

brief contents

PART 1	BACKGROUND	1
1	■ Introduction to Spring Integration	3
2	■ Enterprise integration fundamentals	24
PART 2	MESSAGING.....	43
3	■ Messages and channels	45
4	■ Message Endpoints	63
5	■ Getting down to business	80
6	■ Go beyond sequential processing: routing and filtering	104
7	■ Splitting and aggregating messages	122
PART 3	INTEGRATING SYSTEMS	139
8	■ Handling messages with XML payloads	141
9	■ Spring Integration and the Java Message Service	155
10	■ Email-based integration	180
11	■ Filesystem integration	191
12	■ Spring Integration and web services	208
13	■ Chatting and tweeting	219

PART 4 ADVANCED TOPICS.....237

- 14 ■ Monitoring and management 239
- 15 ■ Managing scheduling and concurrency 258
- 16 ■ Batch applications and enterprise integration 276
- 17 ■ Scaling messaging applications with OSGi 292
- 18 ■ Testing 304

contents

foreword xv
preface xvii
acknowledgments xix
about this book xxii
author online xxvii
about the authors xxviii
about the cover illustration xxix

PART 1 BACKGROUND 1

1 Introduction to Spring Integration 3

- 1.1 Spring Integration's architecture 5
- 1.2 Spring Integration's support for enterprise integration patterns 8
 - Messages 8* ▪ *Message Channels 9* ▪ *Message endpoints 10*
- 1.3 Enterprise integration patterns meet Inversion of Control 13
 - Dependency injection 13* ▪ *Method invocation 16*
- 1.4 Say hello to Spring Integration 19
- 1.5 Summary 22

2 Enterprise integration fundamentals 24

- 2.1 Loose coupling and event-driven architecture 25
 - Why should you care about loose coupling?* 25
 - Type-level coupling* 27
 - Loosening type-level coupling with dependency injection* 28
 - System-level coupling* 30
 - Event-driven architecture* 32
- 2.2 Synchronous and asynchronous communication 32
 - What's the difference?* 33
 - Where does Spring Integration fit in?* 36
- 2.3 Comparing enterprise integration styles 38
 - Integrating applications by transferring files* 39
 - Interacting through a shared database* 39
 - Exposing a remote API through Remote Procedure Calls* 40
 - Exchanging messages* 41
- 2.4 Summary 41

PART 2 MESSAGING43

3 Messages and channels 45

- 3.1 Introducing Spring Integration messages 46
 - What's in a message?* 46
 - How it's done in Spring Integration* 47
- 3.2 Introducing Spring Integration channels 49
 - Using channels to move messages* 50
 - I'll let you know when I've got something!* 50
 - Do you have any messages for me?* 50
 - The right channel for the job* 51
 - A channel selection example* 53
- 3.3 Channel collaborators 57
 - MessageDispatcher* 57
 - ChannelInterceptor* 59
- 3.4 Summary 62

4 Message Endpoints 63

- 4.1 What can you expect of an endpoint? 65
 - To poll or not to poll?* 66
 - Inbound endpoints* 67
 - Outbound endpoints* 68
 - Unidirectional and bidirectional endpoints* 69
- 4.2 Transaction boundaries around endpoints 70
 - Why sharing isn't always a good thing* 70
 - What are transactions, and can we get by without them?* 71

- 4.3 Under the hood 74
 - Endpoint parsing* 75
 - *Endpoint instantiation* 76
- 4.4 Summary 78

5 *Getting down to business* 80

- 5.1 Domain-driven transformation 81
 - Marshalling flight information* 82
 - *Using the simplest possible data representation* 84
 - *Wiring the components together* 86
 - Testing the transformer* 88
 - *Content enricher* 90
 - *Header enricher* 91
- 5.2 Message-driven services 94
 - The Service Activator pattern* 94
 - *The Return Address pattern* 94
- 5.3 Message publishing interceptors 96
- 5.4 Domain-driven Messaging Gateways 97
- 5.5 Chaining endpoints 100
- 5.6 Summary 102

6 *Go beyond sequential processing: routing and filtering* 104

- 6.1 Do you want to get this message? 105
 - Filtering out messages* 105
 - *Using filters for selective processing* 109
- 6.2 Whose message is this, anyway? 110
 - Configuring routers* 111
 - *Routers provided by the framework* 114
 - *Routers with multiple destinations* 117
- 6.3 Under the hood 119
 - The message filter API* 119
 - *The message router API* 120
- 6.4 Summary 121

7 *Splitting and aggregating messages* 122

- 7.1 Introducing correlation 123
 - A real-life example* 124
 - *Correlating messages* 125
- 7.2 Splitting, aggregating, and resequencing 126
 - The art of dividing: the splitter* 126
 - *How to get the big picture: the aggregator* 127
 - *Doing things in the right order: the resequencer* 128

- 7.3 Useful patterns 130
 - Grouping messages based on timing* 131 ■ *Scatter-gather* 132
- 7.4 Under the hood 134
 - Extension points of the CorrelatingMessageHandler* 134
 - How do Resequencer and Aggregator do it?* 135
- 7.5 Summary 136

PART 3 INTEGRATING SYSTEMS 139

8 *Handling messages with XML payloads* 141

- 8.1 XML messaging 142
 - Marshalling LegQuoteCommand into XML* 143 ■ *Enriching the leg quote using XSLT* 148 ■ *XPath support* 150 ■ *Splitting hotel, car rental, and flight quotes* 150 ■ *Routing messages based on their XML payloads* 151 ■ *Validating XML messages* 152
- 8.2 Under the hood 153
 - Supported payload types and return type matching* 154
- 8.3 Summary 154

9 *Spring Integration and the Java Message Service* 155

- 9.1 The relationship between Spring Integration and JMS 156
 - Mapping between JMS and Spring Integration messages* 159
 - Comparing JMS destinations and Spring Integration message channels* 160
- 9.2 JMS support in the Spring Framework 161
- 9.3 Asynchronous JMS message reception with Spring 163
 - Why go asynchronous?* 163 ■ *Spring's MessageListener container* 164 ■ *Message-driven POJOs with Spring* 165
- 9.4 Sending JMS messages from a Spring Integration application 166
- 9.5 Receiving JMS messages in a Spring Integration application 168
- 9.6 Request-reply messaging 169
 - The outbound gateway* 169 ■ *The inbound gateway* 170
- 9.7 Messaging between multiple Spring Integration runtimes 172

- 9.8 Managing transactions with JMS channel adapters and gateways 175
 - JMS transaction basics* 175
 - *A note about distributed transactions* 177
- 9.9 Summary 179

10 *Email-based integration* 180

- 10.1 Sending email 181
 - The outbound channel adapter* 182
 - *Advanced configuration options* 183
 - *Transforming outbound messages* 185
- 10.2 Receiving email 186
 - Polling for emails* 187
 - *Event-driven email reception* 188
 - Transforming inbound messages* 189
- 10.3 Summary 190

11 *Filesystem integration* 191

- 11.1 Can you be friends with the filesystem? 192
 - A file-based collaborative trip diary editor* 193
- 11.2 Writing files 195
 - Configuring the file-writing endpoint* 195
 - *Writing increments from the collaborative editor* 197
- 11.3 Reading files 198
 - A File in Java isn't a file on your disk* 198
 - *Configuring the file-reading endpoint* 199
 - *From the example: picking up incremental updates* 201
- 11.4 Handling file-based messages 201
 - Transforming files into objects* 202
 - *Common scenarios when dealing with files* 202
 - *Configuring file transformers* 203
 - Applying incoming changes to the collaborative editor* 204
- 11.5 Under the hood 204
 - FileReadingMessageSource* 205
- 11.6 Summary 207

12 *Spring Integration and web services* 208

- 12.1 XML web services with Spring WS 210
 - Exposing a Spring WS-based inbound gateway* 211
 - Calling a web service with the outbound gateway* 212
 - *Marshalling support* 213

- 12.2 Simple HTTP endpoints 213
 - Processing HTTP inbound requests* 214
 - *Inbound-only messages using inbound-channel-adapter* 216
 - *Outbound HTTP requests* 217
 - *Outbound channel adapter* 217
- 12.3 Summary 218

13 Chatting and tweeting 219

- 13.1 XMPP 220
 - Sending XMPP messages* 220
 - *Receiving XMPP messages* 225
 - Sending and receiving presence status updates* 225
- 13.2 Twitter 226
 - Receiving messages from a Twitter search* 227
 - *OAuth configuration for the Twitter template* 229
 - *Receiving messages from your Twitter timeline* 230
 - *Sending messages to update your Twitter status* 231
 - *Receiving messages from Twitter retweets, replies, and mentions* 231
 - *Sending and receiving direct messages via Twitter* 233
- 13.3 Future directions 234
- 13.4 Summary 235

PART 4 ADVANCED TOPICS..... 237

14 Monitoring and management 239

- 14.1 Message history 240
- 14.2 Wire Tap 242
- 14.3 JMX support in Spring Integration 247
 - Monitoring channels and endpoints with JMX* 248
 - Integration using JMX adapters* 251
- 14.4 Control Bus 252
 - Spring's support for management annotations* 253
 - Using SpEL for control messages* 254
 - *Using Groovy for control messages* 255
- 14.5 Under the hood 256
- 14.6 Summary 257

- 15** *Managing scheduling and concurrency* 258
- 15.1 Controlling timed events 259
 - Pollers and their configuration* 259
 - *Controlling the polling frequency* 261
 - *Scheduling jobs at precise times* 262
 - Advanced configuration options* 263
 - *Publishing messages according to a schedule* 265
 - 15.2 Managing concurrency 266
 - Breaking down the thread* 266
 - *Configuring the infrastructure* 269
 - 15.3 Under the hood 272
 - The TaskExecutor API* 272
 - *The TaskScheduler API* 273
 - 15.4 Summary 274
- 16** *Batch applications and enterprise integration* 276
- 16.1 Introducing batch jobs 277
 - Online or batch, that's the question* 277
 - *Batch processing: what's it good for?* 278
 - *Batch by example* 279
 - 16.2 Introducing Spring Batch 281
 - A batch job in five minutes* 281
 - *Getting the job done* 284
 - 16.3 Integrating Spring Batch and Spring Integration 285
 - Launching batch jobs through messages* 286
 - *Providing feedback with informational messages* 288
 - *Externalizing batch process execution* 289
 - 16.4 Summary 291
- 17** *Scaling messaging applications with OSGi* 292
- 17.1 The OSGi module system 294
 - The bundle lifecycle in an OSGi environment* 295
 - 17.2 Accessing the Service Registry through Gemini Blueprint 296
 - 17.3 Messaging between bundles 298
 - Reasons to combine OSGi with messaging* 299
 - *Publish-subscribe messaging between bundles* 300
 - *Point-to-point messaging and sharing the load* 301
 - *Using gateways and service activators to avoid Spring Integration dependencies* 301
 - 17.4 Summary 302

18 Testing 304

- 18.1 Matching messages with the Spring Integration testing framework 306
 - Unwrapping payloads* 307
 - *Expectations on headers* 309
- 18.2 Mocking services out of integration tests 311
- 18.3 Testing an asynchronous system 313
 - Can't we wait for the message to come out the other end?* 313
 - Avoiding the wicked ways of debugging* 314
 - *Injecting latches into endpoints* 315
 - *Structuring the configuration to facilitate testing* 317
 - *How do I prove my code thread safe?* 318
- 18.4 Summary 318

index 321