

Index

A

- ad hoc paths, 371
- adaptive serializers, 509–512
- addPort method (Service), 102, 104, 110
- Ajax, 463–480
 - front end to SOAShopper case study, 470–479
 - Java EE 5 Web services with, 468–470
- algorithmic type mappings, 15
- Amazon.com SOAP-based services, 434–444
- annotations, 14, 37, 61, 74, 273–278
 - dependency injection, 82–83, 317, 330
 - creating proxy instances, 279–280
 - external annotation files, 197
 - JAXB annotations
 - implementing type mappings with, 224–235
 - for XML messaging, 289–292
 - mapping annotations, 59–62
 - WSDL/Java mapping and, 78
 - processing, 366–367
 - SEI (service endpoint interface), 273–278
- anonymous complex types, mapping, 201
- Apache Ant 1.7.x
 - configuring, 530–531
 - installing, 527
- Apache Maven 2.0.x
 - configuring, 528–530
 - installing, 527
- application frameworks, 6–7

- ASF (Adaptive Serializer Framework), 198
- AsyncHandler class (javax.xml.ws), 297, 301–302
- asynchronous invocation, 50, 297–304
 - callback, 50, 301–304
 - polling, 50, 297–299
 - with proxies, 299–301
- AttachmentMarshaller class, 73
- AttachmentUnmarshaller class, 73
- auto-deployment, 365
- automatic (“drag-and-drop”) deployment, 80, 402–404
- AXIOM object model, 67

B

- BASIC-AUTH authentication, 405–406
- BigInteger class, 201, 208
- binary data encoding, 72–73
- Binder class (javax.xml.bind), 71–72
- binding
 - mapping vs., 195–199
 - standard WSDL/Java mapping, 38, 78, 267–273
- binding declarations, 62–63
- binding language (JAXB 2.0), 62–65, 235–245
 - asynchronous invocation with SEI proxies, 300
- binding runtime framework, 65–69
- BindingProvider interface (javax.xml.ws), 113
- @BindingType annotation, 132
- body, SOAP messages, 146

book example code
 configuring environment for, 84
 installing, 528

Bridge Pattern, 450

C

callback, asynchronous, 50, 301–304

callbacks, marshal events, 71

Castor utility, 198

 client-side invocation with custom
 Java/XML mappings, 292–296

 server-side invocation with custom
 Java/XML mappings, 325–329

client-side invocation, 9–11, 265–310

 asynchronous, 50, 297–304

 callback, 50, 301–304

 polling, 50, 297–299

 with proxies, 299–301

 custom Java/XML mappings,
 292–296

JAX-WS proxies, 46, 265–285

 asynchronous invocation, 299–301

 creating and using, 267

 fault handling with, 282–285

 invoking Web service with, 279–282

 WSDL/Java mapping, 267–273

 SOAP message handlers, 304–309

 steps of, 34–36

 XML messaging, 285–292

client-side JWS support, 34–36

client-side message handlers, 304–309

client-side thread management, 50

component scope for binding
 declarations, 63

configuration files (handler chain files),
 47, 388–389

connect method (`URLConnection`),
 99, 106

create method (`Service`), 280

createDispatch method (`Service`),
 46, 102, 104, 110, 288

D

data transformation

 JAXB 2.0 specification for, 256–261

 XSLT for. *See* XSLT

`DatatypeConverter` class
 (`javax.xml.bind`), 241

definition scope for binding
 declarations, 63

dependency injection, 82–83, 317, 330

 creating proxy instances, 279–280

deployment, 16–18, 359–375

 automatic (“drag-and-drop”),
 80, 402–404

 with custom Java/XML mappings,
 325–329

 with deployment descriptors, 384–402

 ejb-jar.xml descriptor, 372, 390–394

 platform-specific descriptors,
 397–402

 webservices.xml descriptor,
 357–358, 372, 395–397

 web.xml descriptor,
 384–389, 398, 430

EJB endpoints (WSEE), 371–375

 with Endpoint class, 347–355

 REST services, 125–135

 with JWS, 131–135

 without JWS, 126–131

 SOA-J application framework, 514–519

 without deployment descriptors,
 376–384

 WSEE 1.2 and. *See* WSEE 1.2
 specification

deserialization, 11. *See also* serialization

 during invocation, 12–14, 67

 using schema generated beans,
 186–188

deserializers. *See* serializers

details element (SOAP fault
 messages), 283

development modes, 40–42, 44

Dispatch interface (`javax.xml.ws`), 46

 asynchronous polling with, 297–299

- creating Dispatches from Service instances, 46, 102, 104, 110, 288
- pull messaging (HTTP GET), 101
- push messaging (HTTP POST), 110–112
- for XML messaging, 287–291, 296
- dispatching, 8, 151–166
 - document/literal style, 156–159
 - document/literal wrapped style, 51, 159–162
- EJB endpoints (WSEE), 371–375
- JAX-WS 2.0 WSDL/Java mapping workarounds, 166–175
- rpc/literal style, 51, 154–156
 - document/literal styles vs., 159
 - SOAP messages for, 155
 - summary of process, 162–164
 - WSDL port, determining, 151–153
- document/literal style, 156–159
- document/literal wrapped style, 51, 159–162
- Dojo widget. *See* Ajax
- DOM, implement Web service processing with, 323–325
- drag-and-drop (automatic) deployment, 80, 402–404
- dynamic JAX-WS clients, 45–46
- dynamic proxies, 265–266. *See also* proxies, JAX-WS (SEI)

E

- ease-of-use features in JWS, 36–43
- eBay SOAP-based services, 434–444
- EIS (Enterprise Information Systems)
 - getting records from REST with JWS, 101–105
 - getting records from REST without JWS, 98–100
 - sending records to REST with JWS, 110–114
 - sending records to REST without JWS, 105–109

- XML documents and schema for, 88–97
- EJB 3.0 specification, POJO support, 83
- EJB-JAR, packaging servlet endpoints with, 363–365
- EJB endpoints (WSEE), 82
 - deployment and dispatching, 371–375
 - ejb-jar.xml for, 390–394
- ejb-jar.xml descriptor, 372, 390–394
- encoding binary data, 72–73
- encryption with WSEE-deployed services, 407
- Endpoint API (javax.xml.ws), 52–53
 - Java SE deployment with, 347–355
- endpoint listeners, 152, 311–312
- endpoint publishing at runtime, 52–53
- Enterprise Information Systems. *See* EIS
- example code
 - configuring environment for, 84
 - installing, 528
- exceptions (Java)
 - asynchronous callback, 303–304
 - converting to SOAP fault messages, 49
- ExecutionException exception (java.util.concurrent), 304
- Executor interface (java.util.concurrent), 50, 298
- external annotation files, 197
- external bindings files (JAXB), 63–64, 244–245
- external mapping files (Castor), 292–295

F

- fail-fast validation, 70
- fault bean, 334–337
- fault processing, 329, 332–343
 - with SEI proxies, 282–285
- flexible unmarshalling mode. *See* validation
- Future interface (java.util.concurrent), 297, 302

G

- GET requests (HTTP), 29–30, 32
 - Dispatch interface (javax.xml.ws), 101–102
 - URLConnection class (javax.net), 98–100
- getContext method (Response), 298
- getPort method (Service), 280
 - asynchronous callback, 301
- getRequestContext method (BindingProvider interface), 48
- getResponseContext method (BindingProvider interface), 48, 113
- GlassFish server. *See also* deployment
 - automatic (“drag-and-drop”) deployment, 80, 402–404
 - EJB endpoint deployment and dispatching, 372–375
 - servlet endpoint deployment, 366–368
 - wsimport utility, 171, 266, 270–271, 435
- GlassFish server, starting and stopping, 532
- global scope for binding declarations, 62–63
- granularity constraints, 45

H

- handler annotations (WS-Metadata), 79
- handler chains, 47
 - configuration files, 47, 388–389
 - packaging SIBs for deployment, 362, 365
- handler programming model (WSEE), 82
- @HandlerChain annotation, 47, 79. *See also* EJB endpoints (WSEE)
 - client-side handlers, 305
 - server-side handlers, 343–345
- handleResponse method (AsyncHandler), 302–303
- HandlerResolver interface (javax.xml.ws.handler), 305–306

- handlers, JAX-WS, 47, 340–342
 - client-side message handlers, 304–309
 - server-side message handlers, 343–347
- header blocks, SOAP, 146
- HTTP binding, 49, 132
- HTTP transport for SOAP messages, 29–30
- HttpServlet class, deploying RESTful services, 127–128
- URLConnection class (javax.net)
 - pull messaging (HTTP GET), 98–100
 - push messaging (HTTP POST), 105–107

I

- interceptors, 83
- invocation, 8–11
 - asynchronous, 50, 297–304
 - callback, 50, 301–304
 - polling, 50, 297–299
 - with proxies, 299–301
 - with Java proxies, 279–282. *See also* proxies, JAX-WS (SEI)
- JAXB binding runtime framework and, 66–67
- mapping WSDL/SOAP to. *See* dispatching
- SEI proxies, 46, 265–285
 - asynchronous invocation, 299–301
 - creating and using, 267
 - fault handling with, 282–285
 - invoking Web service with, 279–282
 - WSDL/Java mapping, 267–273
- serialization during, 12–14, 67
- server-side, 8–9, 11, 311–356
 - deployment with custom Java/XML mappings, 325–329
 - deployment with Endpoint class, 347–355
 - fault processing, 329, 332–343
 - JAX-WS architecture, 311–316
 - providers and XML processing without JAXB, 320–325

- server-side handlers, 343–347
 - “Start from WSDL” with SEI, 316–320
 - steps of, 30–34, 312–315
 - validation, 329–332
 - SOA-J application framework, 493–503
 - with XML, 46
 - invocation, client-side, 9–11, 265–310
 - asynchronous, 50, 297–304
 - callback, 50, 301–304
 - polling, 50, 297–299
 - with proxies, 299–301
 - custom Java/XML mappings, 292–296
 - JAX-WS proxies. *See* proxies, JAX-WS (SEI)
 - SOAP message handlers, 304–309
 - steps of, 34–36
 - XML messaging, 285–292
 - InvocationHandler class
 - (`java.lang.reflect`), 266
 - invoke method (Dispatch), 102, 110, 112
 - for XML messaging, 289
 - invoke method (Provider), 133, 320–321
 - invokeAsync method (Dispatch), 298
- J**
- J2EE programming model, 357
 - Java, starting with (development mode), 40–42
 - Java EE 5 platform, 82–84
 - Ajax with, 468–470
 - JWS standards. *See* JSRs
 - SDK installation, 526–527
 - Java exceptions
 - asynchronous callback, 303–304
 - converting to SOAP fault messages, 49
 - Java Interface Proxies, 46
 - Java RMI, 265
 - Java SE 6 platform, installing, 535
 - Java value classes, 57–58, 205–206
 - Java/WSDL mapping, 38, 267–273
 - mapping annotations and, 78
 - Java/XML binding or mapping, 54–59, 137
 - binding versus mapping, 196–197
 - Castor for. *See* Castor utility
 - custom, client-side invocation with, 292–296
 - custom, server-side invocation with, 325–329
 - JAX-WS 2.0 limitations, 180–181
 - JAXB standard for, overview of, 199–209
 - limitations, working around, 182–194
 - dispatching, 166–175
 - using schema compiler, 182–189
 - using schema generator, 189–194
 - mapping annotations, 59–62, 78
 - role of, 178–179
 - `java.lang.reflect.InvocationHandler` class, 266
 - `java.math.BigInteger` class, 201, 208
 - `java.net.URL` class
 - pull messaging (HTTP GET), 98–100
 - push messaging (HTTP POST), 105–107
 - `java.util.concurrent.ExecutionException` exception, 304
 - `java.util.concurrent.Executor` class, 50, 298
 - `java.util.concurrent.Future` interface, 297, 302
 - `java.util.concurrent.Response` interface, 297
 - `javax.jws.soap` package, 78
 - `javax.jws.SOAPBinding` annotation, 51
 - `java.xml.bind.JAXBElement` class, 180, 222–223
 - `javax.net.HttpURLConnection` class
 - pull messaging (HTTP GET), 98–100
 - push messaging (HTTP POST), 105–107
 - `javax.wms` package, 78
 - `javax.xml.bind.annotation.XmlAdapter` class, 245–256
 - `javax.xml.bind.Binder` class, 71–72

- javax.xml.bind.DatatypeConverter class, 241
- javax.xml.soap.SOAPFault class, 284, 338–339
- javax.xml.transform.Transformer class, 216
- javax.xml.validation.Schema class, 216
- javax.xml.ws.AsyncHandler class, 297, 301–302
- javax.xml.ws.BindingProvider interface, 113
- javax.xml.ws.Dispatch class, 46
 - asynchronous polling with, 297–299
 - creating Dispatches from Service instances, 46, 102, 104, 110, 288
 - pull messaging (HTTP GET), 101
 - push messaging (HTTP POST), 110–112
 - for XML messaging, 287–291, 296
- javax.xml.ws.Endpoint API, 52–53
 - Java SE deployment with, 347–355
- javax.xml.ws.handler.HandlerResolver interface, 305–306
- javax.xml.ws.MessageContext class, 48
- javax.xml.ws.Provider interface, 46–47, 131–133
 - server-side invocation, 320–325
- javax.xml.ws.Service class, 34, 45
 - creating Dispatches, 46, 102, 104, 110, 288
 - pull messaging (HTTP GET), 101–102
 - push messaging (HTTP POST), 110–112
- javax.xml.ws.soap.SOAPFaultException exception, 282, 337–338
- javax.xml.ws.WebServiceContext interface, 317, 330
- javax.xml.ws.WebServiceException exception, 340–342
- JAX-RPC 1.1 specification, 40
 - JAX-WS 2.0 improvements over, 5, 54
- JAX-WS 2.0, 524
- JAX-WS 2.0 specification, 5, 43–53
 - annotations. *See also* source code annotations
 - asynchronous, 50, 297–304
 - callback, 50, 301–304
 - polling, 50, 297–299
 - with proxies, 299–301
 - client-side development, 9–11, 265–310
 - asynchronous. *See* asynchronous invocation
 - custom Java/XML mappings, 292–296
 - proxies. *See* proxies, JAX-WS (SEI) SOAP message handlers, 304–309
 - steps of, 34–36
 - XML messaging, 285–292
 - dispatching, 151–166
 - determining WSDL port, 151–153
 - document/literal style, 156–159
 - document/literal wrapped style, 159–162
 - rpc/literal style, 154–156
 - shortcomings of, 165–166
 - working around limitations of, 166–175
 - WS-I Basic Profile, 153–154
 - dynamic and static clients, 45–46
 - feature map, 43
 - SEI proxies, 46, 265–285
 - asynchronous invocation, 299–301
 - creating and using, 267
 - fault handling with, 282–285
 - invoking Web service with, 279–282
 - WSDL/Java mapping, 267–273
 - server-side development, 8–9, 11, 311–356
 - deployment with custom Java/XML mappings, 325–329
 - deployment with Endpoint class, 347–355
 - fault processing, 329, 332–343
 - JAX-WS architecture, 311–316
 - providers and XML processing without JAXB, 320–325

- server-side handlers, 343–347
 - “Start from WSDL” with SEI, 316–320
 - steps of, 30–34, 312–315
 - validation, 329–332
 - WSDL/Java mapping, 38, 267–273
 - mapping annotations and, 78
 - JAX-WS Service Endpoints, 360
 - JAXB 2.0 specification, 54–73, 195–263, 524
 - annotations. *See also* source code annotations
 - binding versus mapping, 195–199
 - bypassing serialization process, 46
 - for data transformation, 256–261
 - feature map, 57
 - implementing type mappings, 209–217
 - with annotations, 224–235
 - with binding language, 235–245
 - recursively, 217–224
 - with XmlAdapter class, 245–256
 - serialization rules, 39
 - standard Java/XML binding, 199–209.
 - See also* Java/XML binding
 - jaxb:class declaration, 237–238
 - JAXBElement class (java.xml.bind), 180, 222–223
 - jaxb:globalBindings declaration, 236–237
 - jaxb:javaType declaration, 239–240
 - jaxb:package declarations, 201
 - jaxb:property declarations, 207
 - jaxb:schemaBindings declaration, 236–237
 - jax:dom declarations, 65
 - JAXP (Java API of XML Processing), 121–125
 - JAXWSServlet class, 367–368
 - JSR-181 processor, 79
 - JSRs (Java Web Services standards), 30–36
 - client-side support, 34–36
 - server-side support, 30–34
 - JWS (Java Web Services), 25–84
 - ease-of-use features, 36–43
 - Java EE 5 platform, impact of, 82–84
 - JAX-WS 2.0. *See* JAX-WS 2.0 specification
 - JAXB. *See* JAXB 2.0 specification
 - in SOA development, 26–36
 - WS-Metadata, 31, 73–80, 357, 358
 - annotations, 37, 74–75, 78–79. *See also* source code annotations
 - auto-deployment, 365
 - feature map, 76
 - WSEE 1.2 specification, 31, 80–82, 357–358
 - encryption with, 407
 - feature map, 81
- L**
- local element declarations, 201
 - logical handlers, 47, 344–345
- M**
- mapping annotations, 59–62
 - WSDL/Java mapping and, 78
 - mapping, binding vs., 195–199
 - Mapping class (Castor), 295
 - mapping strategies, 14–15. *See also* type mappings
 - mapping WSDL to/from Java, 38, 267–273
 - mapping annotations and, 78
 - marshalling (unmarshalling), JAXB, 65–69
 - annotated classes, 233–234
 - event callbacks, 71
 - with schema generated beans, 187–188
 - validation and, 69
 - Maven
 - configuring, 528–530
 - installing, 527
 - message context, 48
 - message handlers (SOAP), 304–309
 - Message mode (Provider interface), 321
 - Message Payload mode (Provider interface), 321

MessageContext class (javax.xml.ws), 48
 MTOM standard, 66, 67, 72–73
 multivariate type mappings, 245. *See also*
 XmlAdapter class

N

named complex types, mapping, 208
 namespace prefix definitions, 537

O

OASIS XML Catalogs 1.1 specification,
 51, 407–409
 one-way operations (WSDL), 50
 openConnection method (URL), 99, 105
 Order Management Service (example),
 26–29

P

packaging Web services, 359–375. *See also*
 deployment
 parameter style (WSDL style), 152–153
 parseMethod attribute (jaxb:javaType),
 240–242
 partial binding of XML documents,
 71–72
 PAYLOAD service mode, 288
 platform-specific deployment
 descriptors, 397–402
 POJO support, 83
 POJOs, annotating, 225
 polling, 50, 297–299
 port components, 30–31, 81, 359
 port, WSDL, 151–153
 portability, JAXB, 70
 POST requests (HTTP), 29–30
 HttpURLConnection class (javax.net),
 105–107
 printMethod attribute (jaxb:javaType),
 240–242
 protocol handlers, 47, 344–345
 Provider interface (javax.xml.ws),
 46–47, 131–133

server-side invocation, 320–325
 proxies, JAX-WS (SEI), 46, 265–285
 asynchronous invocation, 299–301
 creating and using, 267
 fault handling with, 282–285
 invoking Web service with, 279–282
 WSDL/Java mapping, 267–273
 pseudo reference passing, 52
 publish method (Endpoint), 349
 publishing endpoints and runtime, 52–53
 publishing WSDL, 16–17

R

raw XML for messaging, 286–289
 recursive framework for type mappings,
 217–224
 reference passing, 52
 Representational State Transfer. *See*
 REST paradigm
 request handlers, JAX-WS, 47
 request message (SOAP), example of,
 146–147
 in document/literal style, 158–159
 in document/literal wrapped style, 162
 in rpc/literal style, 156
 @RequestWrapper annotation,
 173, 273–275, 277
 @Resource annotation, 317
 response handlers, JAX-WS, 47
 Response interface
 (java.util.concurrent), 297
 response message (SOAP), example of,
 147–148
 @ResponseWrapper annotation,
 173, 275, 278
 REST paradigm, 85–136
 client-side services consumption,
 98–114
 getting EIS records with JWS,
 101–105
 getting EIS records without JWS,
 98–100

- sending EIS records with JWS, 110–114
- sending EIS records without JWS, 105–109
- deployment of, 125–135
 - with JWS, 131–135
 - without JWS, 126–131
- IDL for XML over HTTP without SOAP, 139
- server-side invocation, 315
- SOAP vs., 87, 145–146, 259
- SOAShopper services, 423–431, 444–449
- roles, security, 406
- rpc/literal style, 51, 154–156
 - document/literal styles vs., 159
 - SOAP messages for, 155
- RPC messages, RESTful services vs., 86
- rule-based type mappings, 15
- run-time architecture for endpoint, 311–312
- runtime endpoint publishing, 52–53

S

- SAAJ interface, 67
 - fault processing, 337–339
- SAX (Simple API for XML), 36
- Schema class (javax.xml.validation), 216
- schema compiler, 57, 182–189, 196
- schema generator, 59, 59–60, 189–194, 196
- schema scope for binding declarations, 63
- scope, binding declarations, 62–63
- security, 405–407
 - SOAP message handlers, 304–305
- SEI (service endpoint interface), 9, 34–35, 45, 265, 359–360
 - annotations, 273–278
 - deploying SIB without descriptors, 378–381
 - with Endpoint class, 347–348
 - proxies, 46, 265–285
 - asynchronous invocation, 299–301
 - creating and using, 267
 - fault handling with, 282–285
 - invoking Web service with, 279–282
 - WSDL/Java mapping, 267–273
 - server-side invocation, 316–320
- separation of concerns, 256
- serialization, 11–16, 195
 - binary data encoding, 72–73
 - during invocation, 12–14, 67
 - mapping annotations for, 60
 - recursive, 217–224
 - SOA-J application framework, 503–514
 - validation and. *See* validation
- serialization context, 14, 16, 39–40
 - SOA-J application framework, 485
- Serializer interface, 219–220
- serializers, 218–224
- server-side invocation, 8–9, 11, 311–356
 - deployment with custom Java/XML mappings, 325–329
 - deployment with Endpoint class, 347–355
 - fault processing, 329, 332–343
 - JAX-WS architecture, 311–316
 - providers and XML processing without JAXB, 320–325
 - server-side handlers, 343–347
 - “Start from WSDL” with SEI, 316–320
 - steps of, 30–34, 312–315
 - validation, 329–332
- Service class (java.xml.ws), 34, 45
 - creating Dispatches, 46, 102, 104, 110, 288
 - pull messaging (HTTP GET), 101–102
 - push messaging (HTTP POST), 110–112
- service-oriented architecture. *See* SOA
- service endpoint interface. *See* SEI
- service implementation bean. *See* SIB
- ServiceDeployment class, 139
- @ServiceMode annotation, 322
- servlet endpoints (WSEE), 32, 81
 - deployment, 367
 - packaging using WARs, 361–363
 - web.xml for, 384–389

- setAttachmentMarshaller method
(Marshaller), 73
- SetAttachmentUnmarshaller method
(Unmarshaller), 73
- setContentTypes method
(HttpServletResponse), 129
- setHandlerResolver method
(Service), 305
- setMapping method (Mapping), 295
- setMetadata method (Endpoint API),
53, 349
- setSchema method (Marshaller or
Unmarshaller), 208–209
- SFJ. *See* “Start from Java” development
mode
- SFWJ. *See* “Start from WSDL and Java”
development mode
- SIB (service implementation bean),
38, 79, 169–170, 359
 - deploying without descriptors,
376–378
 - packaging SIBs for deployment,
361–364
 - Provider usage modes, 321–322
 - validation. *See* validation
- simplified packaging, WSEE, 82
- SOA
 - data transformation, 115
 - example of, 26–29
 - need for SOAP, 145–146
 - need for WSDL, 108–109, 138–139
 - role of XML Schema, 91–97
 - schema libraries, 139–141
 - suitability of JAXB, 195–199
 - type mappings, 178–179
 - WSDL-centric framework, 481–482
- SOA-J application framework,
7, 481–521
 - architecture, 483–486
 - building and deploying, 535
 - deployment subsystem, 514–519
 - invocation subsystem, 493–503
 - serialization subsystem, 503–514
 - WSDL-centric development with,
486–493
- SOA Integration
 - definition, 137
 - SOAShopper example architecture,
412–415
 - start from WSDL and Java,
56, 175–177
 - using the Bridge pattern, 450–459
- SOAP, 145–151
 - binary data encoding of messages,
72–73
 - dispatching messages. *See* dispatching
fault messages
 - converting Java exceptions to, 49
 - examples of, 148–150
 - handling with proxies, 282–285
 - mapping to Java invocation. *See*
dispatching
 - message handlers, 304–309
 - REST vs., 145–146, 259
 - SOAShopper services,
417–423, 434–444
 - Version 1.1 standard, 524
- SOAP binding annotations
(WS-Metadata), 78
- SOAP binding for message
processing, 48
- SOAP/JMS deployment, 315
- SOAP messages, 29
 - examples of, 146–150
 - fault. *See* SOAP, fault messages
 - transport protocols for, 29–30
 - validating. *See* validation
- SOAP services, REST services vs., 87
- soap:address element, 151–152, 377,
380, 382, 386–387
 - identifying (dispatching process), 163
- @SOAPBinding annotation, 51, 76, 78
- SOAPFault class (javax.xml.soap),
284, 338–339
- SOAPFaultException exception
(javax.xml.ws.soap), 282, 337–338
- @SOAPMessageHandlers annotation, 79

- SOAShopper case study, 411–461
 - Ajax front-end, 470–479
 - API and Integration Layer, 450–460
 - building and deploying, 534–535
 - eBay and Amazon Services (SOAP), 434–444
 - overview of, 411–416
 - RESTful services and standard XML schema, 423–431
 - service implementation, 431–434
 - SOAP services, 417–423
 - Yahoo! Services (REST), 444–449
 - source code annotations, 14, 37, 61, 74
 - dependency injection, 82–83, 317, 330
 - creating proxy instances, 279–280
 - external annotation files, 197
 - JAXB annotations
 - implementing type mappings with, 224–235
 - for XML messaging, 289–292
 - mapping annotations, 59–62, 78
 - WSDL/Java mapping and, 78
 - processing, 366–367
 - SEI (service endpoint interface), 273–278
 - standard binding, 14
 - standard serialization context, 14, 16, 39–40
 - SOA-J application framework, 485
 - standard WSDL/Java mapping, 38, 267–273
 - mapping annotations and, 78
 - “Start from Java” development mode, 40, 42, 44, 74
 - implementing type mappings, 224–235
 - Java/XML binding and, 59
 - Java/XML bindings and, 56
 - “Start from WSDL” development mode, 40–42, 44
 - server-side invocation, 316–320
 - “Start from WSDL and Java” development mode, 41–42, 175–181
 - annotation-centric design and, 61, 74–75
 - Java/XML binding and, 58–59
 - JAX-WS 2.0 limitations, 175
 - role of Java/XML mapping, 175–181
 - SOAShopper case study, 417
 - WS-Metadata annotations and, 79–80
 - “Start from XML and Java” development mode, 56
 - “Start from XML Schema” development mode
 - implementing type mappings, 235–245
 - Java/XML bindings and, 56
 - @Stateless annotation, 83, 360
 - stateless session bean. *See* EJB endpoints (WSEE)
 - static JAX-WS clients, 45–46
 - static WSDL, 45
 - style attribute (WSDL style), 152–153
 - sun-ejb-jar.xml descriptor, 372, 400–402
 - sun-web.xml descriptor, 398–399
 - Symmetric HTTPS, 405, 407
- T**
- thread control, client-side, 50
 - tools, JWS as, 6–7
 - trade-offs, JWS, 36–43
 - Transformer class
 - (javax.xml.transform), 216
 - type mappings, 14–15, 54
 - binding tools and, 196
 - implementing with JAXB.
 - See* JAXB 2.0 specification
 - Java/XML. *See* Java/XML binding recursive framework for, 217–224
 - schema compiler, 57, 182–189, 196
 - schema generator,
 - 59, 59–60, 189–194, 196
 - SFWJ development and, 61
 - WSDL/Java mapping, 38, 78, 267–273
 - XML/HTTP, 49, 132

U

- unmarshal method (XmlAdapter), 250
- unmarshalling. *See* marshalling
 - (unmarshalling), JAXB
- URL class (java.net)
 - pull messaging (HTTP GET), 98–100
 - push messaging (HTTP POST), 105–107
- use attribute (WSDL style), 152–153

V

- validation, 69–70, 329–332
 - server-side handlers for, 345–347
- value classes, 57–58, 205–206

W

- waitfor task, 404
- WARs, packaging servlet endpoints with, 361–363
- Web Services, as hard to learn, 2–7
- Web Services Addressing (WS-ADDRESSING), 152
- Web services deployment descriptors, 360, 384–402
- Web Services Platform Architecture. *See* WSPA
- Web services security, 405–407
 - SOAP message handlers, 304–305
- @WebFault annotation, 49
- @WebMethod annotation, 173, 277
 - operationName element, 77, 79
- @WebParam annotation, 275, 278
 - name element, 78
- @WebResult annotation, 173, 275, 277
 - name element, 78
- @WebService annotation, 40, 41, 76, 173, 277, 316–318, 348
- WebServiceContext interface (javax.xml.ws), 317, 330
- WebServiceException exception (javax.xml.ws), 340–342
- @WebServiceProvider annotation, 131–132, 321–323
- @WebServiceRef annotation, 83, 279–280
- webservicex.xml descriptor, 357–358, 372, 395–397
- web.xml descriptor, 384–389, 398
 - SOAShopper REST endpoint deployment, 430
- wrappers
 - document/literal wrapped style, 51, 159–162
 - for JAXB-generated code, 39
 - “Start from WSDL and Java” development and, 75
- writeTo method (SOAPMessage), 309
- WS-ADDRESSING 1.0 specification, 152
- WS-I Basic Profile, 153–154
- WS-Metadata 2.0 specification, 31, 73–80, 357, 358
 - annotations, 37, 74–75, 78–79. *See also* source code annotations
 - auto-deployment, 365
 - feature map, 76
- WS-Security standard, 407
- WSDL, 138–145, 360, 523
 - example of, 141–145
 - mapping to Java invocation. *See* dispatching
 - mapping to Java targets, 44–45
 - one-way operations, 50
 - publishing, 16–17
 - SOA-J application framework, 7, 481–521
 - architecture, 483–486
 - building and deploying, 535
 - deployment subsystem, 514–519
 - invocation subsystem, 493–503
 - serialization subsystem, 503–514
 - WSDL-centric development with, 486–493
 - starting with (as development mode), 40–42
 - static, 45

- WSDL artifacts, deploying with, 381–384
 - WSDL documents, 29
 - WSDL faults
 - converting Java exceptions to, 49
 - inputMessageValidationFault fault, 143, 149
 - WSDL/Java mapping, 38, 267–273
 - mapping annotations and, 78
 - WSDL Mapping Annotations (WS-Metadata), 78
 - WSDL port, determining, 151–153
 - WSDL styles, 50–51, 152–153
 - document/literal style, 156–159
 - document/literal wrapped style, 51, 159–162
 - rpc/literal style, 51, 154–156
 - document/literal styles vs., 159
 - SOAP messages for, 155
 - wsdl:binding element, 143–144
 - wsdl:fault element, 332–334, 340
 - fault handling with proxies, 282–285
 - JAX-WS WSDL/Java mapping, 271–273
 - wsdl:input element, 271
 - wsdl:message element, 142–143
 - in document/literal style, 157–158
 - in document/literal wrapped style, 160–161
 - JAX-WS WSDL/Java mapping, 271–273
 - in rpc/literal style, 155–156
 - wsdl:operation, 157
 - wsdl:operation element, 152, 276
 - asynchronous invocation. *See* asynchronous invocation
 - identifying (dispatching process), 163
 - JAX-WS WSDL/Java mapping, 271
 - wsdl:output element, 271
 - wsdl:port element, 151–152
 - deploying SIB without descriptors, 377, 380
 - obtaining (dispatching process), 163
 - wsdl:portType element, 142–143
 - deploying SIB without descriptors, 376, 379
 - JAX-WS WSDL/Java mapping, 267–273
 - obtaining (dispatching process), 163
 - shortcomings of JAX-WS 2.0
 - dispatching, 165–175
 - wsdl:service element, 144–145
 - wsdl:serviceName element, 376, 380
 - wsdl:types element, 141–142
 - in document/literal style, 157–158
 - in document/literal wrapped style, 160–161
 - JAX-WS WSDL/Java mapping, 271, 273
 - in rpc/literal style, 155–156
 - WSEE 1.2 specification, 31, 80–82, 357–358
 - encryption with, 407
 - feature map, 81
 - WSIAP standard, 72–73
 - wsimport utility, 171, 266, 270–271, 435
 - WSPA (Web Services Platform Architecture), 8–18
- X**
- XML, invocation with, 46
 - XML annotations. *See* binding language (JAXB 2.0)
 - XML Catalogs, 51–52
 - XML documents
 - with REST services, 97–114
 - getting EIS records with JWS, 101–105
 - getting EIS records without JWS, 98–100
 - sending EIS records with JWS, 110–114
 - sending EIS records without JWS, 105–109
 - XML/HTTP binding, 49, 132
 - XML/Java mappings. *See* Java/XML binding

- XML messaging, 285–292
 - with custom annotated JAXB classes, 289–292
 - with raw XML, 286–289
 - XML Schema Version 1.0, 524
 - XML schemas, extracting, 330–332
 - XML service providers, 46–47
 - @XmlAccessorType annotation, 183, 204, 228–229, 230
 - XmlAdapter class
 - (`javax.xml.bind.annotation`), 245–256
 - @XmlAttachmentRef annotation, 73
 - @XmlAttribute annotation, 208, 233
 - @XmlElement annotation, 201, 204, 227, 231
 - @XmlElementWrapper annotation, 228
 - @XmlJavaTypeAdapter annotation, 61–62, 241, 245, 247–250, 255–256
 - @XmlMimeType annotation, 72
 - @XmlRootElement annotation, 55, 204, 227
 - @XmlSchema annotation, 229–230
 - namespace element, 201
 - @XmlType annotation, 204, 231, 233
 - XPath
 - for external bindings files, 64
 - for XSLT data transformations, 117
 - `xsl:apply-templates` instructions, 119–120
 - XSLT (Extensible Stylesheet Language Transformations), 114–125
 - versus JAXB 2.0, for data transformation, 256
 - processing in Java. *See* JAXP
 - schema generator and, 189–194
 - Version 1.0 standard, 524
 - `xs:positiveInteger` mapping, 201
 - `xs:schema` nodes, extracting, 330–332
- Y**
- Yahoo! RESTful services, 444–449