# Beginning JSF<sup>™</sup> 2 APIs and JBoss<sup>®</sup> Seam



Kent Ka lok Tong

**Apress**<sup>®</sup>

#### Beginning JSF<sup>™</sup> 2 APIs and JBoss<sup>®</sup> Seam

#### Copyright © 2009 by Kent Ka lok Tong

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-1922-4

ISBN-13 (electronic): 978-1-4302-1923-1

Printed and bound in the United States of America 987654321

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Java<sup>™</sup> and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the US and other countries. Apress, Inc., is not affiliated with Sun Microsystems, Inc., and this book was written without endorsement from Sun Microsystems, Inc.

Lead Editors: Steve Anglin, Matt Moodie Technical Reviewer: Jim Farley Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh Project Manager: Sofia Marchant Copy Editors: Kim Wimpsett and Heather Lang Associate Production Director: Kari Brooks-Copony Production Editor: Ellie Fountain Compositor and Artist: Kinetic Publishing Services, LLC Proofreader: Patrick Vincent Indexer: Toma Mulligan Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at http://www.apress.com. You will need to answer questions pertaining to this book in order to successfully download the code.

# **Contents at a Glance**

About the Author.	ix
About the Technic	cal Reviewer
CHAPTER 1	Getting Started with JSF1
CHAPTER 2	Using Forms
CHAPTER 3	Validating Input
CHAPTER 4	Creating an E-shop101
CHAPTER 5	Creating Custom Components151
CHAPTER 6	Providing a Common Layout for Your Pages173
CHAPTER 7	Building Interactive Pages with Ajax183
CHAPTER 8	Using Conversations
CHAPTER 9	Supporting Other Languages
CHAPTER 10	Using JBoss Seam
INDEX	

# Contents

About the Author About the Techni	ix ical Reviewer
CHAPTER 1	Getting Started with JSF1
	Introducing the "Hello world" Application
	Installing Eclipse2
	Installing JBoss3
	Installing a JSF Implementation7
	Installing Web Beans8
	Creating the "Hello world!" Application with JSF9
	Generating Dynamic Content
	Retrieving Data from Java Code
	Exploring the Life Cycle of the Web Bean25
	Using an Easier Way to Output Text
	Debugging a JSF Application25
	Summary
CHAPTER 2	Using Forms
	Developing a Stock Quote Application
	Getting the Stock Quote Symbol
	Displaying the Result Page
	Displaying the Stock Value
	Marking Input As Required40
	Inputting a Date
	Conversion Errors and Empty Input
	Using a Combo Box
	Using a Single b2 Bean62
	Hooking Up the Web Beans63
	Summary

<b>CHAPTER 3</b>	Validating Input	37
	Developing a Postage Calculator       6         What If the Input Is Invalid?       7         Null Input and Validators.       7         Validating the Patron Code       8         Creating a Custom Validator for the Patron Code       8         Displaying the Error Messages in Red       8         Displaying the Error Message Along with the Field       8         Validating a Combination of Multiple Input Values       9         Summary.       10	57 78 30 32 36 37 96
CHAPTER 4	Creating an E-shop10	)1
	Listing the Products.10Making the Link to Show the Details10Displaying Headers in the Columns11Implementing a Shopping Cart11Displaying the Content of the Shopping Cart12The Checkout Function12Getting the Credit Card Number of the Current User13Forcing the User to Log In13Implementing Logout14Summary.14	)2 )6 15 16 26 27 31 39 16 18
CHAPTER 5	Creating Custom Components15	51
	Displaying a Copyright Notice on Multiple Pages.15Allowing the Caller to Specify the Company Name15Creating a Product Editor15Passing a Method in a Parameter?16Creating a Box Component16Accepting Two Pieces of XHTML Code16Creating a Reusable Component Library16Creating a Component Library17Summary17	51 57 59 52 53 56 58 70 72

CHAPTER 6	Providing a Common Layout for Your Pages
	Using the Same Menu on Different Pages.173Using Global Navigation Rules.177Using Two Abstract Parts.178Creating Page-Specific Navigation Cases180Summary.182
CHAPTER 7	Building Interactive Pages with Ajax
	Displaying a FAQ183Refreshing the Answer Only.185Hiding and Showing the Answer189Using Ajax to Hide or Show the Answer.191Giving a Rating to a Question194Updating the Rating as the User Types199Using a Dialog Box to Get the Rating200Setting the Look and Feel with Skins.204Displaying Multiple Questions206Summary.212
CHAPTER 8	Using Conversations
	Creating a Wizard to Submit Support Tickets
CHAPTER 9	Supporting Other Languages

CHAPTER 10	Using JBoss Seam	253
	Installing Seam.	253
	Re-creating the E-shop Project	254
	Allowing the User to Add Products	257
	Restricting Access to the Product-Editing Page	265
	Creating a Shopping Cart	267
	Turning the Shopping Cart into a Stateful Session Bean	273
	Creating the Checkout Page	277
	Using WebLogic, WebSphere, or GlassFish	284
	Summary	284
INDEX		287

# About the Author

**KENT KA IOK TONG** is the manager of the IT department of the Macau Productivity and Technology Transfer Center. With a master's degree in computer science from the University of New South Wales in Sydney, Australia, and having won the Macao Programming Competition (Open Category) in 1992, Kent has been involved in professional software development, training, and project management since 1993. He is the author of several popular books on web technologies including *Essential JSF*, *Facelets and Seam*, *Enjoying Web Development with Tapestry*, *Enjoying Web Development with Wicket*, and *Developing Web Services with Apache Axis 2*.

# About the Technical Reviewer

JIM FARLEY is a technology architect, strategist, writer, and manager. His career has touched a wide array of domains, from commercial to nonprofit and from finance to higher education. In addition to his day job, Jim teaches enterprise development at Harvard University. Jim is the author of several books on technology and contributes articles and commentary to various online and print publications.

# CHAPTER 1

# **Getting Started with JSF**

n this chapter you'll learn how to set up a development environment and create a "Hello world!" application with JSF.

# Introducing the "Hello world" Application

Suppose that you'd like to develop the application shown in Figure 1-1.



Figure 1-1. A simple "Hello world!" application with a single page

To do that, you'll need to install some software (see Figure 1-2). First, you'll need an IDE to create your application. This book will use Eclipse, but other popular IDEs will do just fine too. Next, you'll need to install JBoss, which provides a platform for running web applications (there are also fine alternatives to JBoss). In addition, your application will use JSF and Web Beans as libraries. So, you'll need to download them too.



Figure 1-2. The software that you'll need

## **Installing Eclipse**

You need to make sure you have the Eclipse IDE for Java EE Developers, as shown in Figure 1-3 (note that the Eclipse IDE for Java Developers is *not* enough, because it doesn't include tools for developing web applications). You can go to http://www.eclipse.org to download it. For example, you'll need the eclipse-jee-ganymede-SR1-win32.zip file if you use Windows. Unzip it into a convenient location, such as c:\eclipse. Then, create a shortcut to run c:\eclipse\eclipse -data c:\workspace. This way, it will store your projects under the c:\workspace folder.



Figure 1-3. Getting the right bundle of Eclipse

To see whether it's working, run it, and make sure you can switch to the Java EE perspective (it should be the default; if not, choose Window > Open Perspective > Other), as shown in Figure 1-4.

[e]	Open Perspective	
le cvs	Repository Exploring	
ll Dat	abase Debug	
ြ Dat	abase Development	=
≫ Deb	ug	
ତ Hibe	ernate	
<b></b> ≣Java	1	
💱 Java	Browsing	
🕆 Java	i EE (default)	
°⊧' Java	Type Hierarchy	

Figure 1-4. The Java EE perspective

### **Installing JBoss**

To install JBoss, go to http://www.jboss.org/jbossas/downloads to download a binary package of JBoss Application Server 5.x (or newer), such as jboss-5.0.1.GA.zip. Unzip it into a folder such as c:\jboss. To test whether it is working, you can try to launch JBoss in Eclipse. To do that, choose Windows > Preferences in Eclipse, and then choose Server > Installed Runtime Environments. You'll see the window shown in Figure 1-5.

	Preferences	(A) (A)
type filter text	Server Runtime Environments	φ•φ••
• General • Ant	<ul> <li>Add, remove, or edit server run Server runtime environments:</li> </ul>	itime environments.
Data Management	Name Type	<u>A</u> dd
Help		Edit
HQL editor		Remove
Install/Update		
⊦ Java		Search
JavaScript		
<ul> <li>JBoss Tools</li> </ul>		
JPA		
JSP Occurrences	8	
• Maven		
<ul> <li>Plug-in Development</li> </ul>		
<ul> <li>Remote Systems</li> </ul>		
Run/Debug		
<ul> <li>Server</li> </ul>		
Audio		
Launching		
Runtime Environments		

Figure 1-5. The installed runtime environments

Click Add, and choose JBoss ➤ JBoss v5.0 (Figure 1-6).

New Server Runtime Environment	
New Server Runtime Environment	
Define a new server runtime environment	
Download additional server	r adapters
Select the type of <u>r</u> untime environment:	
type filter text	
▶ ≌ Apache	
→ 🗁 Basic	
→ l∋ IBM	
- ⊱ JBoss	
🗄 JBoss v3.2.3	=
🗄 JBoss v4.0	
Boss v4.2	
≣ JBoss v5.0	
▶	
→ 🗁 Oracle	•
Publishes and runs J2EE 5 modules on a local server. Provide server functionality.	s basic
□ <u>C</u> reate a new local server	
② < Back Next > Einish Compared to the second se	ancel

Figure 1-6. The JBoss 5.0 runtime

Click Next. Specify c:\jboss as the application server directory (Figure 1-7).

New New	Server Runtime Environment			
New JBoss v5.0 Runtime				
Define a new JBoss v5.0 runtime				
You can use Installed JRE pret	ferences to create a new JRE			
JRE:	Default JRE	•		
Application Server Directory:	c:\jboss	Browse		
⑦ < Back	Next > Finish	Cancel		

Figure 1-7. Specifying the JBoss application server directory

Click Finish. Next, you need to create a JBoss instance. In the bottom part of the Eclipse window, you'll see a Servers tab (you'll see this tab only when you're in the Java EE perspective); right-click anywhere on the tab, choose New ➤ Server, and choose the JBoss v5.0 server runtime environment (Figure 1-8).

	New Server	
Define a New Server		
Choose the type of s	erver to create	
Server's <u>h</u> ost name:	localhost	
	Download additional	server adapters
Select the <u>s</u> erver type	e:	
type filter text		
Boss v3.2.3		
Boss V4.0		
Boss V4.2		
E JBoss V5.0		
		•
Publishes and runs JE	E 5 modules on a local server. Pro	ovides basic
server functionality.		
Server na <u>m</u> e:	JBoss v5.0 at localhost	
Server <u>r</u> untime envir	onment: JBoss v5.0	<u>Add</u>
	Configure runtime	environments
⑦ < <u>Back</u>	<u>N</u> ext > <u>F</u> inish	Cancel

Figure 1-8. Choosing the JBoss runtime environment

Click Next until you see the screen in Figure 1-9, where you can add web applications to the JBoss instance.

C Add and Remove Projects	New Server			
Modify the projects that are	Modify the projects that are configured on the server			
Move projects to the right to	configure them on the	server		
<u>A</u> vailable projects:	You can add selected projects to that JBoss instance.	<u>C</u> onfigured projects:		
	A <u>d</u> d >			
	< <u>R</u> emove			
L				
If you had web application projects in Eclipse, you	Add A <u>I</u> I >>			
would see them listed here.	<< Remove All			
			]	
? <b>S</b>	ack <u>N</u> ext >	<u>F</u> inish Ca	ncel	

Figure 1-9. Adding web applications

For the moment, you'll have none. Click Finish. Then you should see your JBoss instance on the Servers tab (Figure 1-10).

	To run it, click the green butt	on he	ere.			
🖳 Console 🗟 Problems 🖗 Se	rvers 🛛 🗆 Propertie	A	Sei	arch	ן פ	- 0
		菸	0	\$	ļ	80
Server	State	St	tatı	ıs		
🛙 JBoss v5.0 at localhost	🖡 Stopped					
	To stop it, click the	e red	butto	on he	re.	

Figure 1-10. JBoss instance

Click the green icon as shown in Figure 1-10 to run JBoss. Then you will see some messages on the Console tab, as shown here:

```
...
14:47:06,820 INFO [TomcatDeployment] deploy, ctxPath=/
14:47:06,902 INFO [TomcatDeployment] deploy, ctxPath=/jmx-console
14:47:06,965 INFO [Http11Protocol] Starting Coyote HTTP/1.1 on http-127.0.0.1-8080
14:47:06,992 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
14:47:07,001 INFO [ServerImpl] JBoss (Microcontainer) [5.0.1.GA (build:
SVNTag=JBoss_5_0_1_GA date=200902231221)] Started in 26s:587ms
```

**Note** If your computer is not that fast, JBoss will take so long to start that Eclipse may think it has stopped responding. In that case, double-click the JBoss instance, click Timeouts, set the timeout for starting to a longer value such as 100 seconds, and then start JBoss again.

To stop JBoss, click the red icon (as shown earlier in Figure 1-10).

### Installing a JSF Implementation

JSF stands for JavaServer Faces and is an API (basically, it's some Java interfaces). To use JSF, you need an implementation (which means you need Java classes that implement those interfaces). There are two main implementations: the reference implementation from Sun and MyFaces from Apache. In this book, you'll use the former, but you could use MyFaces with no practical difference.

So, go to https://javaserverfaces.dev.java.net to download a binary package of the JSF 2.0 implementation, which is called Mojarra. The file is probably called something like mojarra-2.0.0-PR2-binary.zip; unzip it into a folder, say c:\jsf.

### **Installing Web Beans**

To install Web Beans, go to http://www.seamframework.org/WebBeans to download it. Make sure it is strictly newer than 1.0.0 ALPHA2; otherwise, get the nightly snapshot. The file is probably called something like webbeans-ri-distribution-1.0.0-SNAPSHOT.zip; unzip it into a folder such as c:\webbeans.

Next, you'll need to install Web Beans into JBoss. To do that, you'll need to run Ant 1.7.0 or newer. If you don't have this tool, you can download it from http://ant.apache.org and unzip it into a folder such as c:\ant.

Next, modify the c:\webbeans\jboss-as\build.properties file to tell it where JBoss is, as shown in Listing 1-1. Make sure that there is no leading # character on that line!

#### Listing 1-1. Tell Web Beans Where JBoss Is

```
jboss.home=c:\jboss
java.opts=...
webbeans-ri-int.version=5.2.0-SNAPSHOT
webbeans-ri.version=1.0.0-SNAPSHOT
jboss-ejb3.version=1.0.0
```

Open a command prompt, make sure you're connected to the Internet, and then issue the commands shown in Listing 1-2.

#### Listing 1-2. Issue These Commands at the Command Prompt

```
c:\>cd \webbeans\jboss-as
c:\>set ANT_HOME=c:\ant
c:\>ant update
```

This will output a lot of messages. If everything is fine, you should see a "BUILD SUC-CESSFUL" message at the end, as shown here:

• • •

```
[copy] Copying 2 files to /home/kent/jboss-
```

5.0.1.GA/server/default/deployers/webbeans.deployer/lib-int

[copy] Copying 8 files to /home/kent/jboss-

5.0.1.GA/server/default/deployers/webbeans.deployer

update:

BUILD SUCCESSFUL

# Creating the "Hello world!" Application with JSF

To create the "Hello world!" application, right-click in Package Explorer, and choose New ➤ Dynamic Web Project (Figure 1-11).

<u>N</u> ew	•	🛱 P <u>r</u> oject	
Show In	Shift+Alt+W >	B Application Client P	roject
Copy	Ctrl+C	Connector Project	
Copy Qualified N	lame	🛎 Dynamic Web Proje	ct
Paste	Ctrl+V	S EJB Project	
X Delete	Delete	C Enterprise Applicati	on Project
🗈 Paste	Ctrl+V	🗈 Example	
Import	,	🛱 <u>O</u> ther	Ctrl+N
🖆 Export			
8) Refresh	F5		

Figure 1-11. Creating a dynamic web project

Enter the information shown in Figure 1-12.

New Dynamic Web Project				
Dynamic Web Project				
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.				
Project name: Hello —				
Project contents: The name doesn't really matter.				
Directory: /home/kent/Books/EssentialJSF/v20/workspace/ Browserm				
Target Runtime				
JBoss v5.0 — Run this application in JBoss.				
Dynamic Web Module version				
2.5  +				
Configuration				
Default Configuration for JBoss v5.0				
A good starting point for working with JBoss v5.0 runtime. Additional facets can later be installed to add new functionality to the project.				
EAR Membership				
□ Add project to an EAR				
Image: Contract of the second seco				

Figure 1-12. Entering the project information

Keep clicking Next until you finish. Finally, you should end up with the project structure shown in Figure 1-13.



Figure 1-13. Project structure

To make JAR files from the JSF implementation available to your project, copy the JAR files into JBoss, as shown in Figure 1-14.



Figure 1-14. Copying the JAR files into the JBoss

To see the Web Beans classes available to you at compile time, right-click the project, choose Build Path > Configure Build Path, and add c:\jboss\server\default\deployers\ webbeans.deployer\jsr299-api to the build path.

Next, you'll create the "Hello world!" page. To do that, right-click the WebContent folder, and choose New ➤ HTML. Enter **hello** as the file name, as in Figure 1-15.

🕒 New HTML Page 💷 💽	×			
HTML Page				
Create a new HTML Page.				
Enter or select the parent folder:				
Hello/WebContent				
- ≌ Hello				
≥ .settings				
▶ 🗁 build				
🗁 src				
🔸 🚔 WebContent				
→ ﷺ JSFTest				
• 🗁 Servers				
File name: hello				
Advanced >>				
Reck Next > Finish Cancel				

Figure 1-15. Creating the "Hello world!" page

Click Next, and choose the template named New XHTML File (1.0 Strict), as in Figure 1-16.

New HTM	L Page			
Select HTML Template				
Select a template as initial content in the HTML page.				
-				
☑ Use HTML Template				
Templates are 'New HTML' templates found in the <u>HTML Templates</u> preference page.				
Name	Description			
New HTML File (4.01 strict)	html 4.01 strict			
New HTML File (4.01 transition	ه html 4.01 transitional			
New XHTML File (1.0 frameset)	xhtml 1.0 frameset			
New XHTML File (1.0 strict)	xhtml 1.0 strict			
New XHTML File (1.0 transition	i xhtml 1.0 transitional			
Preview				
<pre><?xml version="1.0" encoding="\${encoding}" ?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org chtml xmlns="http://www.w3.org/1999/xhtml">    </pre>				
<pre>deta http-equiv="Content-Type" content="text/html; charset=\${encoding}" /&gt; etitlo=Incent titlo_hone(titlo=</pre>				
<pre><body> \$foursor}</body></pre>				
	•••			

Figure 1-16. Using the XHTML strict template

Click Finish. This will give you a file named hello.html. This XHTML file will serve as the "Hello world!" page. However, JSF by default assumes that XHTML files use the .xhtml extension, so rename the file as hello.xhtml (see Figure 1-17).

<b>C</b> )	Renam	e Resource	
New na <u>m</u> e:	hello.xhtm	I	
Pre	view >	OK	Cancel

Figure 1-17. Renaming the file

Open the file, and input the content shown in Listing 1-3.

### Listing 1-3. Content of hello.xhtml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Insert title here</title>
</head>
<body>
Hello world!
</body>
</html>
```

Next, modify the web.xml file in the WebContent/WEB-INF folder as shown in Figure 1-18.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"</pre>
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app 2 5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app 2 5.xsd" id="WebApp ID"
version="2.5">
  <display-name>Hello</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
                          You can give it any name
  <servlet>

    This "servlet" is the JSF engine.

                          vou'd like.
   <servlet-name>JSF</servlet-name>
   <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  </servlet>
  <servlet-mapping>
                                                You will access the application
   <servlet-name>JSF</servlet-name>
                                                using a URL like this. This way,
                                                JBoss will send the request to
   <url-pattern>/faces/*</url-pattern>
                                                the JSF engine for handling.
  </servlet-mapping>
</web-app>
This "servlet" is the JSF engine.
                                   http://localhost:8080/Hello/faces/???
You can give it any name
you'd like.
                           The Project Name
                  Hello
                       WebContent
```



Next, create a file called faces-config.xml in the WebContent/WEB-INF folder. This is the configuration file for JSF, as shown in Listing 1-4. Without it, JSF will not initialize. Because you have no particular configuration to set, it contains only an empty <faces-config> element.

#### Listing 1-4. faces-config.xml

```
<faces-config 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-facesconfig_2_0.xsd"
    version="2.0">
```

</faces-config>

To register your application with JBoss, right-click the JBoss instance on the Servers tab, and choose Add and Remove Projects; then you'll see Figure 1-19.

E	dd and Remove Projects				
Add and Remove Projects Modify the projects that are configured on the server					
Move projects to the right to configure them on the server					
<u>Available projects:</u>		<u>C</u> onfigured projects:			
Normal Hello	Add > < < Remove				
@ < <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish Cancel			

Figure 1-19. Adding projects to the JBoss instance

Choose your Hello project to add to the JBoss instance.

Now, start JBoss, and try to access http://localhost:8080/Hello/hello.xhtml in a browser. Note that this URL does *not* include the /faces prefix and thus will *not* be handled by the JSF engine. Instead, JBoss will directly read the hello.xhtml page and return its content (see Figure 1-20). We're doing this just to check whether the basic web application is working.



Figure 1-20. Directly accessing the content of hello.xhtml

If everything is working, the browser should either prompt you to save the file (Firefox) or display the "Hello world!" page (Internet Explorer).

To access it through the JSF engine, use http://localhost:8080/Hello/faces/hello.xhtml instead, as shown in Figure 1-21. Simply put, JSF will take path /hello.xhtml (the view ID) from the URL and use it to load the XHTML file.

17



Figure 1-21. Accessing the hello.xhtml file through JSF

You'll see "Hello world!" displayed in the browser.

### **Generating Dynamic Content**

Displaying static text is not particularly interesting. Next, you'll learn how to output some dynamic text. Modify hello.xhtml as shown in Figure 1-22. The page object created is also shown in the figure.



Figure 1-22. JSF component tree

The component tree generates HTML code, as shown in Figure 1-23. In JSF, the process is called *encoding*.

19



Figure 1-23. JSF component tree generating HTML code

Now access the page again in the browser. Do you need to start JBoss again? No. By default Eclipse will update the web application in JBoss every 15 seconds after you make changes to the source files. If you can't wait, you can right-click the JBoss instance and choose Publish to force it to do it immediately. Anyway, the HTML page should look like Figure 1-24.



Figure 1-24. Generated HTML code

Note that there is no space between "Hello" and "John." This is because JSF ignores the spaces surrounding JSF tags. You can easily fix this problem, but let's ignore it for now; we'll fix it later in the chapter.

## **Retrieving Data from Java Code**

Next, you'll let the UI Output component retrieve the string from Java code. First, create the Java class GreetingService in the hello package. Input the content shown in Listing 1-5.

```
Listing 1-5. GreetingService.java
package hello;
public class GreetingService {
    public String getSubject() {
        return "Paul";
    }
}
```

So, how do you get the UI Output component to call the getSubject() method in the class? Figure 1-25 shows how it works. Basically, in each HTTP request, there is a table of objects, and each object has a name. (Each object is called a *web bean*.) If you set the value attribute of the UI Output component to something like #{foo}, which is called an *EL expression*, at runtime it will ask the JSF engine for an object named foo. The JSF engine will in turn ask the Web Beans manager for an object named foo.



Figure 1-25. Accessing a web bean

For your current case, what if Object1 were a GreetingService object (let's ignore how to create one of those for the moment)? Then the UI Output component can already reach the GreetingService object. How can the output call the getSubject() method on it? To do that, modify the value attribute of the outputText tag as shown in Listing 1-6.