Beginning JSF™ 2 APIs and JBoss® Seam

Kent Ka lok Tong
# Contents at a Glance

About the Author ................................................................... ix
About the Technical Reviewer ....................................................... xi

| CHAPTER 1 | Getting Started with JSF. | 1 |
| CHAPTER 2 | Using Forms | 29 |
| CHAPTER 3 | Validating Input | 67 |
| CHAPTER 4 | Creating an E-shop | 101 |
| CHAPTER 5 | Creating Custom Components | 151 |
| CHAPTER 6 | Providing a Common Layout for Your Pages | 173 |
| CHAPTER 7 | Building Interactive Pages with Ajax | 183 |
| CHAPTER 8 | Using Conversations | 215 |
| CHAPTER 9 | Supporting Other Languages | 231 |
| CHAPTER 10 | Using JBoss Seam | 253 |

INDEX ........................................................................ 287
CHAPTER 3  Validating Input ............................................. 67

Developing a Postage Calculator ........................................ 67
What If the Input Is Invalid? ........................................ 73
Null Input and Validators ........................................ 78
Validating the Patron Code ........................................ 80
Creating a Custom Validator for the Patron Code .................. 82
Displaying the Error Messages in Red ................................ 86
Displaying the Error Message Along with the Field ............... 87
Validating a Combination of Multiple Input Values ............... 96
Summary ...................................................... 100

CHAPTER 4  Creating an E-shop ........................................... 101

Listing the Products .................................................. 102
Making the Link to Show the Details ............................... 106
Displaying Headers in the Columns .................................. 115
Implementing a Shopping Cart ...................................... 116
Displaying the Content of the Shopping Cart ....................... 126
The Checkout Function ............................................. 127
Getting the Credit Card Number of the Current User .......... 131
Forcing the User to Log In ......................................... 139
Implementing Logout ............................................... 146
Protecting the Password .......................................... 148
Summary ...................................................... 149

CHAPTER 5  Creating Custom Components ............................ 151

Displaying a Copyright Notice on Multiple Pages ............... 151
Allowing the Caller to Specify the Company Name ............. 157
Creating a Product Editor ......................................... 159
Passing a Method in a Parameter? ................................. 162
Creating a Box Component ....................................... 163
Accepting Two Pieces of XHTML Code ......................... 166
Creating a Reusable Component Library ....................... 168
Creating a Component Library Without taglib.xml ............ 170
Summary ...................................................... 172
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
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.
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.
In 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.

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.

![Figure 1-4. The Java EE perspective](image)

### 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.

![Figure 1-5. The installed runtime environments](image)
Click Add, and choose JBoss ➤ JBoss v5.0 (Figure 1-6).

![Figure 1-6. The JBoss 5.0 runtime](image)

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

![Figure 1-7. Specifying the JBoss application server directory](image)
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).

![Figure 1-8. Choosing the JBoss runtime environment](image)

Click Next until you see the screen in Figure 1-9, where you can add web applications to the JBoss instance.
If you had web application projects in Eclipse, you would see them listed here. You can add selected projects to that JBoss instance.

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 button here.

To stop it, click the red button here.
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:

```plaintext
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

```java
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

```bash
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 SUCCESSFUL” message at the end, as shown here:

```bash
... [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).

![Figure 1-11. Creating a dynamic web project]

Enter the information shown in Figure 1-12.

![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.

![Creating the “Hello world!” page](image)

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

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

```xml
<?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.
This "servlet" is the JSF engine. You can give it any name you'd like.

http://localhost:8080/Hello/faces/???

You will access the application using a URL like this. This way, JBoss will send the request to the JSF engine for handling.

The Project Name
Hello
WebContent
...

Figure 1-18. web.xml
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

```xml
    xsi:schemaLocation="http://java.sun.com/xml/ns/javae
    http://java.sun.com/xml/ns/javae/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.

**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.

![Diagram](http://example.com/diagram.png)

**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.
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.
This is the JSF HTML namespace. This namespace contains tags like `<outputText>`:

This tag will create a UI Output component.

The page object is called the “component view root.”

Page Object (View Root)

UI Output

Value: John

Such a hierarchical data structure is called the “JSF component tree” or the “JSF view.”

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.
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.
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

```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.
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.