

Eric Jendrock • Ian Evans • Devika Gollapudi
Kim Haase • Chinmayee Srivathsa



The Java EE 6 Tutorial

Basic Concepts

Fourth Edition

*The Java Series
Enterprise Edition*



... from the Source

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/ph

Library of Congress Cataloging-in-Publication Data

The Java EE 6 tutorial : basic concepts / Eric Jendrock ... [et al.]. --
4th ed.

p. cm.

Includes index.

ISBN 0-13-708185-5 (pbk. : alk. paper)

1. Java (Computer program language) 2. Application program interfaces
(Computer software) 3. Application software—Development. 4. Internet
programming. I. Jendrock, Eric.

QA76.73.J38J3652 2010

006.7'6--dc22

2010025759

Copyright © 2011, Oracle and/or its affiliates. All rights reserved.
500 Oracle Parkway, Redwood Shores, CA 94065

Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

ISBN-13: 978-013-708185-1

ISBN-10: 0-137-08185-5

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
First printing, August, 2010

Contents

Preface	xxi
 Part I Introduction	1
 1 Overview	3
Java EE 6 Platform Highlights	4
Java EE Application Model	5
Distributed Multitiered Applications	6
Security	7
Java EE Components	8
Java EE Clients	8
Web Components	10
Business Components	11
Enterprise Information System Tier	12
Java EE Containers	13
Container Services	13
Container Types	14
Web Services Support	15
XML	15
SOAP Transport Protocol	16
WSDL Standard Format	16
Java EE Application Assembly and Deployment	17
Packaging Applications	17
Development Roles	19
Java EE Product Provider	20
Tool Provider	20
Application Component Provider	20

Application Assembler	21
Application Deployer and Administrator	21
Java EE 6 APIs	22
Enterprise JavaBeans Technology	25
Java Servlet Technology	26
JavaServer Faces Technology	26
JavaServer Pages Technology	27
JavaServer Pages Standard Tag Library	27
Java Persistence API	28
Java Transaction API	28
Java API for RESTful Web Services	28
Managed Beans	28
Contexts and Dependency Injection for the Java EE Platform (JSR 299)	29
Dependency Injection for Java (JSR 330)	29
Bean Validation	29
Java Message Service API	29
Java EE Connector Architecture	29
JavaMail API	30
Java Authorization Contract for Containers	30
Java Authentication Service Provider Interface for Containers	30
Java EE 6 APIs in the Java Platform, Standard Edition 6.0	31
Java Database Connectivity API	31
Java Naming and Directory Interface API	31
JavaBeans Activation Framework	32
Java API for XML Processing	32
Java Architecture for XML Binding	33
SOAP with Attachments API for Java	33
Java API for XML Web Services	33
Java Authentication and Authorization Service	33
GlassFish Server Tools	34
2 Using the Tutorial Examples	37
Required Software	37
Java Platform, Standard Edition	37
Java EE 6 Software Development Kit	38

Java EE 6 Tutorial Component	38
NetBeans IDE	40
Apache Ant	41
Starting and Stopping the GlassFish Server	41
Starting the Administration Console	42
▼ To Start the Administration Console in NetBeans IDE	43
Starting and Stopping the Java DB Server	43
▼ To Start the Database Server Using NetBeans IDE	43
Building the Examples	44
Tutorial Example Directory Structure	44
Getting the Latest Updates to the Tutorial	44
▼ To Update the Tutorial Through the Update Center	45
Debugging Java EE Applications	45
Using the Server Log	45
Using a Debugger	46
Part II The Web Tier	47
3 Getting Started with Web Applications	49
Web Applications	50
Web Application Lifecycle	51
Web Modules: The hello1 Example	53
Examining the hello1 Web Module	54
Packaging a Web Module	57
Deploying a Web Module	59
Running a Deployed Web Module	59
Listing Deployed Web Modules	60
Updating a Web Module	60
Dynamic Reloading	60
Undeploying Web Modules	61
Configuring Web Applications: The hello2 Example	62
Mapping URLs to Web Components	62
Examining the hello2 Web Module	63
Building, Packaging, Deploying, and Running the hello2 Example	64
Declaring Welcome Files	66

Setting Context and Initialization Parameters	66
Mapping Errors to Error Screens	67
Declaring Resource References	68
Further Information about Web Applications	71
4 JavaServer Faces Technology	73
What Is a JavaServer Faces Application?	74
JavaServer Faces Technology Benefits	75
Creating a Simple JavaServer Faces Application	77
Developing the Backing Bean	77
Creating the Web Page	78
Mapping the FacesServlet Instance	78
The Lifecycle of the hello Application	79
▼ To Build, Package, Deploy, and Run the Application in NetBeans IDE	80
Further Information about JavaServer Faces Technology	81
5 Introduction to Facelets	83
What Is Facelets?	83
Developing a Simple Facelets Application	85
Creating a Facelets Application	85
Configuring the Application	88
Building, Packaging, Deploying, and Running the guessnumber Facelets Example	89
Templating	91
Composite Components	94
Resources	96
6 Expression Language	99
Overview of the EL	99
Immediate and Deferred Evaluation Syntax	100
Immediate Evaluation	101
Deferred Evaluation	101
Value and Method Expressions	102
Value Expressions	102
Method Expressions	106

Defining a Tag Attribute Type	108
Literal Expressions	109
Operators	111
Reserved Words	111
Examples of EL Expressions	112
 7 Using JavaServer Faces Technology in Web Pages	 113
Setting Up a Page	113
Adding Components to a Page Using HTML Tags	114
Common Component Tag Attributes	117
Adding HTML Head and Body Tags	119
Adding a Form Component	120
Using Text Components	121
Using Command Component Tags for Performing Actions and Navigation	126
Adding Graphics and Images with the <code>h:graphicImage</code> Tag	127
Laying Out Components with the <code>h:panelGrid</code> and <code>h:panelGroup</code> Tags	128
Displaying Components for Selecting One Value	130
Displaying Components for Selecting Multiple Values	132
Using the <code>f:selectItem</code> and <code>f:selectItems</code> Tags	133
Using Data-Bound Table Components	135
Displaying Error Messages with the <code>h:message</code> and <code>h:messages</code> Tags	138
Creating Bookmarkable URLs with the <code>h:button</code> and <code>h:link</code> Tags	139
Using View Parameters to Configure Bookmarkable URLs	140
Resource Relocation Using <code>h:output</code> Tags	141
Using Core Tags	143
 8 Using Converters, Listeners, and Validators	 145
Using the Standard Converters	145
Converting a Component's Value	146
Using <code>DateTimeConverter</code>	147
Using <code>NumberConverter</code>	149
Registering Listeners on Components	151
Registering a Value-Change Listener on a Component	151
Registering an Action Listener on a Component	152
Using the Standard Validators	152

Validating a Component's Value	153
Using LongRangeValidator	154
Referencing a Backing Bean Method	154
Referencing a Method That Performs Navigation	155
Referencing a Method That Handles an Action Event	156
Referencing a Method That Performs Validation	156
Referencing a Method That Handles a Value-Change Event	156
 9 Developing with JavaServer Faces Technology	159
Backings Beans	159
Creating a Backing Bean	160
Using the EL to Reference Backing Beans	161
Writing Bean Properties	162
Writing Properties Bound to Component Values	163
Writing Properties Bound to Component Instances	168
Writing Properties Bound to Converters, Listeners, or Validators	170
Writing Backing Bean Methods	170
Writing a Method to Handle Navigation	171
Writing a Method to Handle an Action Event	172
Writing a Method to Perform Validation	173
Writing a Method to Handle a Value-Change Event	173
Using Bean Validation	174
Validating Null and Empty Strings	177
 10 Java Servlet Technology	179
What Is a Servlet?	180
Servlet Lifecycle	180
Handling Servlet Lifecycle Events	180
Handling Servlet Errors	182
Sharing Information	182
Using Scope Objects	182
Controlling Concurrent Access to Shared Resources	183
Creating and Initializing a Servlet	183
Writing Service Methods	184
Getting Information from Requests	185

Constructing Responses	186
Filtering Requests and Responses	187
Programming Filters	187
Programming Customized Requests and Responses	188
Specifying Filter Mappings	189
Invoking Other Web Resources	191
Including Other Resources in the Response	192
Transferring Control to Another Web Component	192
Accessing the Web Context	193
Maintaining Client State	193
Accessing a Session	193
Associating Objects with a Session	193
Session Management	194
Session Tracking	195
Finalizing a Servlet	195
Tracking Service Requests	196
Notifying Methods to Shut Down	196
Creating Polite Long-Running Methods	197
The mood Example Application	198
Components of the mood Example Application	198
Building, Packaging, Deploying, and Running the mood Example	198
Further Information about Java Servlet Technology	200
Part III Web Services	201
11 Introduction to Web Services	203
What Are Web Services?	203
Types of Web Services	203
“Big” Web Services	204
RESTful Web Services	204
Deciding Which Type of Web Service to Use	206
12 Building Web Services with JAX-WS	207
Creating a Simple Web Service and Clients with JAX-WS	208

	Requirements of a JAX-WS Endpoint	209
	Coding the Service Endpoint Implementation Class	210
	Building, Packaging, and Deploying the Service	210
	Testing the Methods of a Web Service Endpoint	211
	A Simple JAX-WS Application Client	212
	A Simple JAX-WS Web Client	214
	Types Supported by JAX-WS	217
	Web Services Interoperability and JAX-WS	217
	Further Information about JAX-WS	217
13	Building RESTful Web Services with JAX-RS	219
	What Are RESTful Web Services?	219
	Creating a RESTful Root Resource Class	220
	Developing RESTful Web Services with JAX-RS	221
	Overview of a JAX-RS Application	222
	The @Path Annotation and URI Path Templates	223
	Responding to HTTP Resources	226
	Using @Consumes and @Produces to Customize Requests and Responses	229
	Extracting Request Parameters	231
	Example Applications for JAX-RS	235
	A RESTful Web Service	235
	The rsvp Example Application	237
	Real-World Examples	240
	Further Information about JAX-RS	240
Part IV	Enterprise Beans	243
14	Enterprise Beans	245
	What Is an Enterprise Bean?	245
	Benefits of Enterprise Beans	246
	When to Use Enterprise Beans	246
	Types of Enterprise Beans	246
	What Is a Session Bean?	247
	Types of Session Beans	247

When to Use Session Beans	248
What Is a Message-Driven Bean?	249
What Makes Message-Driven Beans Different from Session Beans?	249
When to Use Message-Driven Beans	251
Accessing Enterprise Beans	251
Using Enterprise Beans in Clients	252
Deciding on Remote or Local Access	253
Local Clients	254
Remote Clients	255
Web Service Clients	256
Method Parameters and Access	257
The Contents of an Enterprise Bean	258
Packaging Enterprise Beans in EJB JAR Modules	258
Packaging Enterprise Beans in WAR Modules	259
Naming Conventions for Enterprise Beans	260
The Lifecycles of Enterprise Beans	261
The Lifecycle of a Stateful Session Bean	261
The Lifecycle of a Stateless Session Bean	262
The Lifecycle of a Singleton Session Bean	262
The Lifecycle of a Message-Driven Bean	263
Further Information about Enterprise Beans	264
 15 Getting Started with Enterprise Beans	265
Creating the Enterprise Bean	265
Coding the Enterprise Bean Class	266
Creating the converter Web Client	266
Building, Packaging, Deploying, and Running the converter Example	267
Modifying the Java EE Application	269
▼ To Modify a Class File	269
 16 Running the Enterprise Bean Examples	271
The cart Example	271
The Business Interface	272
Session Bean Class	273
The @Remove Method	276

Helper Classes	276
Building, Packaging, Deploying, and Running the <code>cart</code> Example	276
A Singleton Session Bean Example: <code>counter</code>	278
Creating a Singleton Session Bean	278
The Architecture of the <code>counter</code> Example	283
Building, Packaging, Deploying, and Running the <code>counter</code> Example	285
A Web Service Example: <code>helloservice</code>	286
The Web Service Endpoint Implementation Class	287
Stateless Session Bean Implementation Class	287
Building, Packaging, Deploying, and Testing the <code>helloservice</code> Example	288
Using the Timer Service	290
Creating Calendar-Based Timer Expressions	290
Programmatic Timers	293
Automatic Timers	294
Canceling and Saving Timers	296
Getting Timer Information	296
Transactions and Timers	296
The <code>timersession</code> Example	297
Building, Packaging, Deploying, and Running the <code>timersession</code> Example	299
Handling Exceptions	300
Part V Contexts and Dependency Injection for the Java EE Platform	303
17 Introduction to Contexts and Dependency Injection for the Java EE Platform	305
Overview of CDI	306
About Beans	307
About Managed Beans	307
Beans as Injectable Objects	308
Using Qualifiers	309
Injecting Beans	310
Using Scopes	310
Giving Beans EL Names	312
Adding Setter and Getter Methods	312
Using a Managed Bean in a Facelets Page	313
Injecting Objects by Using Producer Methods	314

Configuring a CDI Application	315
Further Information about CDI	315
18 Running the Basic Contexts and Dependency Injection Examples	317
The simplegreeting CDI Example	317
The simplegreeting Source Files	318
The Facelets Template and Page	318
Configuration Files	319
Building, Packaging, Deploying, and Running the simplegreeting CDI Example	320
The guessnumber CDI Example	322
The guessnumber Source Files	322
The Facelets Page	326
Building, Packaging, Deploying, and Running the guessnumber CDI Example	328
Part VI Persistence	331
19 Introduction to the Java Persistence API	333
Entities	333
Requirements for Entity Classes	334
Persistent Fields and Properties in Entity Classes	334
Primary Keys in Entities	339
Multiplicity in Entity Relationships	341
Direction in Entity Relationships	342
Embeddable Classes in Entities	344
Entity Inheritance	345
Abstract Entities	345
Mapped Superclasses	345
Non-Entity Superclasses	346
Entity Inheritance Mapping Strategies	347
Managing Entities	349
The EntityManager Interface	349
Persistence Units	353
Querying Entities	355

Further Information about Persistence	355
20 Running the Persistence Examples	357
The order Application	357
Entity Relationships in the order Application	358
Primary Keys in the order Application	360
Entity Mapped to More Than One Database Table	363
Cascade Operations in the order Application	363
BLOB and CLOB Database Types in the order Application	364
Temporal Types in the order Application	365
Managing the order Application's Entities	365
Building, Packaging, Deploying, and Running the order Application	368
The roster Application	369
Relationships in the roster Application	369
Entity Inheritance in the roster Application	370
Criteria Queries in the roster Application	372
Automatic Table Generation in the roster Application	374
Building, Packaging, Deploying, and Running the roster Application	374
The address-book Application	376
Bean Validation Constraints in address-book	376
Specifying Error Messages for Constraints in address-book	377
Validating Contact Input from a JavaServer Faces Application	378
Building, Packaging, Deploying, and Running the address-book Application	379
21 The Java Persistence Query Language	381
Query Language Terminology	382
Creating Queries Using the Java Persistence Query Language	382
Named Parameters in Queries	383
Positional Parameters in Queries	383
Simplified Query Language Syntax	384
Select Statements	384
Update and Delete Statements	385
Example Queries	385
Simple Queries	385
Queries That Navigate to Related Entities	386

Queries with Other Conditional Expressions	388
Bulk Updates and Deletes	389
Full Query Language Syntax	390
BNF Symbols	390
BNF Grammar of the Java Persistence Query Language	391
FROM Clause	394
Path Expressions	398
WHERE Clause	400
SELECT Clause	410
ORDER BY Clause	412
GROUP BY and HAVING Clauses	412
22 Using the Criteria API to Create Queries	415
Overview of the Criteria and Metamodel APIs	415
Using the Metamodel API to Model Entity Classes	417
Using Metamodel Classes	418
Using the Criteria API and Metamodel API to Create Basic Typesafe Queries	418
Creating a Criteria Query	418
Query Roots	419
Querying Relationships Using Joins	420
Path Navigation in Criteria Queries	421
Restricting Criteria Query Results	421
Managing Criteria Query Results	424
Executing Queries	425
Part VII Security	427
23 Introduction to Security in the Java EE Platform	429
Overview of Java EE Security	430
A Simple Security Example	430
Features of a Security Mechanism	433
Characteristics of Application Security	434
Security Mechanisms	435
Java SE Security Mechanisms	435

Java EE Security Mechanisms	436
Securing Containers	439
Using Annotations to Specify Security Information	439
Using Deployment Descriptors for Declarative Security	439
Using Programmatic Security	440
Securing the GlassFish Server	440
Working with Realms, Users, Groups, and Roles	441
What Are Realms, Users, Groups, and Roles?	441
Managing Users and Groups on the GlassFish Server	444
Setting Up Security Roles	446
Mapping Roles to Users and Groups	447
Establishing a Secure Connection Using SSL	449
Verifying and Configuring SSL Support	450
Working with Digital Certificates	450
Further Information about Security	454
24 Getting Started Securing Web Applications	455
Overview of Web Application Security	455
Securing Web Applications	457
Specifying Security Constraints	457
Specifying Authentication Mechanisms	461
Declaring Security Roles	468
Using Programmatic Security with Web Applications	469
Authenticating Users Programmatically	469
Checking Caller Identity Programmatically	471
Example Code for Programmatic Security	472
Declaring and Linking Role References	473
Examples: Securing Web Applications	474
▼ To Set Up Your System for Running the Security Examples	474
Example: Basic Authentication with a Servlet	475
Example: Form-Based Authentication with a JavaServer Faces Application	479
25 Getting Started Securing Enterprise Applications	485
Securing Enterprise Beans	486
Securing an Enterprise Bean Using Declarative Security	489

Securing an Enterprise Bean Programmatically	493
Propagating a Security Identity (Run-As)	494
Deploying Secure Enterprise Beans	496
Examples: Securing Enterprise Beans	496
Example: Securing an Enterprise Bean with Declarative Security	497
Example: Securing an Enterprise Bean with Programmatic Security	501
Securing Application Clients	504
Using Login Modules	505
Using Programmatic Login	505
Securing Enterprise Information Systems Applications	506
Container-Managed Sign-On	506
Component-Managed Sign-On	506
Configuring Resource Adapter Security	507
▼ To Map an Application Principal to EIS Principals	508
Part VIII Java EE Supporting Technologies	511
26 Introduction to Java EE Supporting Technologies	513
Transactions	513
Resources	514
The Java EE Connector Architecture and Resource Adapters	514
Java Message Service	514
Java Database Connectivity Software	515
27 Transactions	517
What Is a Transaction?	517
Container-Managed Transactions	518
Transaction Attributes	519
Rolling Back a Container-Managed Transaction	523
Synchronizing a Session Bean's Instance Variables	523
Methods Not Allowed in Container-Managed Transactions	523
Bean-Managed Transactions	524
JTA Transactions	524
Returning without Committing	525

- Methods Not Allowed in Bean-Managed Transactions525
- Transaction Timeouts525
 - ▼ To Set a Transaction Timeout526
- Updating Multiple Databases526
- Transactions in Web Components528
- Further Information about Transactions528

- 28 Resource Connections529**
 - Resources and JNDI Naming529
 - DataSource Objects and Connection Pools530
 - Resource Injection531
 - Field-Based Injection532
 - Method-Based Injection533
 - Class-Based Injection534
 - Resource Adapters and Contracts534
 - Management Contracts536
 - Generic Work Context Contract537
 - Outbound and Inbound Contracts537
 - Metadata Annotations538
 - Common Client Interface540
 - Further Information about Resources541

- Index543**

Preface

This tutorial is a guide to developing enterprise applications for the Java Platform, Enterprise Edition 6 (Java EE 6) using GlassFish Server Open Source Edition.

Oracle GlassFish Server, a Java EE compatible application server, is based on GlassFish Server Open Source Edition, the leading open-source and open-community platform for building and deploying next-generation applications and services. GlassFish Server Open Source Edition, developed by the GlassFish project open-source community at <https://glassfish.dev.java.net/>, is the first compatible implementation of the Java EE 6 platform specification. This lightweight, flexible, and open-source application server enables organizations not only to leverage the new capabilities introduced within the Java EE 6 specification, but also to add to their existing capabilities through a faster and more streamlined development and deployment cycle. Oracle GlassFish Server, the product version, and GlassFish Server Open Source Edition, the open-source version, are hereafter referred to as GlassFish Server.

The following topics are addressed here:

- “Before You Read This Book” on page xxi
- “Oracle GlassFish Server Documentation Set” on page xxii
- “Related Documentation” on page xxiv
- “Symbol Conventions” on page xxiv
- “Typographic Conventions” on page xxv
- “Default Paths and File Names” on page xxv
- “Documentation, Support, and Training” on page xxvi
- “Searching Oracle Product Documentation” on page xxvii
- “Third-Party Web Site References” on page xxvii

Before You Read This Book

Before proceeding with this tutorial, you should have a good knowledge of the Java programming language. A good way to get to that point is to work through *The Java Tutorial, Fourth Edition*, Sharon Zakhour et al. (Addison-Wesley, 2006).

Oracle GlassFish Server Documentation Set

The GlassFish Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for GlassFish Server documentation is <http://docs.sun.com/coll/1343.13>. For an introduction to GlassFish Server, refer to the books in the order in which they are listed in the following table.

TABLE P-1 Books in the GlassFish Server Documentation Set

Book Title	Description
<i>Release Notes</i>	Provides late-breaking information about the software and the documentation and includes a comprehensive, table-based summary of the supported hardware, operating system, Java Development Kit (JDK), and database drivers.
<i>Quick Start Guide</i>	Explains how to get started with the GlassFish Server product.
<i>Installation Guide</i>	Explains how to install the software and its components.
<i>Upgrade Guide</i>	Explains how to upgrade to the latest version of GlassFish Server. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
<i>Administration Guide</i>	Explains how to configure, monitor, and manage GlassFish Server subsystems and components from the command line by using the <code>asadmin(1M)</code> utility. Instructions for performing these tasks from the Administration Console are provided in the Administration Console online help.
<i>Application Deployment Guide</i>	Explains how to assemble and deploy applications to the GlassFish Server and provides information about deployment descriptors.
<i>Your First Cup: An Introduction to the Java EE Platform</i>	For beginning Java EE programmers, provides a short tutorial that explains the entire process for developing a simple enterprise application. The sample application is a web application that consists of a component that is based on the Enterprise JavaBeans specification, a JAX-RS web service, and a JavaServer Faces component for the web front end.
<i>Application Development Guide</i>	Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the GlassFish Server. These applications follow the open Java standards model for Java EE components and application programmer interfaces (APIs). This guide provides information about developer tools, security, and debugging.

TABLE P-1 Books in the GlassFish Server Documentation Set *(Continued)*

Book Title	Description
<i>Add-On Component Development Guide</i>	Explains how to use published interfaces of GlassFish Server to develop add-on components for GlassFish Server. This document explains how to perform <i>only</i> those tasks that ensure that the add-on component is suitable for GlassFish Server.
<i>Embedded Server Guide</i>	Explains how to run applications in embedded GlassFish Server and to develop applications in which GlassFish Server is embedded.
<i>Scripting Framework Guide</i>	Explains how to develop scripting applications in such languages as Ruby on Rails and Groovy on Grails for deployment to GlassFish Server.
<i>Troubleshooting Guide</i>	Describes common problems that you might encounter when using GlassFish Server and explains how to solve them.
<i>Error Message Reference</i>	Describes error messages that you might encounter when using GlassFish Server.
<i>Reference Manual</i>	Provides reference information in man page format for GlassFish Server administration commands, utility commands, and related concepts.
<i>Domain File Format Reference</i>	Describes the format of the GlassFish Server configuration file, <code>domain.xml</code> .
<i>Java EE 6 Tutorial</i>	Explains how to use Java EE 6 platform technologies and APIs to develop Java EE applications.
<i>Message Queue Release Notes</i>	Describes new features, compatibility issues, and existing bugs for GlassFish Message Queue.
<i>Message Queue Administration Guide</i>	Explains how to set up and manage a Message Queue messaging system.
<i>Message Queue Developer's Guide for JMX Clients</i>	Describes the application programming interface in Message Queue for programmatically configuring and monitoring Message Queue resources in conformance with the Java Management Extensions (JMX).

Related Documentation

Javadoc tool reference documentation for packages that are provided with GlassFish Server is available as follows.

- The API specification for version 6 of Java EE is located at http://download.oracle.com/docs/cd/E17410_01/javaee/6/api/.
- The API specification for GlassFish Server 3.0.1, including Java EE 6 platform packages and nonplatform packages that are specific to the GlassFish Server product, is located at <https://glassfish.dev.java.net/nonav/docs/v3/api/>.

Additionally, the Java EE Specifications at <http://www.oracle.com/technetwork/java/javaee/tech/index.html> might be useful.

For information about creating enterprise applications in the NetBeans Integrated Development Environment (IDE), see <http://www.netbeans.org/kb/>.

For information about the Java DB database for use with the GlassFish Server, see <http://www.oracle.com/technetwork/java/javadb/overview/index.html>.

The GlassFish Samples project is a collection of sample applications that demonstrate a broad range of Java EE technologies. The GlassFish Samples are bundled with the Java EE Software Development Kit (SDK) and are also available from the GlassFish Samples project page at <https://glassfish-samples.dev.java.net/>.

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P-2 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	ls [-l]	The -l option is not required.
{ }	Contains a set of choices for a required command option.	-d {y n}	The -d option requires that you use either the y argument or the n argument.
\${ }	Indicates a variable reference.	\${com.sun.javaRoot}	References the value of the com.sun.javaRoot variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.

TABLE P-2 Symbol Conventions (Continued)

Symbol	Description	Example	Meaning
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P-3 Typographic Conventions

Typeface	Meaning	Example
<i>AaBbCc123</i>	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name%</code> su Password:
<i>AaBbCc123</i>	A placeholder to be replaced with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the <i>User's Guide</i> . A <i>cache</i> is a copy that is stored locally. Do <i>not</i> save the file.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P-4 Default Paths and File Names

Placeholder	Description	Default Value
<i>as-install</i>	Represents the base installation directory for the GlassFish Server or the SDK of which the GlassFish Server is a part.	Installations on the Solaris operating system, Linux operating system, and Mac operating system: <i>user's-home-directory/glassfishv3/glassfish</i> Windows, all installations: <i>SystemDrive:\glassfishv3\glassfish</i>
<i>as-install-parent</i>	Represents the parent of the base installation directory for GlassFish Server.	Installations on the Solaris operating system, Linux operating system, and Mac operating system: <i>user's-home-directory/glassfishv3</i> Windows, all installations: <i>SystemDrive:\glassfishv3</i>
<i>tut-install</i>	Represents the base installation directory for the <i>Java EE Tutorial</i> after you install the GlassFish Server or the SDK and run the Update Tool.	<i>as-install/docs/javaee-tutorial</i>
<i>domain-root-dir</i>	Represents the directory in which a domain is created by default.	<i>as-install/domains/</i>
<i>domain-dir</i>	Represents the directory in which a domain's configuration is stored. In configuration files, <i>domain-dir</i> is represented as follows: <i>\${com.sun.aas.instanceRoot}</i>	<i>domain-root-dir/domain-name</i>

Documentation, Support, and Training

The Oracle web site provides information about the following additional resources:

- Documentation (<http://docs.sun.com/>)
- Support (<http://www.sun.com/support/>)
- Training (<http://education.oracle.com/>)

Searching Oracle Product Documentation

Besides searching Oracle product documentation from the `http://docs.sun.com` web site, you can use a search engine by typing the following syntax in the search field:

search-term **site:docs.sun.com**

For example, to search for “broker,” type the following:

broker site:docs.sun.com

To include other Oracle web sites in your search (for example, the Java Developer site on the Oracle Technology Network at `http://www.oracle.com/technetwork/java/index.html`), use `oracle.com` in place of `docs.sun.com` in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Acknowledgments

The Java EE tutorial team would like to thank the Java EE specification leads: Roberto Chinnici, Bill Shannon, Kenneth Saks, Linda DeMichiel, Ed Burns, Roger Kitain, Ron Monzillo, Dhru Pandey, Sankara Rao, Binod PG, Sivakumar Thyagarajan, Kin-Man Chung, Jan Luehe, Jitendra Kotamraju, Marc Hadley, Paul Sandoz, Gavin King, Emmanuel Bernard, Rod Johnson, Bob Lee, and Rajiv Mordani.

We would also like to thank the Java EE 6 SDK team, especially Carla Carlson, Snjezana Sevo-Zenzerovic, Adam Leftik, and John Clingan.

The *JavaServer Faces* technology and *Facelets* chapters benefited from the documentation reviews and example code contributions of Jim Driscoll and Ryan Lubke.

The EJB technology, Java Persistence API, and Criteria API chapters were written with extensive input from the EJB and Persistence teams, including Marina Vatkina and Mitesh Meswani.

We'd like to thank Pete Muir for his reviews of the CDI chapters and Tim Quinn for assistance with the application client container. Thanks also to the NetBeans engineering and documentation teams, particularly Petr Jiricka, John Jullion-Ceccarelli, and Troy Giunipero, for their help in enabling NetBeans IDE support for the code examples.

We would like to thank our manager, Alan Sommerer, for his support and steady influence.

We also thank Dwayne Wolff for developing the illustrations and Jordan Douglas for updating them. Julie Bettis, our editor, contributed greatly to the readability and flow of the book. Sheila Cepero helped smooth our path in many ways. Steve Cogorno provided invaluable help with our tools.

Finally, we would like to express our profound appreciation to Greg Doench, John Fuller, Vicki Rowland, Evelyn Pyle, and the production team at Addison-Wesley for graciously seeing our large, complicated manuscript to publication.

Introduction to Facelets

The term *Facelets* refers to the view declaration language for JavaServer Faces technology. JavaServer Pages (JSP) technology, previously used as the presentation technology for JavaServer Faces, does not support all the new features available in JavaServer Faces 2.0. JSP technology is considered to be a deprecated presentation technology for JavaServer Faces 2.0. Facelets is a part of the JavaServer Faces specification and also the preferred presentation technology for building JavaServer Faces technology-based applications.

The following topics are addressed here:

- “What Is Facelets?” on page 83
- “Developing a Simple Facelets Application” on page 85
- “Templating” on page 91
- “Composite Components” on page 94
- “Resources” on page 96

What Is Facelets?

Facelets is a powerful but lightweight page declaration language that is used to build JavaServer Faces views using HTML style templates and to build component trees. Facelets features include the following:

- Use of XHTML for creating web pages
- Support for Facelets tag libraries in addition to JavaServer Faces and JSTL tag libraries
- Support for the Expression Language (EL)
- Templating for components and pages

Advantages of Facelets for large-scale development projects include the following:

- Support for code reuse through templating and composite components
- Functional extensibility of components and other server-side objects through customization
- Faster compilation time
- Compile-time EL validation
- High-performance rendering

In short, the use of Facelets reduces the time and effort that needs to be spent on development and deployment.

Facelets views are usually created as XHTML pages. JavaServer Faces implementations support XHTML pages created in conformance with the XHTML Transitional Document Type Definition (DTD), as listed at http://www.w3.org/TR/xhtml1/#a_dtd_XHTML-1.0-Transitional. By convention, web pages built with XHTML have an `.xhtml` extension.

JavaServer Faces technology supports various tag libraries to add components to a web page. To support the JavaServer Faces tag library mechanism, Facelets uses XML namespace declarations. Table 5–1 lists the tag libraries supported by Facelets.

TABLE 5–1 Tag Libraries Supported by Facelets

Tag Library	URI	Prefix	Example	Contents
JavaServer Faces Facelets Tag Library	http://java.sun.com/jsf/facelets	ui:	ui:component ui:insert	Tags for templating
JavaServer Faces HTML Tag Library	http://java.sun.com/jsf/html	h:	h:head h:body h:outputText h:inputText	JavaServer Faces component tags for all UIComponents
JavaServer Faces Core Tag Library	http://java.sun.com/jsf/core	f:	f:actionListener f:attribute	Tags for JavaServer Faces custom actions that are independent of any particular RenderKit

TABLE 5-1 Tag Libraries Supported by Facelets *(Continued)*

Tag Library	URI	Prefix	Example	Contents
JSTL Core Tag Library	http://java.sun.com/jsp/jstl/core	c:	c:forEach c:catch	JSTL 1.1 Core Tags
JSTL Functions Tag Library	http://java.sun.com/jsp/jstl/functions	fn:	fn:toUpperCase fn:toLowerCase	JSTL 1.1 Functions Tags

In addition, Facelets supports tags for composite components for which you can declare custom prefixes. For more information on composite components, see “Composite Components” on page 94.

Based on the JavaServer Faces support for Expression Language (EL) syntax defined by JSP 2.1, Facelets uses EL expressions to reference properties and methods of backing beans. EL expressions can be used to bind component objects or values to methods or properties of managed beans. For more information on using EL expressions, see “Using the EL to Reference Backing Beans” on page 161.

Developing a Simple Facelets Application

This section describes the general steps involved in developing a JavaServer Faces application. The following tasks are usually required:

- Developing the backing beans
- Creating the pages using the component tags
- Defining page navigation
- Mapping the FacesServlet instance
- Adding managed bean declarations

Creating a Facelets Application

The example used in this tutorial is the guessnumber application. The application presents you with a page that asks you to guess a number between 0 and 10, validates your input against a random number, and responds with another page that informs you whether you guessed the number correctly or incorrectly.

Developing a Backing Bean

In a typical JavaServer Faces application, each page of the application connects to a backing bean, a type of managed bean. The backing bean defines the methods and properties that are associated with the components.

The following managed bean class, `UserNumberBean.java`, generates a random number from 0 to 10:

```
package guessNumber;

import java.util.Random;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class UserNumberBean {

    Integer randomInt = null;
    Integer userNumber = null;
    String response = null;
    private long maximum=10;
    private long minimum=0;

    public UserNumberBean() {
        Random randomGR = new Random();
        randomInt = new Integer(randomGR.nextInt(10));
        System.out.println("Duke's number: " + randomInt);
    }

    public void setUserNumber(Integer user_number) {
        userNumber = user_number;
    }

    public Integer getUserNumber() {
        return userNumber;
    }

    public String getResponse() {
        if ((userNumber != null) && (userNumber.compareTo(randomInt) == 0)) {
            return "Yay! You got it!";
        } else {
            return "Sorry, " + userNumber + " is incorrect.";
        }
    }

    public long getMaximum() {
        return (this.maximum);
    }

    public void setMaximum(long maximum) {
        this.maximum = maximum;
    }

    public long getMinimum() {
        return (this.minimum);
    }

    public void setMinimum(long minimum) {
        this.minimum = minimum;
    }
}
```

Note the use of the `@ManagedBean` annotation, which registers the backing bean as a resource with JavaServer Faces implementation. The `@SessionScoped` annotation registers the bean scope as session.

Creating Facelets Views

Creating a page or view is the responsibility of a page author. This task involves adding components on the pages, wiring the components to backing bean values and properties, and registering converters, validators, or listeners onto the components.

For the example application, XHTML web pages serve as the front end. The first page of the example application is a page called `greeting.xhtml`. A closer look at various sections of this web page provides more information.

The first section of the web page declares the content type for the page, which is XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

The next section declares the XML namespace for the tag libraries that are used in the web page:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
```

The next section uses various tags to insert components into the web page:

```
h:head>
  <title>Guess Number Facelets Application</title>
</h:head>
<h:body>
  <h:form>
    <h:graphicImage value="#{resource['images:wave.med.gif']}" />
    <h2>
      Hi, my name is Duke. I am thinking of a number from
      #{userNumberBean.minimum} to #{userNumberBean.maximum}.
      Can you guess it?
    <p></p>
    <h:inputText
      id="userNo"
      value="#{userNumberBean.userNumber}"
      <f:validateLongRange
        minimum="#{userNumberBean.minimum}"
        maximum="#{userNumberBean.maximum}" />
    </h:inputText>

    <h:commandButton id="submit" value="Submit"
      action="response.xhtml" />
    <h:message showSummary="true" showDetail="false"
      style="color: red;
```

```
font-family: 'New Century Schoolbook', serif;
font-style: oblique;
text-decoration: overline"
id="errors1"
for="userNo"/>
</h2>
</h:form>
</h:body>
```

Note the use of the following tags:

- Facelets HTML tags (those beginning with `h:`) to add components
- The Facelets core tag `f:validateLongRange` to validate the user input

An `inputText` component accepts user input and sets the value of the backing bean property `userNumber` through the EL expression `#{userNumberBean.userNumber}`. The input value is validated for value range by the JavaServer Faces standard validator `f:validateLongRange`.

The image file, `wave.med.gif`, is added to the page as a resource. For more details about the resources facility, see “Resources” on page 96.

A `commandButton` component with the ID `submit` starts validation of the input data when a user clicks the button. Using implicit navigation, the component redirects the client to another page, `response.xhtml`, which shows the response to your input.

You can now create the second page, `response.xhtml`, with the following content:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">

    <h:head>
        <title>Guess Number Facelets Application</title>
    </h:head>
    <h:body>
        <h:form>
            <h:graphicImage value="#{resource['images:wave.med.gif']}" />
            <h2>
                <h:outputText id="result" value="#{userNumberBean.response}" />
            </h2>
            <h:commandButton id="back" value="Back" action="greeting.xhtml" />
        </h:form>
    </h:body>
</html>
```

Configuring the Application

Configuring a JavaServer Faces application involves mapping the Faces Servlet in the web deployment descriptor file, such as a `web.xml` file, and possibly adding managed

bean declarations, navigation rules, and resource bundle declarations to the application configuration resource file, `faces-config.xml`.

If you are using NetBeans IDE, a web deployment descriptor file is automatically created for you. In such an IDE-created `web.xml` file, change the default greeting page, which is `index.xhtml`, to `greeting.xhtml`. Here is an example `web.xml` file, showing this change in bold.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/greeting.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

Note the use of the context parameter `PROJECT_STAGE`. This parameter identifies the status of a JavaServer Faces application in the software lifecycle.

The stage of an application can affect the behavior of the application. For example, if the project stage is defined as `Development`, debugging information is automatically generated for the user. If not defined by the user, the default project stage is `Production`.

Building, Packaging, Deploying, and Running the guessnumber Facelets Example

You can use either NetBeans IDE or Ant to build, package, deploy, and run the `guessnumber` example. The source code for this example is available in the `tut-install/examples/web/guessnumber` directory.

▼ To Build, Package, and Deploy the guessnumber Example Using NetBeans IDE

- 1 In NetBeans IDE, select File→Open Project.
- 2 In the Open Project dialog, navigate to:
tut-install/examples/web/
- 3 Select the `guessnumber` folder.
- 4 Select the Open as Main Project check box.
- 5 Click Open Project.
- 6 In the Projects tab, right-click the `guessnumber` project and select Deploy.
This option builds and deploys the example application to your GlassFish Server instance.

▼ To Build, Package, and Deploy the guessnumber Example Using Ant

- 1 In a terminal window, go to:
tut-install/examples/web/guessnumber/
- 2 Type the following command:
ant
This command calls the default target, which builds and packages the application into a WAR file, `guessnumber.war`, that is located in the `dist` directory.
- 3 Make sure that the GlassFish Server is started.
- 4 To deploy the application, type the following command:
ant deploy

▼ To Run the guessnumber Example

- 1 Open a web browser.
- 2 Type the following URL in your web browser:
`http://localhost:8080/guessnumber`
The web page shown in Figure 5–1 appears.

FIGURE 5-1 Running the guessnumber Application



- 3 In the text field, type a number from 0 to 10 and click Submit.
Another page appears, reporting whether your guess is correct or incorrect.
- 4 If you guessed incorrectly, click the Back button to return to the main page.
You can continue to guess until you get the correct answer.

Templating

JavaServer Faces technology provides the tools to implement user interfaces that are easy to extend and reuse. Templating is a useful Facelets feature that allows you to create a page that will act as the base, or *template*, for the other pages in an application. By using templates, you can reuse code and avoid recreating similarly constructed pages. Templating also helps in maintaining a standard look and feel in an application with a large number of pages.

Table 5-2 lists Facelets tags that are used for templating and their respective functionality.

TABLE 5-2 Facelets Templating Tags

Tag	Function
<code>ui:component</code>	Defines a component that is created and added to the component tree.
<code>ui:composition</code>	Defines a page composition that optionally uses a template. Content outside of this tag is ignored.

TABLE 5-2 Facelets Templating Tags (Continued)

Tag	Function
<code>ui:debug</code>	Defines a debug component that is created and added to the component tree.
<code>ui:decorate</code>	Similar to the composition tag but does not disregard content outside this tag.
<code>ui:define</code>	Defines content that is inserted into a page by a template.
<code>ui:fragment</code>	Similar to the component tag but does not disregard content outside this tag.
<code>ui:include</code>	Encapsulate and reuse content for multiple pages.
<code>ui:insert</code>	Inserts content into a template.
<code>ui:param</code>	Used to pass parameters to an included file.
<code>ui:repeat</code>	Used as an alternative for loop tags, such as <code>c:forEach</code> or <code>h:dataTable</code> .
<code>ui:remove</code>	Removes content from a page.

For more information on Facelets templating tags, see the documentation at http://download.oracle.com/docs/cd/E17410_01/javaee/6/javaserverfaces/2.0/docs/pdl/docs/facelets/.

The Facelets tag library includes the main templating tag `ui:insert`. A template page that is created with this tag allows you to define a default structure for a page. A template page is used as a template for other pages, usually referred to as client pages.

Here is an example of a template saved as `template.xhtml`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html">

    <h:head>
        <meta http-equiv="Content-Type"
            content="text/html; charset=UTF-8" />
        <link href="/resources/css/default.css"
            rel="stylesheet" type="text/css" />
        <link href="/resources/css/cssLayout.css"
            rel="stylesheet" type="text/css" />
        <title>Facelets Template</title>
    </h:head>

    <h:body>
        <div id="top" class="top">
            <ui:insert name="top">Top Section</ui:insert>
        </div>
        <div>
            <div id="left">
                <ui:insert name="left">Left Section</ui:insert>
            </div>
        </div>
    </h:body>
</html>
```

```

        </div>
        <div id="content" class="left_content">
            <ui:insert name="content">Main Content</ui:insert>
        </div>
    </div>
</h:body>
</html>

```

The example page defines an XHTML page that is divided into three sections: a top section, a left section, and a main section. The sections have style sheets associated with them. The same structure can be reused for the other pages of the application.

The client page invokes the template by using the `ui:composition` tag. In the following example, a client page named `templateClient.xhtml` invokes the template page named `template.xhtml` from the preceding example. A client page allows content to be inserted with the help of the `ui:define` tag.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html">

    <h:body>
        <ui:composition template="./template.xhtml">
            <ui:define name="top">
                Welcome to Template Client Page
            </ui:define>

            <ui:define name="left">
                <h:outputLabel value="You are in the Left Section"/>
            </ui:define>

            <ui:define name="content">
                <h:graphicImage value="#{resource['images:wave.med.gif']}" />
                <h:outputText value="You are in the Main Content Section" />
            </ui:define>
        </ui:composition>
    </h:body>
</html>

```

You can use NetBeans IDE to create Facelets template and client pages. For more information on creating these pages, see <http://netbeans.org/kb/docs/web/jsf20-intro.html>.

Composite Components

JavaServer Faces technology offers the concept of composite components with Facelets. A composite component is a special type of template that acts as a component.

Any component is essentially a piece of reusable code that behaves in a particular way. For example, an `inputText` component accepts user input. A component can also have validators, converters, and listeners attached to it to perform certain defined actions.

A composite component consists of a collection of markup tags and other existing components. This reusable, user-created component has a customized, defined functionality and can have validators, converters, and listeners attached to it like any other component.

With Facelets, any XHTML page that contains markup tags and other components can be converted into a composite component. Using the resources facility, the composite component can be stored in a library that is available to the application from the defined resources location.

Table 5–3 lists the most commonly used composite tags and their functions.

TABLE 5–3 Composite Component Tags

Tag	Function
<code>composite:interface</code>	Declares the usage contract for a composite component. The composite component can be used as a single component whose feature set is the union of the features declared in the usage contract.
<code>composite:implementation</code>	Defines the implementation of the composite component. If a <code>composite:interface</code> element appears, there must be a corresponding <code>composite:implementation</code> .
<code>composite:attribute</code>	Declares an attribute that may be given to an instance of the composite component in which this tag is declared.
<code>composite:insertChildren</code>	Any child components or template text within the composite component tag in the using page will be reparented into the composite component at the point indicated by this tag's placement within the <code>composite:implementation</code> section.
<code>composite:valueHolder</code>	Declares that the composite component whose contract is declared by the <code>composite:interface</code> in which this element is nested exposes an implementation of <code>ValueHolder</code> suitable for use as the target of attached objects in the using page.

TABLE 5-3 Composite Component Tags (Continued)

Tag	Function
<code>composite:editableValueHolder</code>	Declares that the composite component whose contract is declared by the <code>composite:interface</code> in which this element is nested exposes an implementation of <code>EditableValueHolder</code> suitable for use as the target of attached objects in the using page.
<code>composite:actionSource</code>	Declares that the composite component whose contract is declared by the <code>composite:interface</code> in which this element is nested exposes an implementation of <code>ActionSource2</code> suitable for use as the target of attached objects in the using page.

For more information and a complete list of Facelets composite tags, see the documentation at http://download.oracle.com/docs/cd/E17410_01/javaee/6/javaserverfaces/2.0/docs/pdldocs/facelets/.

The following example shows a composite component that accepts an email address as input:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:composite="http://java.sun.com/jsf/composite"
    xmlns:h="http://java.sun.com/jsf/html">

    <h:head>
        <title>This content will not be displayed</title>
    </h:head>
    <h:body>
        <composite:interface>
            <composite:attribute name="value" required="false"/>
        </composite:interface>

        <composite:implementation>
            <h:outputLabel value="Email id: "></h:outputLabel>
            <h:inputText value="#{cc.attrs.value}"></h:inputText>
        </composite:implementation>
    </h:body>
</html>
```

Note the use of `cc.attrs.value` when defining the value of the `inputText` component. The word `cc` in JavaServer Faces is a reserved word for composite components. The `#{cc.attrs.attribute-name}` expression is used to access the attributes defined for the composite component's interface, which in this case happens to be `value`.

The preceding example content is stored as a file named `email.xhtml` in a folder named `resources/emcomp`, under the application web root directory. This directory is

considered a library by JavaServer Faces, and a component can be accessed from such a library. For more information on resources, see “Resources” on page 96.

The web page that uses this composite component is generally called a *using page*. The using page includes a reference to the composite component, in the xml namespace declarations:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:em="http://java.sun.com/jsf/composite/emcomp/">

    <h:head>
        <title>Using a sample composite component</title>
    </h:head>

    <body>
        <h:form>
            <em:email value="Enter your email id" />
        </h:form>
    </body>
</html>
```

The local composite component library is defined in the xml namespace with the declaration `xmlns:em="http://java.sun.com/jsf/composite/emcomp/"`. The component itself is accessed through the use of `em:email` tag. The preceding example content can be stored as a web page named `emuserpage.xhtml` under the web root directory. When compiled and deployed on a server, it can be accessed with the following URL:

```
http://localhost:8080/application-name/faces/emuserpage.xhtml
```

Resources

Web resources are any software artifacts that the web application requires for proper rendering, including images, script files, and any user-created component libraries. Resources must be collected in a standard location, which can be one of the following.

- A resource packaged in the web application root must be in a subdirectory of a resources directory at the web application root: `resources/resource-identifier`.
- A resource packaged in the web application's classpath must be in a subdirectory of the `META-INF/resources` directory within a web application: `META-INF/resources/resource-identifier`.

The JavaServer Faces runtime will look for the resources in the preceding listed locations, in that order.

Resource identifiers are unique strings that conform to the following format:

[locale-prefix/][library-name/][library-version/]resource-name[/resource-version]

Elements of the resource identifier in brackets ([]) are optional, indicating that only a *resource-name*, which is usually a file name, is a required element.

Resources can be considered as a library location. Any artifact, such as a composite component or a template that is stored in the resources directory, becomes accessible to the other application components, which can use it to create a resource instance.

Index

Numbers and Symbols

- @AccessTimeout annotation, 281
- @ApplicationScoped annotation, 310–312
- @ConcurrencyManagement annotation, 280
- @Consumes annotation, 229–231
- @ConversationScoped annotation, 310–312
- @DeclareRoles annotation, 490–492
- @DELETE annotation, 220–235
- @DenyAll annotation, 491
- @Dependent annotation, 310–312
- @DependsOn annotation, 279
- @DiscriminatorColumn annotation, 347–348
- @DiscriminatorValue annotation, 347–348
- @Embeddable annotation, 344–345
- @EmbeddedId annotation, 339
- @Entity annotation, 334
- @GET annotation, 220–235
- @HttpConstraint
 - annotation, 457, 476
- @HttpMethodConstraint annotation, 457, 476
- @Id annotation, 339
- @IdClass annotation, 339
- @Inject annotation, 310
- @Local annotation, 253, 272
- @Lock annotation, 280–282
- @ManagedBean annotation, 77, 85–87
- @ManyToMany annotation, 341, 342
- @ManyToOne annotation, 341
- @Named annotation, 312
- @NamedQuery annotation, 382
- @OneToMany annotation, 341, 342, 343
- @OneToOne annotation, 341, 342, 343
- @Path annotation, 220–235
- @PathParam annotation, 231–235
- @PermitAll annotation, 491
- @PersistenceContext annotation, 350
- @PersistenceUnit annotation, 350
- @POST annotation, 220–235
- @PostActivate annotation, 273, 274
- @PostConstruct annotation, 261–264, 273, 274
- @PreDestroy annotation, 261–264, 273, 274
- @PrePassivate annotation, 273, 274
- @Produces annotation, 229–231, 314
- @PUT annotation, 220–235
- @Qualifier annotation, 309
- @QueryParam annotation, 231–235
- @Remote annotation, 253, 272
- @Remove annotation, 261, 273, 276
- @RequestScoped annotation, 310–312
- @Resource annotation, 531–534
- @RolesAllowed annotation, 490
- @RunAs annotation, 494–496
- @Schedule and @Schedules annotations, 294–295
- @ServletSecurity annotation, 457, 476
- @SessionScoped annotation, 310–312
- @Singleton annotation, 278
- @Startup annotation, 278
- @Stateful annotation, 273
- @Timeout annotation, 293
- @Timeout method, 296
- @Transient annotation, 335
- @WebFilter annotation, 187
- @WebInitParam annotation, 184, 188
- @WebListener annotation, 181

- @WebMethod annotation, 275
- @WebService annotation, 208
- @WebServiceRef annotation, 70
- @WebServlet annotation, 62, 183–184

A

- abstract schemas, 382
- access control, 434
- action events, 127
 - actionListener attribute, 126, 154, 156
 - ActionListener interface, 152
 - actionListener tag, 143, 152
 - referencing methods that handle action events, 156, 172
 - writing a backing-bean method to handle action events, 172–173
- Administration Console, 34
 - starting, 42–43
- afterBegin method, 523
- afterCompletion method, 523
- annotations, 3
 - JAX-RS, 220–235
 - security, 439, 476–477, 486, 490–492
- Ant tool, 41
- appliance tool, 34
- applet containers, 15
- applets, 9, 10
- application client containers, 15
- application clients, 8–9
 - securing, 504–505
- applications
 - JavaServer Faces, 74
 - security, 436–437
 - undeploying, 61–62
- asadmin tool, 34
- attributes referencing backing bean methods, 154
 - action attribute, 154, 155
 - actionListener attribute, 154, 156
 - validator attribute, 155, 156
 - valueChangeListener attribute, 155, 156
- audit modules, pluggable, 441
- auditing, 435
- auth-constraint element, 459

- authenticate method, 469–471
- authenticating users, 461–468
- authentication, 434–435, 449
 - basic, 462
 - certificate-based mutual, 465
 - client, 465
 - digest, 464–465
 - form-based, 463–464, 479–484
 - mutual, 465–466
 - user name/password-based mutual, 466
- authentication mechanism, EJB, 493
- authorization, 434–435
- authorization constraints, 458, 459
- authorization providers, pluggable, 441
- auto commit, 28

B

- backing bean methods
 - See* attributes referencing backing bean methods
 - See* referencing backing bean methods
 - See* writing backing bean methods
- backing bean properties, 146, 160–161, 162
 - bound to component instances, 168–169
 - properties for UISelectItems composed of SelectItem instances, 168
 - UIData properties, 164–165
 - UIInput and UIOutput properties, 163
 - UISelectBoolean properties, 165
 - UISelectedItem properties, 167
 - UISelectItems properties, 167–168
 - UISelectMany properties, 165–166
 - UISelectOne properties, 166–167
 - writing, 162–170
- backing beans, 74, 159–162
 - developing, 77, 85–87
 - method binding, 122
 - properties
 - See* backing bean properties
- basic authentication, 462
 - EJB, 493
 - example, 475–479
- bean-managed transactions, *See* transactions, bean-managed

- bean validation, 29
 - Bean Validation
 - constraints, 376–377
 - examples, 376–380
 - Java Persistence API, 337–339
 - JavaServer Faces applications, 174–178, 378–379
 - beans, defined for CDI, 307
 - beans.xml file, 315
 - beforeCompletion method, 523
 - BLOBs, *See* persistence, BLOBs
 - bookmarkable URLs, component tags, 139–140
 - BufferedReader class, 185
 - build artifacts, removing, 61–62
 - business logic, 246
 - business methods, 256
 - client calls, 275
 - exceptions, 276
 - locating, 266
 - requirements, 275
 - transactions, 521, 523, 525, 526
- C**
- CallbackHandler interface, 504
 - capture-schema tool, 34
 - certificate authorities, 451
 - certificates, 436
 - digital, 437, 450–453
 - managing, 451
 - server
 - generating, 452–453
 - using for authentication, 446
 - class files, removing, 61–62
 - clients
 - authenticating, 465
 - securing, 504–505
 - CLOBs, *See* persistence, CLOBs
 - collections
 - persistence, 335–337, 426
 - commit method, 523
 - commits, *See* transactions, commits
 - Common Client Interface, Connector
 - Architecture, 540–541
 - component binding, 162
 - binding attribute, 162
 - component classes
 - UIData class, 164–165
 - UIInput and UIOutput classes, 163
 - UISelectBoolean class, 165
 - UISelectedItem class, 167
 - UISelectItems class, 167
 - UISelectMany class, 165–166
 - UISelectOne class, 166–167
 - component-managed sign-on, 506
 - component properties, *See* backing bean properties
 - component tag attributes
 - action attribute, 171
 - actionListener attribute, 126, 154, 172
 - binding attribute, 117, 119, 162
 - columns attribute, 129
 - converter attribute, 122, 146–147
 - for attribute, 124, 138
 - id attribute, 117
 - immediate attribute, 117, 118
 - redisplay attribute, 124
 - rendered attribute, 117, 118
 - style attribute, 117, 119, 138
 - styleClass attribute, 117, 119
 - validator attribute, 122, 173
 - value attribute, 117, 119, 162
 - valueChangeListener attribute, 122, 156, 173
 - component tags, 162
 - attributes
 - See* component tag attributes
 - body tag, 119–120
 - bookmarkable URLs, 139–140
 - button tag, 139–140
 - column tag, 115
 - commandButton tag, 115, 126–127
 - commandLink tag, 115, 127
 - dataTable tag, 115, 135–138, 164
 - form tag, 115, 120
 - graphicImage tag, 115, 127
 - head tag, 119–120
 - inputHidden tag, 115, 121
 - inputSecret tag, 115, 121, 124
 - inputText tag, 115, 121, 123–124

component tags (*Continued*)

- inputTextarea tag, 115, 121
- link tag, 139–140
- message tag, 115, 138–139
- messages tag, 115, 138–139
- output tag, 141–142
- outputFormat tag, 115, 125
- outputLabel tag, 116, 123, 124–125
- outputLink tag, 116, 123, 125
- outputMessage tag, 123
- outputText tag, 116, 123, 127, 165
- panelGrid tag, 116, 128–129
- panelGroup tag, 116, 128–129
- resource relocation, 141–142
- selectBooleanCheckbox tag, 116, 130, 165
- selectItems tag, 167
- selectManyCheckbox tag, 116, 132–133, 165
- selectManyListbox tag, 116, 132
- selectManyMenu tag, 116, 132
- selectOneListbox tag, 116, 131
- selectOneMenu tag, 117, 131, 166, 167
- selectOneRadio tag, 117, 131

components

- buttons, 115
- check boxes, 116
- combo boxes, 116, 117
- data grids, 115
- hidden fields, 115
- hyperlinks, 115
- labels, 116
- list boxes, 116
- password fields, 115
- radio buttons, 117
- table columns, 115
- tables, 116
- text areas, 115
- text fields, 115

composite components, Facelets, 94–96

concurrent access, 517

confidentiality, 449

Connection interface, 523, 528

connection pooling, 530

connections, securing, 449–453

connectors, *See* Java EE Connector architecture

container-managed sign-on, 506

container-managed transactions, *See* transactions,
container-managed

containers, 13–15

See also applet containers*See also* application client containers*See also* EJB containers*See also* web containers

configurable services, 13

nonconfigurable services, 13

securing, 439–440

security, 430–435

services, 13

trust between, 496

context parameters, 57

specifying, 66–67

context roots, 57–58

Contexts and Dependency Injection (CDI) for the

Java EE platform, 29, 305–315

beans, 307

configuring applications, 315

EL, 312

examples, 317–330

Facelets pages, 313

injectable objects, 308

injecting beans, 310

managed beans, 307–308

overview, 306

producer methods, 314

qualifiers, 309

scopes, 310–312

setter and getter methods, 312–313

conversational state, 247

conversion model

See also converter tags

converter attribute, 122, 146–147

Converter implementations, 145–151

converterId attribute, 146

javax.faces.convert package, 145

Converter implementation classes

BigDecimalConverter class, 145

BigIntegerConverter class, 145

BooleanConverter class, 145

ByteConverter class, 145

- CharacterConverter class, 145
- DateTimeConverter class, 145, 146, 147
- DoubleConverter class, 145
- EnumConverter class, 145
- FloatConverter class, 145
- IntegerConverter class, 146
- LongConverter class, 146
- NumberConverter class, 146, 147, 149–151
- ShortConverter class, 146
- converter tags
 - convertDateTime tag, 147
 - convertDateTime tag attributes, 148–149
 - converter tag, 147
 - convertNumber tag, 147, 149–151
 - convertNumber tag attributes, 150–151
- cookie parameters, 234
- createTimer method, 293
- credential, 444
- Criteria API, 415–426
 - creating queries, 418–419
 - examples, 372–373
 - expressions, 421–422, 422–423
 - path navigation, 421
 - query execution, 425–426
 - query results, 421–423, 424–425
- cryptography, public-key, 451
- custom validators
 - validate method, 173
 - Validator implementation
 - backing bean methods, 170

D

- data encryption, 465
- data integrity, 434, 517, 518
- data sources, 530
- databases
 - See also* transactions
 - clients, 246
 - connections, 276, 525
 - data recovery, 517
 - EIS tier, 6
 - message-driven beans and, 250
 - multiple, 524, 526–527

- DataSource interface, 530
- debugging, Java EE applications, 45–46
- declarative security, 430, 456, 486
- Dependency Injection for Java (JSR 330), 29, 305
- deployer roles, 21
- deployment, 267–269
- deployment descriptors, 17, 430, 439–440, 456
 - enterprise bean, 440
 - enterprise beans, 259, 486, 488
 - Java EE, 18
 - runtime, 18
 - security-role-mapping element, 447–448
 - security-role-ref element, 473–474
 - web application, 53, 440
 - runtime, 54
 - web applications, 51
- destroy method, 195
- development roles, 19–22
 - application assemblers, 21
 - application client developers, 21
 - application component providers, 20–21
 - application deployers and administrators, 21
 - enterprise bean developers, 20
 - Java EE product providers, 20
 - tool providers, 20
 - web component developers, 20
- digest authentication, 464–465
- digital signatures, 451
- DNS, 31
- document roots, 53
- doFilter method, 187, 188, 190
- doGet method, 184
- domains, 42
- doPost method, 184
- downloading, GlassFish Server, 38

E

- EAR files, 17
- EIS tier, 12
 - security, 506–509
- EJB, security, 486–496
- EJB containers, 14
 - container-managed transactions, 518

- EJB containers (*Continued*)
 - services, 245, 246, 486–496
- EJB JAR files, 258
- ejb-jar.xml file, 259, 440, 488
- EJBContext interface, 523, 525
- EL, 78, 99–112
 - backing beans, 161–162
 - composite expressions, 106
 - deferred evaluation expressions, 100
 - expression examples, 112
 - immediate evaluation expressions, 100
 - literal expressions, 106, 109
 - literals, 105
 - lvalue expressions, 100, 102
 - managed beans, 312
 - method expressions, 100, 106
 - operators, 111
 - overview, 99–100
 - parameterized method calls, 107–108
 - reserved words, 111
 - rvalue expressions, 100, 102
 - tag attribute type, 108–109
 - type conversion during expression evaluation, 106
 - value expressions, 100, 102
- embeddable classes, *See* persistence: embeddable classes
- end-to-end security, 438
- enterprise applications, 3
- enterprise beans, 11, 25–26
 - See also* business methods
 - See also* Java EE components
 - See also* message-driven beans
 - See also* session beans
 - accessing, 251
 - classes, 258
 - compiling, 267–269
 - contents, 258–260
 - defined, 245
 - dependency injection, 252
 - deployment, 258
 - distribution, 253
 - exceptions, 300–301
 - getCallerPrincipal method, 493–494
 - implementor of business logic, 11
 - interfaces, 251–258, 258
 - isCallerInRole method, 493–494
 - JAX-RS resources, 237–240
 - JNDI lookup, 252
 - lifecycles, 261–264
 - local access, 254–255
 - local interfaces, 254
 - packaging, 258, 267–269
 - performance, 253
 - programmatic security, 493–494
 - remote access, 255–256
 - remote interfaces, 256
 - securing, 486–496
 - singletons, 238
 - timer service, 290–300
 - types, 246
 - web services, 247, 256–257, 286–289
- Enterprise Information Systems, *See* EIS tier
- entities
 - abstract, 345
 - abstract schema names, 384
 - application-managed entity managers, 350–351
 - cascading operations, 343
 - orphans, 343–344
 - collections, 397
 - container-managed entity managers, 350
 - creating, 365–366
 - discriminator columns, 347
 - entity manager, 349–353
 - finding, 351–352, 366
 - inheritance, 345–349, 370–371
 - inheritance mapping, 347–349
 - lifecycle, 352
 - managing, 349–355, 365–367
 - mapping to multiple tables, 363
 - non-entity superclasses, 346
 - overview, 333–345
 - persistent fields, 334–339
 - persistent properties, 334–339
 - persisting, 352
 - primary keys, 339–341
 - querying, 355

- relationships, 366
- removing, 353, 367
- requirements, 334
- superclasses, 345–346
- synchronizing, 353
- validating, 337–339
- entity providers, 227–229
- entity relationships
 - bidirectional, 342
 - many-to-many, 341, 369–370
 - many-to-one, 341
 - multiplicity, 341
 - one-to-many, 341
 - one-to-one, 341
 - query language, 342
 - unidirectional, 342
- equals method, 340
- event and listener model
 - See also* value-change events
 - listener class, 170
 - ValueChangedEvent class, 156
- examples, 37–46
 - basic authentication, 475–479
 - Bean Validation, 376–380
 - building, 44
 - CDI, 317–330
 - classpath, 268
 - Criteria API, 372–373
 - directory structure, 44
 - JAX-RS, 235–240
 - JAX-WS, 208–216
 - persistence, 357–380
 - primary keys, 340
 - query language, 366–367, 385–390
 - required software, 37–41
 - security, 430–433
 - form-based authentication, 479–484
 - servlet, 198–199
 - servlets, 62–70, 266
 - session beans, 266, 271–278
 - singleton session beans, 278–286
 - timer service, 297–299
 - web clients, 266
 - web services, 286–289

- exceptions
 - business methods, 276
 - enterprise beans, 300–301
 - mapping to error screens, 67–68
 - rolling back transactions, 301, 523
 - transactions, 520, 521
- Expression Language
 - See* EL
- expressions
 - lvalue expressions, 161
 - tag attribute type, 108–109

F

- Facelets, 83–97
 - See also* EL
 - composite components, 94–96
 - configuring applications, 88–89
 - features, 83–85
 - resources, 96–97
 - templating, 91–93
 - XHTML pages, 87–88
- Facelets applications, developing, 85–91
- FacesServlet, mapping, 78–79
- filter chains, 187, 190
- Filter interface, 187
- filters, 187
 - defining, 187
 - mapping to web components, 189
 - mapping to web resources, 189
 - overriding request methods, 189
 - overriding response methods, 189
 - response wrappers, 189
- foreign keys, 359
- form-based authentication, 463–464
- form parameters, 234
- forward method, 192

G

- garbage collection, 264
- GenericServlet interface, 180
- getCallerPrincipal method, 493–494

- getConnection method, 530
- getRemoteUser method, 471
- getRequestDispatcher method, 191
- getRollbackOnly method, 525
- getServletContext method, 193
- getSession method, 193
- getStatus method, 525
- getUserPrincipal method, 471

GlassFish Server

- adding users to, 445–446
- downloading, 38
- enabling debugging, 46
- installation tips, 38
- securing, 440–441
- server log, 45–46
- SSL connectors, 450
- starting, 41
- stopping, 42
- tools, 34–35

groups, 444

- managing, 444–446

H

- hashCode method, 340
- header parameters, 234
- helper classes, 258
 - session bean example, 276
- HTTP, 207
 - basic authentication, 462
 - over SSL, 465
- HTTP methods, 226–229
- HTTP request URLs, 185
 - query strings, 186
 - request paths, 185
- HTTP requests, 185
 - See also* requests
- HTTP responses, 186
 - See also* responses
- status codes, 67–68
- HTTPS, 437, 450, 451, 459–460
- HttpServletRequest interface, 180
- HttpServletRequest interface, 185, 471
- HttpServletResponse interface, 186

- HttpSession interface, 193

I

- identification, 434–435
- implicit navigation, 76
- include method, 192
- init method, 184
- InitialContext interface, 32
- initParams attribute, 184
- injectable objects, 308
- integrity, 449
 - of data, 434
- internationalizing JavaServer Faces applications,
 - FacesContext.getLocale method, 148
- invalidate method, 194
- isCallerInRole method, 493–494
- isUserInRole method, 471

J

- JAAS, 33, 435, 505
 - login modules, 505
- JACC, 30, 441
- JAF, 32
- JAR files, 17
 - query language, 396
- JAR signatures, 436
- JASPIC, 30–31
- Java API for JavaBeans Validation, *See* Bean Validation
- Java API for XML Binding, 33
- Java API for XML Processing, 32
- Java API for XML Web Services, *See* JAX-WS
- Java Authentication and Authorization Service, 435
 - See also* JAAS
- Java Authentication Service Provider Interface for Containers, 30–31
- Java Authorization Contract for Containers, 30
 - See also* JACC
- Java BluePrints, 44
- Java Cryptography Extension (JCE), 435

- Java Database Connectivity API, *See* JDBC API
- Java DB, 34
 - starting, 43
 - stopping, 43
- Java EE 6 platform, APIs, 22–31
- Java EE applications, 6–12
 - debugging, 45–46
 - deploying, 267–269
 - iterative development, 269
 - tiers, 6–12
- Java EE clients, 8–9
 - application clients, 8–9
 - See also* application clients
 - web clients, 49–71
 - See also* web clients
- Java EE components, 8
- Java EE Connector Architecture, 514
- Java EE Connector architecture, 29–30
- Java EE modules, 17, 18
 - See also* web modules
 - application client modules, 19
 - EJB modules, 19, 258
 - resource adapter modules, 19
- Java EE platform, 6–12
- Java EE security model, 13
- Java EE servers, 14
- Java EE transaction model, 13
- Java Generic Security Services, 435
- Java GSS-API, 435
- Java Message Service (JMS) API, 29, 514–515
 - See also* message-driven beans
- Java Naming and Directory Interface API, 31–32
 - See also* JNDI
- Java Persistence API, 28
- Java Persistence API query language, *See* query language
- Java Persistence Criteria API, *See* Criteria API
- Java Secure Sockets Extension, 435
- Java Servlet technology, 26, 179–200
 - See also* servlets
- Java Transaction API, *See* JTA
- JavaBeans Activation Framework, 32
- JavaBeans components, 9
- JavaMail API, 30
- JavaServer Faces application development, 77–81
 - backing beans, 159–162
 - bean property, 164
 - Bean Validation, 174–178
 - web pages, 113–144
- JavaServer Faces applications
 - HTML tags, 114–142
 - lifecycle, 79–80
 - queueing messages, 173
- JavaServer Faces core tag library, 113, 143
 - See also* validator tags
 - action attribute, 126
 - actionListener tag, 143, 152
 - attribute tag, 143
 - convertDateTime tag, 143, 147
 - convertDateTime tag attributes, 148–149
 - converter tag, 143, 147
 - converterId attribute, 146
 - convertNumber tag, 143, 147, 149–151
 - convertNumber tag attributes, 150–151
 - facet tag, 129, 143
 - loadBundle tag, 143
 - metadata tag, 140
 - param tag, 125, 143
 - selectItem tag, 116, 131, 133, 134, 143
 - selectItems tag, 116, 131, 133, 134, 143
 - type attribute, 151
 - validateDoubleRange tag, 144, 152
 - validateLength tag, 144, 152
 - validateLongRange tag, 144, 153, 154
 - validator tag, 144
 - valueChangeListener tag, 143, 151–152
 - viewparam tag, 140
- JavaServer Faces HTML tag library, *See* component tags
- JavaServer Faces tag libraries, 84
 - JavaServer Faces core tag library, 113, 143
 - JavaServer Faces HTML tag library, 113
 - namespace directives, 114
- JavaServer Faces technology, 10, 26–27, 73–81
 - See also* component tags
 - See also* Facelets
 - advantages, 75–76

JavaServer Faces technology (*Continued*)

- FacesContext class
 - Validator interface, 173
- features, 74–75
- JavaServer Pages Standard Tag Library, *See* JSTL
- javax.servlet.http package, 180
- javax.servlet package, 180
- JAX-RS, 28, 219–241
 - introduction, 204–205
 - other information sources, 240–241
 - reference implementation, 219
- JAX-WS, 33
 - defined, 207
 - examples, 208–216
 - introduction, 204
 - service endpoint interfaces, 208
 - specification, 217
- JAXB, 33
- JAXP, 32
- JCE, 435
- JDBC API, 31, 515, 530
- JNDI, 31–32, 529
 - data source naming subcontexts, 32
 - enterprise bean lookup, 252
 - enterprise bean naming subcontexts, 32
 - environment naming contexts, 32
 - naming contexts, 31
 - naming environments, 31
 - naming subcontexts, 32
- JSR 299, *See* Contexts and Dependency Injection (CDI) for the Java EE platform
- JSR 311, *See* JAX-RS
- JSSE, 435
- JSTL, 27
- JTA, 28
 - See also* transactions, JTA
- JTS API, 524

K

- Kerberos, 435, 436
- key pairs, 451
- keystores, 436, 450–453
 - managing, 451

- keytool utility, 451

L

- LDAP, 31
- lifecycle, JavaServer Faces, 79–80
- listener classes, 180
 - defining, 180
- listener interfaces, 180
- listeners
 - HTTP, 440
 - IIOP, 440
- local interfaces, defined, 254
- log, server, 45–46
- login
 - configuring, 461–468
- login configuration, 467–468
- login method, 469–471
- login modules, 505
- logout method, 469–471

M

- managed beans, defined for CDI, 307–308
- Managed Beans specification, 28, 305
- matrix parameters, 234
- message-driven beans, 25, 249–251
 - accessing, 249
 - defined, 249
 - garbage collection, 264
 - onMessage method, 250
 - transactions, 518, 524
- message listeners, JMS, 249
- message security, 457
- MessageBodyReader interface, 227–229
- MessageBodyWriter interface, 227–229
- messages
 - integrity, 465
 - MessageFormat pattern, 125, 143
 - outputFormat tag, 125
 - param tag, 125, 143
 - parameter substitution tags, 143
 - queueing messages, 173

- securing, 438
- metadata annotations
 - resource adapters, 538–540
 - security, 439
- Metamodel API, 415–417
 - using, 372, 417–418
- method expressions, 155
- method permissions, 489
 - annotations, 490–492
- mutual authentication, 465–466

N

- naming contexts, 31
- naming environments, 31
- navigation model
 - action attribute, 126, 154, 155
 - action methods, 171
 - ActionEvent class, 156
 - logical outcome, 171
 - referencing methods that perform
 - navigation, 155, 171
 - writing a backing bean method to perform
 - navigation processing, 171–172
- NDS, 31
- NetBeans IDE, 40
- NIS, 31
- non-repudiation, 434

O

- onMessage method, message-driven beans, 250

P

- package-appclient tool, 34
- parameters, extracting, 231–235
- path parameters, 233
- path templates, 223–226
- permissions, security policy, 441
- persistence
 - BLOBs, 364–365

- cascade operations, 363–364
- CLOBs, 364–365
- collections, 335–337
- configuration, 353
- context, 349–355
- embeddable classes, 344–345
- entities, 333–345
- examples, 357–380
- many-to-many, 369–370
- maps, 336
- one-to-many, 359
- one-to-one, 358–359
- overview, 333–356
- persistence units, 353–355
- persistent fields, 335
- primary keys, 339–341
 - compound, 361–363
 - generated, 360
- properties, 335
- queries, 333–356, 366–367, 382–384
 - See also* query language
 - creating, 418–419
 - Criteria, 415–426
 - dynamic, 382
 - executing, 425–426
 - expressions, 421–422, 422–423
 - joins, 420
 - parameters, 383
 - path navigation, 421
 - results, 421–423, 424–425
 - static, 382
 - typesafe, 415–426
- query language, 342
- relationships, 358–359
- scope, 353–355
- self-referential relationships, 358
- temporal types, 365
- persistence units
 - query language, 381, 396
- pluggable audit modules, 441
- pluggable authorization providers, 441
- POJOs, 4
- policy files, 436
- primary keys, 359

primary keys (*Continued*)

- compound, 361–363
 - defined, 339–341
 - examples, 340
 - generated, 360
- principal, 444
- PrintWriter class, 186
- producer methods, 314
- programmatic security, 430, 440, 456, 487
- proxies, 207
- public key certificates, 465
- public-key cryptography, 451

Q

- qualifiers, using, 309
- Quality of Service, 435
- query language
- ABS function, 407
 - abstract schemas, 382, 384, 396
 - ALL expression, 405
 - ANY expression, 405
 - arithmetic functions, 405–407
 - ASC keyword, 412
 - AVG function, 410
 - BETWEEN expression, 389, 402
 - Boolean literals, 400
 - Boolean logic, 408
 - case expressions, 407–408
 - collection member expressions, 397, 404
 - collections, 397, 404
 - compared to SQL, 386, 395, 398
 - comparison operators, 389, 402
 - CONCAT function, 406
 - conditional expressions, 388, 400, 401, 409
 - constructors, 411–412
 - COUNT function, 410
 - DELETE expression, 389, 390
 - DELETE statement, 385
 - DESC keyword, 412
 - DISTINCT keyword, 386
 - domain of query, 381, 394, 396
 - duplicate values, 386
 - enum literals, 400
 - equality, 409–410
 - ESCAPE clause, 403
 - examples, 366–367, 385–390
 - EXISTS expression, 405
 - FETCH JOIN operator, 398
 - FROM clause, 384, 394–398
 - grammar, 390–413
 - GROUP BY clause, 384, 412–413
 - HAVING clause, 384, 412–413
 - identification variables, 384, 394, 396
 - identifiers, 394–395
 - IN operator, 398, 402–403
 - INNER JOIN operator, 398
 - input parameters, 387, 401
 - IS EMPTY expression, 389
 - IS FALSE operator, 409
 - IS NULL expression, 388
 - IS TRUE operator, 409
 - JOIN statement, 386, 387, 397–398
 - LEFT JOIN operator, 398
 - LEFT OUTER JOIN operator, 398
 - LENGTH function, 406
 - LIKE expression, 388, 403
 - literals, 400
 - LOCATE function, 406
 - LOWER function, 406
 - MAX function, 410
 - MEMBER expression, 404
 - MIN function, 410
 - MOD function, 407
 - multiple declarations, 396
 - multiple relationships, 387
 - named parameters, 386, 401
 - navigation, 386–388, 388, 397, 399
 - negation, 409
 - NOT operator, 409
 - null values, 403–404, 408–409
 - numeric comparisons, 409
 - numeric literals, 400
 - operator precedence, 401–402
 - operators, 401–402
 - ORDER BY clause, 384, 412
 - parameters, 386
 - parentheses, 401

- path expressions, 382, 398–399
- positional parameters, 401
- range variables, 396–397
- relationship fields, 382
- relationships, 382, 386, 387
- return types, 410
- root, 397
- scope, 381
- SELECT clause, 384, 410–412
- setNamedParameter method, 386
- SIZE function, 407
- SQRT function, 407
- state fields, 382
- string comparison, 409
- string functions, 405–407
- string literals, 400
- subqueries, 404–405
- SUBSTRING function, 406
- SUM function, 411
- syntax, 384–385, 390–413
- TRIM function, 406
- types, 399, 409
- UPDATE expression, 385, 389, 390
- UPPER function, 406
- WHERE clause, 384, 400–410
- wildcards, 403
- query parameters, 232
- query roots, 419–420

R

- realms, 441, 443
 - admin-realm, 443
 - certificate, 443
 - adding users, 446
 - configuring, 440
 - file, 443
- referencing backing bean methods, 154–157
 - for handling action events, 156, 172
 - for handling value-change events, 156–157
 - for performing navigation, 155, 171
 - for performing validation, 156, 173
- relationship fields, query language, 382

- relationships
 - direction, 342–344
 - unidirectional, 359
- remote interfaces, defined, 256
- request method designator, 226–229
- request method designators, 220–235
- request parameters, extracting, 231–235
- RequestDispatcher interface, 191
- requests, 185
 - See also* HTTP requests
 - customizing, 188
 - getting information from, 185
- resource adapters, 29–30, 514, 534–538
 - metadata annotations, 538–540
 - security, 507–508
- resource classes, 220–235
- resource injection, 531–534
- resource methods, 220–235
- resources, 514–515, 529–541
 - See also* data sources
- ResponseBuilder class, 227–229
- responses, 186
 - See also* HTTP responses
 - buffering output, 186
 - customizing, 188
 - setting headers, 184
- RESTful web services, 28, 219–241
 - defined, 219–220
- roles, 444
 - application, 447–448
 - declaring, 468–469
 - mapping to groups, 447–448
 - mapping to users, 447–448
 - referencing, 490–492
 - security, 446–447, 468–469, 489, 490–492
- rollback method, 523, 525
- rollbacks, *See* transactions, rollbacks
- root resource classes, 220
- run-as identity, 494–496

S

- SAAJ, 33
- SASL, 435

schema, deployment descriptors, 439–440

schemagen tool, 34

scopes, using, 310–312

secure connections, 449–453

Secure Socket Layer (SSL), 449–453

security

- annotations, 439, 476–477, 486

- web applications, 456

- application, 436–437

- characteristics of, 434–435

- application client tier

- callback handlers, 505

- application clients, 504–505

- callback handlers, 504, 505

- constraints, 457–461

- container trust, 496

- containers, 430–435, 439–440

- context

- enterprise beans, 493–494

- declarative, 430, 439–440, 456, 486

- deploying enterprise beans, 496

- EIS applications, 506–509

- component-managed sign-on, 506–507

- container-managed sign-on, 506

- end-to-end, 438

- enterprise beans, 486–496

- example, 430–433

- groups, 444

- introduction, 429–454

- JAAS login modules, 505

- Java SE, 435–436

- login forms, 504

- login modules, 505

- mechanism features, 433–434

- mechanisms, 435–438

- message, 457

- message-layer, 438

- method permissions, 489

- annotations, 490–492

- policy domain, 444

- programmatic, 430, 440, 456, 469–474, 487

- propagating identity, 494–496

- realms, 443

- resource adapters, 507–508

- role names, 468–469, 490–492

- roles, 444, 446–447, 468–469, 489

- run-as identity, 494–496

- transport-layer, 437–438, 449–453

- users, 443

- web applications, 455–484

- overview, 455

- web components, 455–484

- security constraints, 457–461

- multiple, 460–461

- security domain, 444

- security identity

- propagating, 494–496

- specific identity, 495

- security-role-mapping element, 447–448

- security-role-ref element, 473–474

- security role references, 473–474

- security roles, 446–447, 489

- server, authentication, 465

- server log, 45–46

- servers, certificates, 450–453

- service methods, servlets, 184

- Servlet interface, 180

- ServletContext interface, 193

- ServletInputStream class, 185

- ServletOutputStream class, 186

- ServletRequest interface, 185

- ServletResponse interface, 186

- servlets, 10, 180

- binary data, 185, 186

- character data, 185, 186

- compiling, 267–269

- creating, 183–184

- examples, 62–70, 198–199, 266

- finalizing, 195

- initializing, 184

- lifecycle, 180–182

- lifecycle events, 180

- packaging, 267–269

- service methods, 184, 196, 197

- tracking service requests, 196

- session beans, 25, 247–249

- activation, 261

- bean-managed concurrency, 280, 282–283

- business interfaces, 251
- clients, 247
- concurrent access, 280–283
- container-managed concurrency, 280
- databases, 523
- eager initialization, 278
- examples, 266, 271–278, 278–286, 286–289
- handling errors, 283
- no-interface views, 251
- passivation, 261
- requirements, 273
- singleton, 248, 278–286
- stateful, 247, 248
- stateless, 247–248, 249
- transactions, 518, 523, 524
- web services, 257, 287–288
- sessions, 193–195
 - associating attributes, 193–194
 - associating with user, 195
 - invalidating, 194
 - notifying objects associated with, 194
- SessionSynchronization interface, 523
- setRollbackOnly method, 523, 525
- sign-on
 - component-managed, 506
 - container-managed, 506
- Simple Authentication and Security Layer, 435
- SingleThreadModel interface, 183
- SOAP, 203–205, 207, 217
- SOAP messages, 16, 33
 - securing, 438
- SOAP with Attachments API for Java, *See* SAAJ
- SQL, 31, 386, 395, 398
- SQL92, 408
- SSL, 437, 449–453, 459–460, 465
 - connectors
 - GlassFish Server, 450
 - handshake, 449
 - verifying support, 450
- standard converters
 - converter tags, 143, 147
 - NumberConverter class, 146
 - using, 145–151

- standard validators
 - See also* validator tags
 - using, 152–154
- state fields, query language, 382
- substitution parameters, defining, *See* messages, param tag

T

- templating, Facelets, 91–93
- timer service, 290–300
 - automatic timers, 290, 294–295
 - calendar-based timer expressions, 290–293
 - cancelling timers, 296
 - creating timers, 293–294
 - examples, 297–299
 - exceptions, 296
 - getInfo method, 296
 - getNextTimeout method, 296
 - getTimeRemaining method, 296
 - getting information, 296
 - programmable timers, 290, 293–294
 - saving timers, 296
 - transactions, 296–297
- transactions, 513, 517–528
 - application-managed, 350–351
 - attributes, 519–522
 - bean-managed, 524–525, 525
 - boundaries, 518, 523, 524
 - business methods
 - See* business methods, transactions
 - commits, 518, 523
 - container-managed, 518–523
 - container-managed transaction
 - demarcation, 518
 - defined, 518
 - exceptions
 - See* exceptions, transactions
 - JDBC, 526
 - JTA, 524
 - managers, 521, 524, 526–527
 - message-driven beans, 250
 - See also* message-driven beans, transactions
 - nested, 518, 524

transactions (*Continued*)

- rollbacks, 518, 523, 525
 - scope, 519
 - session beans
 - See* session beans, transactions
 - timeouts, 525–526
 - timer service, 296–297
 - web components, 528
- transport-guarantee element, 459–460
- transport guarantees, 459–460
- transport-layer security, 437–438, 449–453
- truststores, 450–453
- managing, 451

U

- UnavailableException class, 184
- undeploying, modules and applications, 61–62
- unified expression language, *See* EL
- Uniform Resource Identifiers (URIs), 219
- URI path parameters, 233
- URI path templates, 223
- user-data-constraint element, 459–460
- user data constraints, 458, 459–460
- users, 443
- adding to GlassFish Server, 445–446
 - managing, 444–446
- UserTransaction interface, 523, 525, 528
- using pages, 96
- utility classes, 258

V

- validation, entities, 337–339
- validation model
- referencing a method that performs validation, 156
 - validator attribute, 122, 155, 156, 173
 - Validator interface, 170, 173
 - writing a backing bean method to perform validation, 173
- Validator implementation classes, 152–153
- DoubleRangeValidator class, 144, 152

- LengthValidator class, 144, 152
 - LongRangeValidator class, 144, 153, 154
- validator tags, 144
- validateDoubleRange tag, 152
 - validateLength tag, 152
 - validateLongRange tag, 153, 154
- validators
- custom validators, 144
 - registering, 153–154
- value binding
- acceptable types of component values, 163
 - properties, 163–168
 - value attribute, 162
 - value expressions, 164
- value-change events
- processValueChangeEvent method, 174
 - referencing methods that handle value-change events, 156–157
 - type attribute, 151
- ValueChangeEvent class, 151
- valueChangeListener attribute, 122, 155, 173
- ValueChangeListener class, 151–152, 174
- valueChangeListener tag, 143, 151–152
- writing a backing bean method to handle value-change events, 173–174
- value expressions, 161
- ValueExpression class, 162

W

- W3C, 32, 207, 217
- WAR files, 17
- web applications, 53
- configuring, 51, 62
 - deployment descriptors, 51
 - document roots, 53
 - maintaining state across requests, 193–195
 - presentation-oriented, 49
 - securing, 455–484
 - security
 - overview, 455
 - service-oriented, 49
 - specifying context parameters, 66–67
 - specifying initialization parameters, 67

- specifying welcome files, 66
 - web beans, *See* Contexts and Dependency Injection (CDI) for the Java EE platform
 - web clients, 8, 49–71
 - examples, 266
 - web components, 10, 50–51
 - See also* Java EE components
 - applets bundled with, 10
 - concurrent access to shared resources, 183
 - forwarding to other web components, 192
 - including other web resources, 192
 - invoking other web resources, 191
 - mapping exceptions to error screens, 67–68
 - mapping filters to, 189
 - scope objects, 182
 - securing, 455–484
 - sharing information, 182
 - transactions, 528
 - types, 10
 - utility classes bundled with, 10
 - web context, 193
 - web containers, 15, 51
 - loading and initializing servlets, 180
 - mapping URLs to web components, 62
 - web modules, 19, 53
 - deploying, 59
 - dynamic reloading, 60
 - undeploying, 61–62
 - updating, 60
 - viewing deployed, 60
 - web pages
 - XHTML, 78, 84
 - web-resource-collection element, 458–459
 - web resource collections, 458
 - web resources, 53
 - Facelets, 96–97
 - mapping filters to, 189
 - unprotected, 458
 - web services, 15–16
 - See also* enterprise beans, web services
 - declaring references to, 70
 - endpoint implementation classes, 287
 - examples, 208–216, 286–289
 - introduction, 203
 - JAX-RS compared to JAX-WS, 203–205
 - web.xml file, 53, 440, 488
 - welcome files, specifying, 66
 - work flows, 248
 - writing backing bean methods, 170–174
 - for handling action events, 172–173
 - for handling value-change events, 173–174
 - for performing navigation, 171–172
 - for performing validation, 173
 - writing backing bean properties
 - converters, 170
 - listeners, 170
 - validators, 170
 - WSDL, 16, 203–205, 207, 217
 - wsgen tool, 35
 - wsimport tool, 35
- X**
- xjc tool, 34
 - XML, 15–16, 207