

contents

<i>foreword</i>	<i>xiii</i>
<i>preface</i>	<i>xv</i>
<i>acknowledgments</i>	<i>xvii</i>
<i>about this book</i>	<i>xix</i>
<i>about the cover illustration</i>	<i>xxiii</i>

PART I GETTING STARTED WITH REACTIVE WEB APPLICATIONS ... I

I	<i>Did you say reactive?</i>	3			
1.1	Putting reactive into context	4			
	<i>Origins of reactive</i>	<i>4</i> ▪ <i>The Reactive Manifesto</i>	<i>5</i> ▪ <i>Reactive programming</i>	<i>6</i> ▪ <i>The emergence of reactive technologies</i>	<i>7</i>
1.2	Rethinking computational resource utilization	8			
	<i>Threaded versus evented web application servers</i>	<i>9</i> ▪ <i>Developing web applications fit for multicore architectures</i>	<i>11</i> ▪ <i>The horizontal application architecture</i>	<i>14</i>	
1.3	Failure-handling as first-class concern	17			
	<i>Failure is inevitable</i>	<i>17</i> ▪ <i>Building applications with failure in mind</i>	<i>20</i> ▪ <i>Dealing with load</i>	<i>22</i>	
1.4	Summary	25			

- 2 Your first reactive web application 26**
- 2.1 Creating and running a new project 26
 - 2.2 Connecting to Twitter's streaming API 29
 - Getting the connection credentials to the Twitter API 29* ▪ *Working around a bug with OAuth authentication 30* ▪ *Streaming data from the Twitter API 30* ▪ *Asynchronously transforming the Twitter stream 35*
 - 2.3 Streaming tweets to clients using a WebSocket 38
 - Creating an actor 39* ▪ *Setting up the WebSocket connection and interacting with it 40* ▪ *Sending tweets to the WebSocket 42*
 - 2.4 Making the application resilient and scaling out 45
 - Making the client resilient 45* ▪ *Scaling out 46*
 - 2.5 Summary 49
- 3 Functional programming primer 50**
- 3.1 A few words on functional programming 50
 - 3.2 Immutability 51
 - The fallacy of mutable state 51* ▪ *Immutable values as snapshots of reality 53* ▪ *Expression-oriented programming 53*
 - 3.3 Functions 55
 - Functions in object-oriented programming languages 56*
 - Functions as first-class values 56* ▪ *Moving behavior around 57*
 - Composing functions 58* ▪ *The size of functions 59*
 - 3.4 Manipulating immutable collections 61
 - Transformations instead of loops 61* ▪ *Higher-order functions for manipulating collections 62*
 - 3.5 Making the switch to a declarative programming style 68
 - Never use the get method on an Option 69* ▪ *Only use immutable values and data structures 69* ▪ *Aim for small and crisp functions 69* ▪ *Iterate and refine your functional style 70*
 - 3.6 Summary 70
- 4 Quick introduction to Play 71**
- 4.1 Play application structure and configuration 72
 - Introducing the Simple Vocabulary Teacher 72* ▪ *Creating a minimal Play application scaffold 73* ▪ *Building the project 75*
 - 4.2 Request handling 77
 - The request lifecycle 77* ▪ *Request routing 80* ▪ *Controllers, actions, and results 85* ▪ *WebSockets 91* ▪ *Altering the default request-handling pipeline 95*
 - 4.3 Summary 99

PART 2 CORE CONCEPTS..... 101

5 **Futures 103**

- 5.1 Working with futures 103
 - Future fundamentals 104* ▪ *Futures in Play 110* ▪ *Testing futures 119*
- 5.2 Designing asynchronous business logic with futures 120
 - Identifying parallelizable elements 121* ▪ *Composing the service's futures 123* ▪ *Propagating and handling errors 128*
- 5.3 Summary 132

6 **Actors 134**

- 6.1 Actor fundamentals 135
 - A simple Twitter analytics service 135* ▪ *Laying out the foundation: actors and their children 136*
- 6.2 Letting it crash—supervision and recovery 149
 - Robust storage 150* ▪ *Letting it crash 152* ▪ *Watching actors die and reviving them 154*
- 6.3 Reacting to load patterns for monitoring and preventing service overload 155
 - Control-flow messages 155* ▪ *Prioritizing messages 159*
 - Circuit breakers 161*
- 6.4 Summary 163

7 **Dealing with state 164**

- 7.1 Working with state in a stateless Play web application 165
 - Databases 166* ▪ *Client-side state using the Play session 177*
 - Server-side state using a distributed cache 179*
- 7.2 Command and Query Responsibility Segregation and Event Sourcing 181
 - The Twitter SMS service 182* ▪ *Setting up the SMS gateway 185*
 - Writing the event stream with persistent actors 188* ▪ *Configuring Akka persistence to write to MongoDB 191* ▪ *Handling an incoming command: subscribing to user mentions 192*
 - Transforming the event stream into a relational model 194*
 - Querying the relational model 197* ▪ *A word on eventual consistency 199*
- 7.3 Summary 200

8 *Responsive user interfaces* 201

- 8.1 Integrating Scala.js and Play 202
 - The application structure* 203 ▪ *Setting up the build process* 203
 - Creating a simple Scala.js application* 205
- 8.2 Integrating Scala.js and AngularJS 207
 - Setting up the AngularJS bindings* 207 ▪ *Creating the AngularJS application* 208 ▪ *Initializing the AngularJS dashboard module and its dependencies* 210 ▪ *Initializing the Dashboard controller* 210 ▪ *Creating the partial view* 211 ▪ *Loading the AngularJS application in HTML* 211
- 8.3 Integrating existing JavaScript libraries with Scala.js 212
 - Wrapping an existing JavaScript library as an AngularJS service* 212 ▪ *Creating a service to fetch data for a graph* 214
 - Displaying metrics using the Chart.js library* 216
- 8.4 Handling client-side failure 218
 - Preventing bugs with tests* 219 ▪ *Detecting WebSocket connection failure* 220 ▪ *Notifying users* 221 ▪ *Monitoring client-side errors* 222
- 8.5 Summary 223

PART 3 ADVANCED TOPICS.....225

9 *Reactive Streams* 227

- 9.1 Why Reactive Streams 228
 - Streaming with nonblocking back pressure* 228 ▪ *Manipulating asynchronous streams* 229
- 9.2 Introducing Akka Streams 230
 - Core principles* 230 ▪ *Manipulating streaming tweets* 231
- 9.3 Summary 243

10 *Deploying reactive Play applications* 244

- 10.1 Preparing a Play application for production 245
 - Creating a simple application to deploy* 246 ▪ *Writing and running integration tests with Selenium* 248 ▪ *Preparing the application for production* 249
- 10.2 Setting up continuous integration 252
 - Running Jenkins via Docker* 253 ▪ *Configuring Jenkins to build our application* 254

- 10.3 Deploying the application 256
 - Deployment on Clever Cloud* 256
 - *Deployment on your own server* 258
 - *Which deployment model to use* 261
- 10.4 Summary 262

11 Testing reactive web applications 263

- 11.1 Testing reactive traits 264
 - Testing responsiveness* 264
 - *Testing resilience* 265
 - *Testing elasticity* 265
 - *Where to test?* 265
- 11.2 Testing individual reactive components 266
 - Testing individual components for responsiveness* 266
 - *Testing individual components for resilience* 271
- 11.3 Testing the entire reactive application 274
 - Creating a simple application to generate random numbers* 274
 - Testing for resilience with Gatling* 277
 - *Testing for scalability with Bees with Machine Guns* 281
- 11.4 Summary 286

appendix A Installing the Play Framework 287

appendix B Recommended reading 290

appendix C Further reading 291

index 293