span>self<span class=\"token punctuation\">)</span><span class=\"token punctuation\">:</span>\n <span class=\"token triple-quoted-string string\">\"\"\"String for representing the Model object.\"\"\"</span>\n <span class=\"token keyword\">return</span> <span class=\"token string-interpolation\"><span class=\"token string\">f'</span><span class=\"token interpolation\"><span class=\"token punctuation\">{</span>self<span class=\"token punctuation\">.</span>last\_name<span class=\"token punctuation\">}</span></span><span class=\"token string\">, </span><span class=\"token interpolation\"><span class=\"token punctuation\">{</span>self<span class=\"token punctuation\">.</span>first\_name<span class=\"token punctuation\">}</span></span><span class=\"token string\">'</span></span>\n</code></pre></div>\n<p>All of the fields/methods should now be familiar. The model defines an author as having a first name, last name, and dates of birth and death (both optional). It specifies that by default the <code>\_\_str\_\_()</code> returns the name in <em>last name</em>, <em>firstname</em> order. The <code>get\_absolute\_url()</code> method reverses the <code>author-detail</code> URL mapping to get the URL for displaying an individual author.</p>"}},{"type":"prose","value":{"id":"re-run\_the\_database\_migrations","title":"Re-run the database migrations","isH3":false,"content":"<p>All your models have now been created. Now re-run your database migrations to add them to your database.</p>\n<div class=\"code-example\"><pre class=\"brush: bash notranslate\"><code>python3 manage.py makemigrations\npython3 manage.py migrate\n</code></pre></div>"}},{"type":"prose","value":{"id":"language\_model\_—\_challenge","title":"Language model — challenge","isH3":false,"content":"<p>Imagine a local benefactor donates a number of new books written in another language (say, Farsi). The challenge is to work out how these would be best represented in our library website, and then to add them to the models.</p>\n<p>Some things to consider:</p>\n<ul>\n <li>Should \"language\" be associated with a <code>Book</code>, <code>BookInstance</code>, or some other object?</li>\n <li>Should the different languages be represented using model, a free text field, or a hard-coded selection list?</li>\n</ul>\n<p>After you've decided, add the field. You can see what we decided on GitHub <a href=\"https://github.com/mdn/django-locallibrary-tutorial/blob/master/catalog/models.py\" class=\"external\" rel=\" noopener\">here</a>.</p>\n<p>Don't forget that after a change to your model, you should again re-run your database migrations to add the changes.</p>\n<div class=\"code-example\"><pre class=\"brush: bash notranslate\"><code>python3 manage.py makemigrations\npython3 manage.py migrate\n</code></pre></div>"}},{"type":"prose","value":{"id":"summary","title":"Summary","isH3":false,"content":"<p>In this article we've learned how models are defined, and then used this information to design and implement appropriate models for the <em>LocalLibrary</em> website.</p>\n<p>At this point we'll divert briefly from creating the site, and check out the <em>Django Administration site</em>. This site will allow us to add some data to the library, which we can then display using our (yet to be created) views and templates.</p>"}},{"type":"prose","value":{"id":"see\_also","title":"See also","isH3":false,"content":"<ul>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/intro/tutorial02/\" class=\"external\" rel=\" noopener\">Writing your first Django app, part 2</a> (Django docs)</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/topics/db/queries/\" class=\"external\" rel=\" noopener\">Making queries</a> (Django Docs)</li>\n <li><a href=\"https://docs.djangoproject.com/en/4.0/ref/models/querysets/\" class=\"external\" rel=\" noopener\">QuerySet API Reference</a> (Django Docs)</li>\n</ul><ul class=\"prev-next\">\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django/skeleton\_website\"><span class=\"button-wrap\"> Previous </span></a></li>\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django\"><span class=\"button-wrap\"> Overview: Django</span></a></li>\n <li><a class=\"button secondary\" href=\"/en-US/docs/Learn/Server-side/Django/Admin\_site\"><span class=\"button-wrap\"> Next </span></a></li>\n</ul>"}},{"type":"prose","value":{"id":"in\_this\_module","title":"In this module","isH3":false,"content":"<ul>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Introduction\">Django introduction</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/development\_environment\">Setting up a Django development environment</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Tutorial\_local\_library\_website\">Django Tutorial: The Local Library website</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/skeleton\_website\">Django Tutorial Part 2: Creating a skeleton website</a></li>\n <li><strong>Django Tutorial Part 3: Using models</strong></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Admin\_site\">Django Tutorial Part 4: Django admin site</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Home\_page\">Django Tutorial Part 5: Creating our home page</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Generic\_views\">Django Tutorial Part 6: Generic list and detail views</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Sessions\">Django Tutorial Part 7: Sessions framework</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Authentication\">Django Tutorial Part 8: User authentication and permissions</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Forms\">Django Tutorial Part 9: Working with forms</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Testing\">Django Tutorial Part 10: Testing a Django web application</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/Deployment\">Django Tutorial Part 11: Deploying Django to production</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/web\_application\_security\">Django web application security</a></li>\n <li><a href=\"/en-US/docs/Learn/Server-side/Django/django\_assessment\_blog\">DIY Django mini blog</a></li>\n</ul>"}}],"toc":[{"text":"Overview","id":"overview"},{"text":"Designing the LocalLibrary models","id":"designing\_the\_locallibrary\_models"},{"text":"Model primer","id":"model\_primer"},{"text":"Defining the LocalLibrary Models","id":"defining\_the\_locallibrary\_models"},{"text":"Re-run the database migrations","id":"re-run\_the\_database\_migrations"},{"text":"Language model — challenge","id":"language\_model\_—\_challenge"},{"text":"Summary","id":"summary"},{"text":"See also","id":"see\_also"},{"text":"In this module","id":"in\_this\_module"}],"summary":"In this article we've learned how models are defined, and then used this information to design and implement appropriate models for the LocalLibrary website.","popularity":0.0073,"modified":"2022-08-17T07:28:09.000Z","other\_translations":[{"title":"Tutorial Django Parte 3: Uso de modelos","locale":"es","native":"Español"},{"title":"Django didactique Section 3: Utilisation des modèles de données","locale":"fr","native":"Français"},{"title":"Django チュートリアル Part 3: モデルの使用","locale":"ja","native":"日本語"},{"title":"Django Tutorial Part 3: Using models","locale":"ko","native":"한국어"},{"title":"Tutorial Django Parte 3: Usando models","locale":"pt-BR","native":"Português (do Brasil)"},{"title":"Django учебник Часть 3: Использование моделей","locale":"ru","native":"Русский"},{"title":"Django Tutorial Part 3: 使用模型","locale":"zh-CN","native":"中文 (简体)"},{"title":"Django Tutorial Part 3: Using models","locale":"zh-TW","native":"正體中文 (繁體)"}],"source":{"folder":"en-us/learn/server-side/django/models","github\_url":"https://github.com/mdn/content/blob/main/files/en-us/learn/server-side/django/models/index.md","last\_commit\_url":"https://github.com/mdn/content/commit/4d2655895e92601ca5292d5e4ebdf961055137f4","filename":"index.md"},"short\_title":"Django Tutorial Part 3: Using models","parents":[{"uri":"/en-US/docs/Learn","title":"Guides"},{"uri":"/en-US/docs/Learn/Server-side","title":"Server-side website programming"},{"uri":"/en-US/docs/Learn/Server-side/Django","title":"Django Web Framework (Python)"},{"uri":"/en-US/docs/Learn/Server-side/Django/Models","title":"Django Tutorial Part 3: Using models"}],"pageTitle":"Django Tutorial Part 3: Using models - Learn web development | MDN","noIndexing":false}}