Tuscany SCA in Action

SIMON LAWS
MARK COMBELLACK
RAYMOND FENG
HALEH MAHBOD
SIMON NASH

MANNING
Greenwich
(74° w. long.)
TABLE OF CONTENTS

VII
Preface xvii
Acknowledgments xix
About this Book xx
About the Authors xxiii
About the Title xxv
About the Cover Illustration xxviii

Part 1 Understanding Tuscany and SCA 1

Introduction Tuscany and SCA 3

1.1 The Big Picture 5

The Basics of SCA 5 • Tuscany’s Java runtime for SCA 8

1.2 Designing a Sample Composite Application 10

The Travel-Booking Application 10 • SCA Components, Services, and References 12 • A User Scenario Demonstrating Service Interactions 12

1.3 Implementing a Composite Application 14

A Jump-Start to Building and Running Your First SCA Component 14 • Defining More Complex Components 18
Creating Component Implementations 21 • Wiring
### CONTENTS

- components to form a composite application 24
- Deploying a composite application using contributions 25

1.4 Working with other SOA technologies 28
- API wrapping 29
- Using SCA implementations 29
- Using SCA remote bindings 30
- Tuscany and an Enterprise Service Bus 31

1.5 Summary 32

### 2 Using SCA components 33

2.1 Implementing an SCA component 34
- Choosing an implementation type 35
- Configuring SCA components using component definitions 36
- Discovering or defining the component type 38

2.2 Using components to provide services 39
- Defining services 40
- Interface definition in SCA 41
- Configuring services in component definitions 43
- Local and remotable interfaces 44
- Bidirectional interfaces and callbacks 45
- Conversational interfaces 45

2.3 Connecting components using references and wires 46
- Defining references 47
- Wiring references to services 48
- Wire elements 50
- Automatic wiring 51
- Reference multiplicity 53
- Wiring with different multiplicities 54

2.4 Configuring components using properties 56
- Defining properties 57
- Configuring values for properties 58
- Using complex types for properties 59

2.5 Enabling communication flexibility using bindings 63
- Configuring bindings for services and references 63
- The default binding 65
- Domains, bindings, and wiring 66

2.6 Summary 69

### 3 SCA composite applications 71

3.1 Running a composite application in a single process 72
- Preparing the contributions 73
- Writing the launcher 75
- Running the launcher 76

3.2 Understanding the SCA domain 76
- The domain as a contribution repository 77
- The domain as a naming and visibility boundary 78
- The domain
as an execution environment 79 • Using a single execution node with a local domain 81 • Distributed execution within a domain 83

3.3 Running a distributed composite application 85
Creating an SCA domain 86 • Installing contributions into the domain 87 • Deploying composites for execution 89 • Assigning composites to execution nodes 91
Creating and starting execution nodes 93 • Running the domain manager from a saved configuration 95

3.4 Using SCA composites as application building blocks 96
Different ways of using SCA composites 97 • Using composites as component implementations 97 • Including composites in other composites 101 • Composite reuse in action 103

3.5 Summary 105

PART 2 USING TUSCANY .............................................. 107

4 Service interaction patterns 109
4.1 Understanding the range of SCA service interaction patterns 110
4.2 Remote interaction 112
Configuring remote interaction 113 • Exploiting remote interaction 114
4.3 Local interaction 115
Configuring local interaction 116 • Exploiting local interaction 116
4.4 Request response interaction 118
Configuring request response interaction 118 • Exploiting request response interaction 119
4.5 One-way interaction 119
Configuring one-way interaction 121 • Exploiting one-way interaction 121
4.6 Callback interaction 122
Configuring callback interaction 123 • Exploiting callback interaction 126
4.7 Conversational interaction 127
   Configuring conversational interaction 128 • Exploiting conversational interaction 129

4.8 Summary 130

5 Implementing components using the Java language 132

5.1 Defining a Java component implementation 133

5.2 Using SCA annotations in Java implementations 135

5.3 Services and references with Java interfaces 137
   Identifying local and remote services 137 • Implicit and explicit definition of component interfaces 139 • Interface compatibility and mapping 140 • Transforming messages between interfaces 141 • Pass-by-reference and pass-by-value 142

5.4 Java component services 142
   The @Service annotation 142 • Alternatives to the @Service annotation 143

5.5 Java component references 144
   The @Reference annotation and reference injection 145
   Reference naming 146 • Reference multiplicity 146

5.6 Java component properties 147
   The @Property annotation and property injection 148
   Property naming 149 • Property types 149 • Property value multiplicity 150

5.7 Java component instance creation and scope 151
   Stateless, composite, and conversational scopes 151
   Interacting with component instance creation and destruction 152

5.8 Making callbacks 153
   The credit card security callback scenario 154 • Creating a bidirectional interface with the @Callback annotation 155
   The service programming model for callbacks 155 • The client programming model for callbacks 157 • Getting the callback proxy from the request context 158 • Using callable references to provide callback flexibility 159 • Using a callback ID to identify a specific callback 161 • Redirecting the callback to another service 162

5.9 Holding conversations 163
   Defining and controlling conversations in Java implementations 163 • Starting, using, and stopping
7.3 Connecting component services with binding.sca 203
7.4 Connecting component services with web services 204
   Exposing an SCA service as a web service 205 • Accessing a web service using the SCA Web Services binding 207
   Configuration options for the SCA Web Services binding 209
7.5 Connecting component services with CORBA 211
   Exposing an SCA service as a CORBA service 211
   Accessing a CORBA service using the SCA CORBA binding 214 • Configuration options for the SCA CORBA binding 216
7.6 Connecting component services with RMI 217
   Exposing an SCA service as an RMI service 218 • Accessing an RMI service using the SCA RMI binding 219
   Configuration options for the SCA RMI binding 221
7.7 Connecting component services with JMS 222
   Exposing an SCA service using JMS 222 • Accessing a JMS service using the SCA JMS service binding 225
   Configuration options for the SCA JMS binding 227
7.8 Connecting to EJBs 228
   Exposing an SCA service as an EJB 229 • Accessing an EJB using the SCA EJB binding 229 • Configuration options for the SCA EJB binding 230
7.9 Summary 231

8 Web clients and Web 2.0 232
8.1 Servlets as SCA component implementations 233
   Creating the currency converter user interface using a servlet 233
8.2 Writing web component implementations using JSPs 238
   Exposing the currency converter using a JSP 238
8.3 HTML pages as SCA component implementations 241
   Using an HTML page for the TuscanySCATours user interface 241
8.4 Exposing file system resources 244
   Exposing the TuscanySCATours help pages 245
8.5 Exposing component services as Atom and RSS feeds 246
   Exposing the TuscanySCATours blog as an Atom feed 247
   Extending the TuscanySCATours blog with an RSS feed 250
8.6 Referencing Atom and RSS feeds  252
          Logging the TuscanySCATours blog Atom feed  252
          Logging the TuscanySCATours blog RSS feed  254
8.7 Summary  256

9 Data representation and transformation  257
9.1 Data exchange between SCA components  259
          Using WSDL to describe the CreditCardPayment interface  260
          Using WSDL in an SCA composite  263
9.2 Representing data within component implementations  265
          Passing data to component references using JAXB objects  266
          Accepting data in component services as SDO objects  271
9.3 Describing data contracts within SCA compositions  274
          Specifying contracts on the component type  275
          Specifying contracts on component services and references  277
          Providing contract configuration to bindings  277
9.4 Data transformations  278
          Converting the data coming from the browser from JSON to
          JAXB  279
          Converting from JAXB to AXIOM in order to send a
          SOAP request  280
          Converting from AXIOM to SDO  280
9.5 The Tuscany databinding framework  280
9.6 Summary  282

10 Defining and applying policy  284
10.1 An overview of policy within an SCA domain  285
10.2 The policy runtime  287
          Policy interceptors  287
          The interceptor interface  288
10.3 Using intents and policy sets for implementation policy  289
          Adding implementation intents to the composite file  290
          Choosing a policy set to satisfy the intent  291
10.4 Using intents and policy sets for interaction policy  293
          Adding interaction intents to the composite file  294
          Adding interaction intents to the component implementation  296
          Choosing a policy set to satisfy the intent at the service  298
          Choosing a policy set to satisfy the intent at the reference  300
          Running the payment example with authentication enabled  301
10.5 Other features of the SCA Policy Framework 303
   Dealing with policy sets directly 304 • Profile intents 305
   Intent qualification 305 • Default intents 306
10.6 Tuscany intents and policy sets 306
10.7 Summary 308

PART 3 DEPLOYING TUSCANY APPLICATIONS .......... 309

11 Running and embedding Tuscany 311
11.1 Understanding the Tuscany runtime environment 312
   The SCA domain and Tuscany nodes 312 • Tuscany
   node configuration 313 • Hosting options for a
   Tuscany node 314
11.2 Running Tuscany standalone 315
11.3 Running Tuscany using APIs 317
11.4 Running Tuscany with web applications 319
   Configuring WEB-INF/web.xml 321 • Customizing
   class loading policy 321 • Deploying Tuscany-enabled
   web applications 322
11.5 Configuring distributed nodes 322
   Defining the contents of the domain code repository 323
   Specifying the deployed composites 324 • Defining the
   nodes in the execution cloud 324 • Configuring bindings
   for the nodes in the execution cloud 325
11.6 Embedding Tuscany with a managed container 326
11.7 Summary 328

12 A complete SCA application 329
12.1 Getting ready to run the application 330
12.2 Assembling the travel-booking application 332
   The application user interface (fullapp-ui) 334 • Coordinating
   the application (fullapp-coordination) 336 • Partner services
   (fullapp-packagedtrip and bespoketrip) 337 • Currency
   conversion (fullapp-currency) 341 • Constructing trips
   (fullapp-shoppingcart) 341 • Payment processing
   (payment and creditcard) 344
12.3 The travel-booking application in a distributed domain 346

12.4 Hints and tips for building composite applications 349
  Prototyping and then filling out 349
  Application organization 349
  Developing contributions in a team 351
  Testing contributions in a single VM 351
  Top-down and bottom-up development 352
  Recursive composition 352
  SCA and versioning 353

12.5 Summary 353

PART 4 EXPLORING THE TUSCANY RUNTIME .............. 355

13 Tuscany runtime architecture 357
  13.1 An overview of the Tuscany architecture 358
  13.2 A structural perspective of the Tuscany architecture 359
    Tuscany core functions 360
    Tuscany runtime extension points and plug-ins 362
    Defining extension points and plug-ins 364
  13.3 A behavioral perspective of the Tuscany architecture 367
    Starting and stopping the Tuscany runtime 369
    Loading SCA applications 370
    Building SCA composites 373
    Augmenting the composite with runtime artifacts 374
    Starting and stopping an SCA component 378
    Invoking SCA references and services 379
  13.4 Summary 381

14 Extending Tuscany 382
  14.1 The high-level view of developing a Tuscany extension 383
  14.2 Developing a POJO implementation type 385
    Add the implementation. pojo XML schema 385
    Adding implementation. pojo XML processing 386
    Determining the component type for implementation. pojo 390
    Controlling implementation. pojo invocation and lifecycle 392
    The end-to-end picture for the POJO implementation type 394
    Packaging the POJO implementation type 396
14.3 Developing a new binding type 397

Adding the binding:echo XML schema 398 • Adding the binding:echo XML processor 399 • Controlling binding:echo invocation and lifecycle 402 • The end-to-end picture for the Echo binding type 407

Packaging the echo binding type 409

14.4 Summary 410

appendix A Setting up 411
appendix B What’s next? 418
appendix C OSOA SCA specification license 423
appendix D Travel sample license 425

index 431