Table of Contents

Step 0: What is it about? 15

Step 1: Checking your Work Environment 17
1.1: A Computer 17
1.2: Opinionated Choices 17
1.3: IDE 18
1.4: Terminal 18
1.5: Git 18
1.6: PHP 19
1.7: Composer 19
1.8: Docker and Docker Compose 19
1.9: Symfony CLI 19

Step 2: Introducing the Project 21
2.1: Revealing the Project 21
2.2: Learning is Doing 22
2.3: Getting the Project Source Code 22
2.4: Navigating the Source Code 23

Step 3: Going from Zero to Production 25
3.1: Initializing the Project 26
3.2: Creating some Public Resources 27
3.3: Launching a Local Web Server 28
3.4: Adding a favicon 29
3.5: Preparing for Production 30
3.6: Going to Production 30

Step 4: Adopting a Methodology 33
4.1: Implementing a Git Strategy 33
4.2: Deploying to Production Continuously 34

Step 5: Troubleshooting Problems 35
5.1: Installing more Dependencies 35
5.2: Understanding Symfony Environments 36
5.3: Managing Environment Configurations 36
5.4: Logging all the Things 37
5.5: Discovering the Symfony Debugging Tools 37
5.6: Debugging Production 40

Step 6: Creating a Controller 43
6.1: Being Lazy with the Maker Bundle 43
6.2: Choosing a Configuration Format 44
6.3: Generating a Controller 44
6.4: Adding an Easter Egg 46

Step 7: Storing Data 49
7.1: Adding PostgreSQL to Docker Compose 49
7.2: Starting Docker Compose 50
7.3: Accessing the Local Database 50
7.4: Adding PostgreSQL to SymfonyCloud 51
7.5: Accessing the SymfonyCloud Database 52
7.6: Exposing Environment Variables 53

Step 8: Describing the Data Structure 55
8.1: Managing the Data 55
8.2: Configuring Doctrine ORM 55
8.3: Understanding Symfony Environment Variable Conventions 56
8.4: Creating Entity Classes 57
8.5: Linking Entities 60
8.6: Adding more Properties 63
8.7: Migrating the Database 64
8.8: Updating the Local Database 65
8.9: Updating the Production Database 65

Step 9: Setting up an Admin Backend 67
9.1: Configuring EasyAdmin 67
9.2: Customizing EasyAdmin 70

Step 10: Building the User Interface 73
10.1: Installing Twig 73
10.2: Using Twig for the Templates 74
10.3: Using Twig in a Controller 75
10.4: Creating the Page for a Conference 76
10.5: Linking Pages Together 78

10.6: Paginating the Comments 79
10.7: Refactoring the Controller 83

Step 11: Branching the Code 85
11.1: Adopting a Git Workflow 85
11.2: Describing your Infrastructure 86
11.3: Creating Branches 86
11.4: Storing Sessions in Redis 86
11.5: Deploying a Branch 88
11.6: Debugging Production Deployments before Deploying 89
11.7: Testing Production Deployments before Deploying 89
11.8: Merging to Production 90
11.9: Cleaning up 90

Step 12: Listening to Events 91
12.1: Adding a Website Header 91
12.2: Discovering Symfony Events 92
12.3: Implementing a Subscriber 93
12.4: Sorting Conferences by Year and City 94

Step 13: Managing the Lifecycle of Doctrine Objects 97
13.1: Defining Lifecycle Callbacks 97
13.2: Adding Slugs to Conferences 98
13.3: Generating Slugs 100
13.4: Defining a Complex Lifecycle Callback 101
13.5: Configuring a Service in the Container 102
13.6: Using Slugs in the Application 103

Step 14: Accepting Feedback with Forms 105
14.1: Generating a Form Type 105
14.2: Displaying a Form 106
14.3: Customizing a Form Type 108
14.4: Validating Models 110
14.5: Handling a Form 111
14.6: Uploading Files 112
14.7: Debugging Forms 114
14.8: Displaying Uploaded Photos in the Admin Backend 115
14.9: Excluding Uploaded Photos from Git 116
14.10: Storing Uploaded Files on Production Servers 116

Step 15: Securing the Admin Backend 119
15.1: Generating a Password for the Admin User 119
15.2: Configuring the Security 120

Step 16: Preventing Spam with an API 123
16.1: Signing up on Akismet 123
16.2: Depending on Symfony HTTPClient Component 123
16.3: Designing a Spam Checker Class 124
16.4: Using Environment Variables 125
16.5: Storing Secrets 126
16.6: Checking Comments for Spam 127
16.7: Managing Secrets in Production 128

Step 17: Testing 129
17.1: Writing Unit Tests 129
17.2: Writing Functional Tests for Controllers 131
17.3: Defining Fixtures 133
17.4: Crawling a Website in Functional Tests 134
17.5: Working with a Test Database 135
17.6: Submitting a Form in a Functional Test 136
17.7: Reloading the Fixtures 137
17.8: Automating your Workflow with a Makefile 137
17.9: Resetting the Database after each Test 138
17.10: Using a real Browser for Functional Tests 140

Step 18: Going Async 143
18.1: Flagging Comments 143
18.2: Understanding Messenger 146
18.3: Coding a Message Handler 146
18.4: Restricting Displayed Comments 149
18.5: Going Async for Real 150
18.6: Adding RabbitMQ to the Docker Stack 150
18.7: Restarting Docker Services 151
18.8: Consuming Messages 151
18.9: Exploring the RabbitMQ Web Management Interface 152
18.10: Running Workers in the Background 153
18.11: Retrying Failed Messages 154
18.12: Deploying RabbitMQ 155
18.13: Running Workers on SymfonyCloud 156

Step 19: Making Decisions with a Workflow 159
19.1: Describing Workflows 159
19.2: Using a Workflow 161

Step 20: Emailing Admins 165
20.1: Setting an Email for the Admin 165

20.2: Sending a Notification Email 166
20.3: Extending the Notification Email Template....................168
20.4: Generating Absolute URLs in a Command.....................168
20.5: Wiring a Route to a Controller 169
20.6: Using a Mail Catcher...171
20.7: Accessing the Webmail..171
20.8: Managing Long-Running Scripts......................................173
20.9: Sending Emails Asynchronously 173
20.10: Testing Emails...174
20.11: Sending Emails on SymfonyCloud 175

Step 21: Caching for Performance 177
21.1: Adding HTTP Cache Headers...177
21.2: Activating the Symfony HTTP Cache Kernel 178
21.3: Avoiding SQL Requests with ESI 179
21.4: Purging the HTTP Cache for Testing 183
21.5: Grouping similar Routes with a Prefix.............................185
21.6: Caching CPU/Memory Intensive Operations.....................186
21.7: Profiling and Comparing Performance 187
21.8: Configuring a Reverse Proxy Cache on Production 188
21.9: Enabling ESI Support on Varnish.....................................188
21.10: Purging the Varnish Cache..189

Step 22: Styling the User Interface with Webpack 191
22.1: Using Sass...191
22.2: Leveraging Bootstrap..192
22.3: Styling the HTML 193
22.4: Building Assets..194

Step 23: Resizing Images 197
23.1: Optimizing Images with Imagine 198
23.2: Adding a new Step in the Workflow..................................199
23.3: Storing Uploaded Data in Production 201

Step 24: Running Crons 203
24.1: Cleaning up Comments..203
24.2: Using Class Constants, Container Parameters, and Environment Variables 204
24.3: Creating a CLI Command...205
24.4: Setting up a Cron on SymfonyCloud.................................206

Step 25: Notifying by all Means..209
25.1: Sending Web Application Notifications in the Browser..........209

25.2: Notifying Admins by Email 212
25.3: Chatting with Admins 215
25.4: Going Asynchronous across the Board 220
25.5: Notifying Users by Email..221

Step 26: Exposing an API with API Platform 223
26.1: Installing API Platform 223
26.2: Exposing an API for Conferences 224
26.3: Exposing an API for Comments 225
26.4: Configuring CORS 227
26.5: Configuring CORS on SymfonyCloud.......................................227

Step 27: Building an SPA...229
27.1: Creating the Application...229
27.2: Creating the SPA Main Template 231
27.3: Running an SPA in the Browser 232
27.4: Adding a Router to handle States 232
27.5: Styling the SPA...234
27.6: Fetching Data from the API 235
27.7: Using Cordova to build a Smartphone Application.................239
27.8: Deploying the SPA on SymfonyCloud 240