CHAPTER 41

JSF and Visual Web Development

Objectives

- To explain what JSF is (§41.1).
- To create a JSF page using NetBeans (§41.2).
- To use JSF UI components (e.g., Static Text, Text Field, Button, Drop Down List, List Box, Radio Button Group, Check Box Group, Text Area, Table) (§41.3).
- To use JSF containers Grid Panel, Group Panel, and Layout Panel to group components (§41.4).
- To bind data with JSF UI components (§41.5).
- To maintain persistency using session tracking (§41.6).
- To validate input using Message components (§41.7).
- To improve efficiency using virtual forms (§41.8).



41–2 Chapter 41 JSF and Visual Web Development

41.1 Introduction

The use of servlets, introduced in Chapter 39, is the foundation of the Java Web technology. It is a primitive way to write server-side applications. JSP provides a scripting capability and allows you to embed Java code in HTML. It is easier to develop Web programs using JSP than servlets. However, JSP has some problems. First, it can be very messy, because it mixes Java code with HTML. Second, using JSP to develop user interface is tedious. JavaServer Faces (JSF) comes to rescue. JSF supports visual Web development. You can create a Web user interface using a tool without writing any code. JSF completely separates Web UI from Java code, so the application developed using JSF is easy to debug and maintain.

A couple of tools support JSF. This chapter demonstrates visual JSF using NetBeans 6.7. Visual JSF is a plug-in for NetBeans 6.7. To install it, choose **Tools**, **Pluggins**, and select **Visual JSF**.

41.2 Getting Started with Visual JSF

A simple example will illustrate the basics of developing visual JSF projects using NetBeans. The example is a server-side application that displays the date and time on the server, as shown in Figure 41.1.

🖉 htt	🕼 http://localhost:8080/jsfdemo/ - Windows Internet Explorer							
G	9.	http://localhost:8080/jsfdemo/		Google	P -			
\$	¢	Http://localhost:8080/jsfdemo/		📙 🔄 💌 🐜 🖛 🚽 <u>P</u> age 🕶 🍥 T	ools ▼ "			
Cu	rrent	time on the server is Wed Sep 05 21	:09:	33 EDT 2007	•			

FIGURE 41.1 The application displays the date and time on the server.

Here are the steps to create the application.

- Choose File > New Project to display the New Project dialog box. In this box, choose Web in the Categories pane and Web Application in the Projects pane. Click Next to display the New Web Application dialog box.
- 2. In the New Web Application dialog box, enter and select the following fields, as shown in Figure 41.2(a):

Project Name: jsfdemo Project Location: c:\book Check Set as Main Project

Click *Next* to display the dialog box for choosing servers and settings. Select the following fields as shown in Figure 41.2(b). (Note: Some examples do not work in V3 without modifications of project files due to bugs in the new version. So, this chapter uses V2.)

Server: GlassFish V2 Java EE Version: Java EE 5

Click *Next* to display the dialog box for choosing frameworks, as shown in Figure 41.3. Check *Visual Web JavaServer Faces*. Click *Finish* to create the project, as shown in Figure 41.4.

create a project

servlets

JSP

JSF

NetBeans Visual Web Tool

41.2 Getting Started with JSF 41-3

New Web Application				
Steps	Name and Location	New Web Application		
Choose Project Name and Location Server and Settings Frameworks	Project Location: C:'book Browse Project Eoder: C:'book/jsfdemo	Steps Server and Settings 1. Choose Project Add to Enterprise Application: www.downews.org 2. Name and Location Add to Enterprise Application: www.downews.org 3. Server and Settings Server: GlassFish V3 4. Frameworks Server: GlassFish V3		
	Use Dedicated Folder for Storing Lbraries Ubraries Folder: Different users and projects can share the same compliation libraries (see Help for details). Set as Main Project	☐ Use dedicated library folder for server JAR files Java EE Version: Java EE 5 → Context Bath: jjsfdemo		
	< <u>Back</u> Next > Einish Cancel Help	< <u>Back</u> Next > Einish Cancel Help		
<u>.</u>	(a)	(b)		



New Web Application	×				
Steps	Frameworks				
1. Choose Project 2. Name and Location	Select the frameworks you want to use in your web application.				
4. Frameworks	Spring Web MVC 2.5				
	JavaServer Faces Struts 1.3.8				
- A	Hibernate 3.2.5				
< <u>Back</u> Next > <u>Fi</u> nish Cancel <u>H</u> elp					

FIGURE 41.3 Checking Visual Web JavaServer Faces creates a visual Web project.

A visual Web project is created with an empty page named Page1.jsp, as shown in Figure 41.4. You see **Page1** with subtabs in the content pane. The **Design** tab enables you to design the UI visually. The **JSP** tab enables you to view and modify the JSP codes. The **Java** tab enables you to view and modify the Java code. These three tabs are synchronized. Whenever you make a change in the Design, the corresponding code in the JSP and Java will change and vice versa.

The Palette window contains the UI components that you can drag and drop to the I Design pane.

Tip There are several windows for a visual Web project. If a window (e.g., the Palette window) is not displayed, choose an appropriate menu from the Windows menu to display it.

Now you are ready to create a page for displaying the current date and time on the server. Here are the steps:

 Rename Page1.jsp to CurrentTime.jsp. To do so, right-click the context menu on Page1.jsp in the Project pane and choose **Refactor** > **Rename** to display the Rename dialog box. renaming files Enter CurrentTime as the new name and click *Refactor* to finish renaming.

Design tab JSP tab Java tab

palette window

41–4 Chapter 41 JSF and Visual Web Development

NetBeans IDE 6.7									<u>x</u>
<u>File Edit View Navigate Source Refactor Run De</u>	bug <u>P</u> rofil	e Tea <u>r</u>	<u>m T</u> ools <u>I</u>	<u>W</u> indow <u>H</u>	elp				_
12 12 12 15 C	~	Ĩ	B D	> 🗊 • (D •		Q- Search	(Ctrl+I)	
Projects 4 % ; Files ; Services	Page 1	*						Palette	₽ %
🖃 🕀 jsfdemo	Design	JSP	Java	67 8 :	Any Size	•		<u>W</u> oodstock Ba	sic 🔺
🖨 🐻 Web Pages								🖓 Label	E
I ⊕… D WEB-INF								A Static Text	
⊕… 🕦 resources								Text Field	
Page 1.jsp		1.1	1.1				1 1 1		
⊕ Lonfiguration Files								Text Area	
Server Resources								🕑 Button	
E Source Packages			Design you	ır page by dr	agging component	s from the Palette.		📾 Hyperlink	
		_	~					🛛 😹 Image Hyperlink	c
		De De	erine a comp	onent's ben wielst sliel th	avior by adding co	de: Lither double-clip	IK	Drop Down List	
Project Woodstock Themes		uieu	omponent or	nync-click u	ie component and	choose its event hai	iluler,		Ψ.
Component Libraries			Page layo	ut is currentl	v set to Grid Lavou	ut, which positions		: No Properties	₽ %
Data Source References			component	s at specific :	and y coordinate:	s. If you would like			
		1	to use flow l	ayout, which	positions compone	ents left to right and	I	<no properties<="" th=""><th>~ </th></no>	~
: Navigator 🖉 🕷	: : :	to	op to botton	n, change the	e Page Layout prop	perty to Flow Layout	s 1 1	<no propercies<="" th=""><th></th></no>	
At 18 A - 1 - 1 - 1		••••••			• . • . • . • . • . • .	•••••••••••••••••••••••••••••••••••••••			0
<no avaiiadie="" view=""></no>									
	<u> </u>								
								£	INS

FIGURE 41.4 An empty page is created in a new Visual Web project.

Static Text

2. Click to select the *Static Text* icon from the Palette pane and drop it to the Design pane, as shown in Figure 41.5. In the id field in the Properties pane for the static text, enter **stCurrentTime** to change the id. Whenever you click a component in the Design pane, its corresponding properties are displayed in the Properties pane. You can view and modify the component's properties in the Properties pane.

jsfdemo - NetBeans IDE 6.7						x			
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help									
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Q- Search (Ctrl+	:)							
Projects 🕼 🕺 ; Files 🕴 ; Services	🛃 CurrentTime * 🛛 🗱			Palette		₽ %			
🖃 🚇 jsfdemo	Design JSP Ja	iva 🗑 🕲 📮 Any Size	•	Woodstock Ba	ISÎC	•			
🖨 🚯 Web Pages				🔊 Label					
🖶 🖟 🕠 WEB-INF	·¤	· _ · · · · · · · • · · · · · · · · · ·	<u> </u>	A Static Text					
i resources	□Static Text	5.01.0		I Text Field					
CurrentTime.jsp		Edit Java Source		Tout tour					
🕀 👦 Configuration Files		Bind to Data		E lext Area					
Server Resources		Add		🕑 Button					
Source Packages		Property Bindings		📾 Hyperlink		-			
		Add Binding Attribute		: atCurrentTime	Proportion	n v			
Test Libraries		Edit ISD Source		General	Properties				
Project Woodstock Themes				id General	stCurrentTime	n E			
Component Libraries		Edit Text			Stearchine	-			
Data Source References				style	height: 22px;				
		Select Parent		styleClass		õ			
stCurrentTime - Navigator		Edit Event Handler		text					
EurrentTime		cuit event Handler				_			
🕀 🗗 🖾 RequestBean1		Set Initial Focus		stCurrentTime		•			
SessionBean 1				staticText1 (Static	Text)				
ApplicationBean1		Virtual Forms	1 1 1						
		Span to Grid			100:8	INS			

FIGURE 41.5 A static text is placed in the Design pane.

41.2 Getting Started with JSF 41-5

- 3. Right-click on **stCurrentTime** in the Design pane to display a content menu, as shown in Figure 41.5. Choose *Add Binding Attributes*. This step automatically generates add binding attributes ates code that enables you to access **stCurrentTime** in the Java source code.
- 4. Click the Java tab in the content pane and enter

M41_LIAN0807_08_SE_C41.QXD 11/16/09 10:18 AM Page 41-5

```
stCurrentTime.setText("Current time on the server is "
    + new java.util.Date());
```

in the **prerender** method, as shown in Figure 41.6.



FIGURE 41.6 You can modify the code in the Java tab.

5. Right-click on the **jsfdemo** project to display a context menu and choose **Run** to run the JSF. This starts a Web browser to display the current date and time, as shown in Figure 41.1.

Note

By default, the first page in the project is set as the main page. You can access it using the URL http://localhost:8080/jsfdemo. The explicit URL for this page is http://localhost:8080/jsfdemo/faces/ CurrentTime.jsp.

Note

If you choose the JSP tab, you will see the corresponding JSP file, as shown in Figure 41.7. The contents of this file in XML mirrors the UI components in the Design pane. Whenever you add, remove, or change the UI components in the Design pane, the contents in the JSP are also updated. It is possible to modify the JSP file directly, but this is not recommended for new users. Modifying the JSP file mistakenly could corrupt the entire project. You can completely ignore the JSP file when using this tool.

Note

Clicking the Java tab in the content pane, you will see the Java source file, known as the *page bean file*. This file contains several methods. Among them are the four JSF life-cycle methods **init**, **preprocess**, **prerender**, and **destroy**.

The **init** method is called whenever the page is navigated to, either by directly via a URL, or indirectly via page navigation. The **preprocess** method is called after the component tree has been restored, but before any event processing takes place. The **prerender** method is called

JSP file

explicit URL

page bean file init preprocess prerender

41–6 Chapter 41 JSF and Visual Web Development



FIGURE 41.7 You can see the code in the JSP tab.

destroy

just before rendering takes place. This method will be called only for the page that will actually be rendered. The **destroy** method is called after rendering is completed for this request.

All these methods are automatically implemented. You can add the code to customize their behaviors. Normally you should customize the **prerender** method to set appropriate properties for the components for rendering.

41.3 JSF UI Components

JSF provides many user interface components that you can use in visual design. These components are comparable to Swing components. They are JavaBeans components with similar properties and events as their counterparts in Swing. The frequently used components are grouped into the *Basic* and *Layout* nodes in the Palette window. This section introduces some standard components such as text fields, drop down lists, list boxes, radio buttons, check boxes, text areas, and buttons, through an example. The example is to design a form, as shown in Figure 41.8. Clicking the *Register* button displays the input in a separate form, as shown in Figure 41.9.

The example can be completed in three phases:

- 1. Create the initial input form.
- 2. Create the result form.
- 3. Write the code for handling the event.

Phase I: Creating the input form

 Right-click the jsfdemo node in the Project window to display a context menu, and choose New > Visual Web JSF Page to display the New Visual Web JSF Page dialog box. Enter Registrations in the File Name field. Click *Finish* to create Registrations.jsp, as shown in Figure 41.10. Right-click Registration.jsp in the Projects pane to display a context menu and choose *Set As Start Page* so this page will start when you run the jsfdemo project.

create a new page

41.3 JSF UI Components 41-7

http://localhost:8080/jsfdemo/faces/Registrations.jsp - Windows Internet Explorer
🚱 🗸 💌 http://localhost:8080/jsfdemo/faces/Registrations.jsp 🔽 🍫 🗙 Google
$ \begin{array}{c} & & \\ & \\ & \\ & \\ & \\ & \\ & \\ & \\ & \\ $
Student Registration Form
Last Name: Smith MI: T First Name: John
Gender: @ Male C Female
Major: Computer Science Minor: Computer Science Mathematics
Hobby: 🔽 Tennis 🔽 Golf 🔲 Ping Pong
Remarks:
No remarks
Register

FIGURE 41.8 The form consists of various UI components.



FIGURE 41.9 The result from the input form is displayed in this form.

2. Drop a Static Text in the Design pane with the text Student Registration Form . You can set a new font and color in the style property in the Properties window, if you wish.	set font and color
3. Drop a text field and set its id to tfLastName, column to 15, and label to Last Name:, as shown in Figure 41.11.	create a text field
4. Drop a Text Field and change its id to tfMi, column to 1, and label to MI:.	create a text field
5. Drop a Text Field and change its id to tfFirstName, column to 10, and label to First Name:.	create a text field
6. Drop a Radio Button Group. Change its id property to rbgGender , columns property to 2, and label to Gender .	radio button group
7. Right-click the radio button group object in the Design to display its context menu, choose <i>Configure Default Options</i> to display the Options Customizer dialog box. Set the appropriate display and value, as shown in Figure 41.12(a).	radio button values
8. Drop a Drop Down List and change its id to ddMajor and label to Major:.	drop down list

41–8 Chapter 41 JSF and Visual Web Development

🗊 jsfdemo - NetBeans IDE 6.7						
File Edit View Navigate Source Refactor Run De	ebu	ig Profile Team Tools Window Help				
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		- " " " >)			
Projects 4 % ; Files ; Services		Start Page 📽 🚰 CurrentTime 📽 🚰 Registrations 🕺	Palette D ≋			
□⊕ jsfdemo 🖌	• I	Design JSP Java 🙀 🥸 🖽 Any Size 🗸	🗆 Woodsto 🔺			
🖕 🕞 Web Pages			🔊 Label 📃			
ten 🗊 Web-INF			A Static Text			
🕀 🖟 🗊 resources	HE		To the second			
CurrentTime.jsp			L Text Field			
Registrations.jsp			🚟 Text Area			
🕀 🔈 Configuration Files		Design company has described and have been blacked by the Deletter and the	🕑 Button			
Server Resources			📾 Hyperlink			
Source Packages		Define a component's behavior by adding code: Either double-click	Mage Hyperi			
		the component or right-click the component and choose its event handler.	Tinge Hyperi			
			Registra D 🕷			
Project Woodstock Themes	-	Page layout is currently set to Grid Layout, which positions	General 🔺			
Component Libraries	_	components at specific x and y coordinates. If you would like	id 🛄 🗉			
	丰	to use flow layout, which positions components left to right and	Appearance			
Registrations - Navigator	8	top to bottom, change the Page Layout property to Flow Layout.	Background			
			Dealerson C			
			Registrations 🕜			
Filters:			Registrations (request)			
<u>.</u>			INS INS			

FIGURE 41.10 A new JSF page is created.

🗊 jsfdemo - NetBeans IDE 6.7							
<u>File Edit View Navigate Source Refactor Run Debug Profile Team Tools W</u> indow <u>H</u> elp							
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	→ 🔐 🏷 🐘 • 🕞 • 🔍 • Q• Search (Ctrl+I)						
Projects @ % ; Files ; Services	ime 🖗 Registrations * 🕺 🚺 🐨 🖬 : Palette	≥ ≈					
□ 🕀 jsfdemo 🔺	Design JSP Java 🙀 🥸 📮 Any Size 🗆 Woodstock Basic	-					
🖨 🖟 Web Pages							
i web-INF	A Static Text						
tesources ⊑		-					
		ib 92					
En Configuration Files	ast Name						
Server Resources							
🕀 🔓 Source Packages	columns 15)					
🕀 🚡 Test Packages 👻	label Last Name:	1					
textField1 - Navigator] -					
	label Label (String) It set, a label is rendered	0					
thA staticText1:Student Registri	adjacent to the component with the value of this attribute as the label text.						
	E	INS					







41.3 JSF UI Components 41-9

list values

check box group

- 9. Right-click the list in the Design to display its context menu, and choose *Configure* list values *Default Options* to display the Options Customizer dialog box. Set the appropriate display and value, as shown in Figure 41.12(b).
- 10. Drop a List Box and change its id to lbxMinor, label to Minor:, rows to 2, and list box multiple to true.
- 11. Right-click the box in the Design to display its context menu, and choose *Configure Default Options* to display the Options Customizer dialog box. Set display and value the same as shown in Figure 41.12(b).
- 12. Drop a Check Box Group. Change its id to chkgHobby, label to Hobby:, and columns to 3. Right-click the box in the Design to display its context menu, choose *Configure Default Options* to display the Options Customizer dialog box. Set the display name and values for Tennis, Golf, and Ping Pong.
- 13. Drop a Static Text with the text **Remarks**:.
- 14. Drop a Text Area with id and columns set to taRemarks and 70.
- 15. Drop a Button and set its **id** to **btRegister** and its **text** to **Register**.
- 16. Add binding attribute for tfLastName, tfMi, tfFirstName, rbgGender, ddMajor, lbxMinor, chkgHobby, and taRemarks. To add binding attribute for a component, right-click the component in the Design pane to display the context menu and choose Add Binding Attribute.

The user interface for the input is now created as shown in Figure 41.13. All these components are contained in **form1**. Now create **form2** for displaying the data collected from the first form.

create a text area create a button

add binding attribute

jsfdemo - NetBeans IDE 6.7									
<u>File Edit View N</u> avigate <u>S</u> ource Ref <u>a</u> ctor <u>R</u> un <u>D</u> ebug <u>P</u> rofile Tea <u>m</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp									
n n 2 5 0 0 .	- T T	Q- Search (Ctrl+I))						
Pro 4 % ; Files ; Services	Start Page * P Current Ime * P Registrations * *	: Palette	x.						
jstdemo A	Design JSP Java 🚱 🖽 Any Size 👻	A Page Alert	^						
Web Pages									
	Student Registration Form	Property Sheet							
		Property Sheet Se	ection						
Registrations.isp		Property							
⊕ S Configuration Files	Last Name:	Form							
Server Resources		Woodstock Com	posite						
🕀 🚡 Source Packages	Gender: Male Female	Alert	- -						
🕀 🕞 Test Packages 🔻		: Registrations isn	Properties D &						
: Registrations.jsp - Navigator 🛛 🕫 🕷		- Properties							
	Major: Mathematics	Name	Registrations						
	Computer Science	Extension	jsp						
	Hobby: Tennis Golf Pina Pona	All Files	C:\book\jsfdem						
		File Size	937						
		Modification Time	Jul 11, 2009 9:09:5						
	Remarks:	⊡Text							
		Encoding	UTF-8						
		Registrations.isp	A						
		JSF JSP File	Ŭ						
Filters: 🔲 🐼 🕲									
	L		E Wrong INS						

FIGURE 41.13 form1 is created to contain input components.

Phase II: Creating the result form

- 1. Uncheck the **visible** property for **form1** to set it false so **form1** is not displayed in the Design pane.
- 2. Drop a Form from the *Layout* tab in the Palette pane to create a form named **form2**, as create another form shown in Figure 41.14.

41–10 Chapter 41 JSF and Visual Web Development

🕥 jsfdemo - NetBeans IDE 6.7		
<u>File Edit View Navigate Source Refactor R</u>	n <u>D</u> ebug <u>P</u> rofile Tea <u>m</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp	
1a 1	- F B > B. D -	Q- Search (Ctrl+I)
Pro @ % ; Files ; Services	Start Page 🗱 🗿 CurrentTime 🗱 🗿 Registrations * 🕺] : Palette
⊡ ⊕ jsfdemo ▲	Design JSP Java 🐻 🕲 🖽 Any Size 🗸	- Page Haginent box
🖨 🔥 Web Pages		- A Page Alert
🕸 🖓 WEB-INF		Property Sheet
🗊 🛺 resources 📰		Property Sheet Section
CurrentTime.jsp		Dreporty
Registrations.jsp		Property
🕀 🔥 Configuration Files		Form
Server Resources		Woodstock Composite
Berlin Gource Packages Source Pack		🖄 Alert
🕀 🚡 Test Packages 🗸 🗸	······································	
		; form2 - Properties
: torm2 - Navigator W %		General
Registrations		id form2 =
e pagel	*****	- Appearance
		style position: abs
+ ead1		styleClass
⊡ ≝≫ body1		Behavior
torm1		autoComplete
		taroet
		form2
♥ ddMajorDefaultOptions		form2 (Form)
IbxMinorDefaultOptions		
		E Wrong INS

FIGURE 41.14 form1 is invisible. form2 is created.

3. Drop eight Static Texts into **form2** and set their **id** to **stLastName**, **stMi**, **stFirstName**, **stGender**, **stMajor**, **stMinor**, **stHobby**, and **stRemark**, respectively, as shown in Figure 41.15. Add binding attributes for all these static texts.

jsfdemo - NetBeans IDE 6.0.1 Elle Edit View Navigate Source Refactor Build	<u>y</u> un <u>P</u> rofile Versjoning <u>T</u> ools <u>W</u> indow <u>H</u> el	_ _ 					
12 12 12 ≫							
J Projects ◀0 × J Files J Services	Registrations * ×	I Palette X					
□ □ ···· ··· ··· ··· ··· ··· ··· ··· ··	Design JSP Java 🐼 🕑	Any Size					
B C WEB-INF B C resources		A Static Text					
CurrentTime.jsp	Static Text	Text Field					
form2 - Navigator 40 ×	Static Text	I Text Area ■ Text Area ■ Button					
● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	Static Text	🖙 Hyperlink					
aA. stLastName	Static Text	☐ J form2 - Prop ▶ ×					
B. A stFirstName	Static Text	General id f					
⊕…A stGender ⊕…A stMajor	Static Text	form2					
terme A stMinor	- <u>-</u>	form2 (Form)					
A stRemark	<u> </u>	· · · · · · · · · · · · · · · · · · ·					

FIGURE 41.15 Static texts are added to form2.

- 4. Set the **rendered** property in **form2** to **false**. **form2** will not be displayed when the page is started.
- 5. Now check the **visible** property of **form1** to **true** and the **visible** property of **form2 false**.
- 6. Add binding attributes for **form1** and **form2**. In the Navigator pane, right-click **form1** to display the context menu and choose *Add Binding Attribute*. Similarly you can add binding attribute for **form2**.

41.4 JSF UI Containers 41-11

Phase III: Handling the event

We are ready to implement the code to handle the event for the *Register* button. Double-click the *Register* button to generate the handler method **btRegister_action** in the page bean event handler file. Implement the methods as follows:

```
1 public String btRegister_action() {
     // TODO: Replace with your code
 3
     form1.setRendered(false);
                                                                              form1 not rendered
 4
     form2.setRendered(true);
                                                                              form2 rendered
 5
     form2.setVisible(true);
     stLastName.setText("Last name is " + tfLastName.getText());
 6
                                                                              get text
     stMi.setText("MI is " + tfMi.getText());
 7
     stFirstName.setText("First name is " + tfFirstName.getText());
 8
 9
10
     stGender.setText("Selected gender is " +
       rbgGender.getSelected().toString());
11
                                                                              selected radio button
     stMajor.setText("Selected major is " +
12
       ddMajor.getSelected().toString());
13
                                                                              selected item
14
15
     String[] selectedMinors = (String[])(lbxMinor.getSelected());
                                                                              selected items
16
     String minors = "";
     for (int i = 0; i < selectedMinors.length; i++)</pre>
17
18
       minors += selectedMinors[i];
19
     stMinor.setText("Selected minors are " + minors);
20
21
     String[] selectedHobbies = (String[])(chkgHobby.getSelected());
     String hobbies = "";
22
23
     for (int i = 0; i < selectedHobbies.length; i++)</pre>
24
       hobbies += selectedHobbies[i];
25
     stHobby.setText("Selected hobbies are " + hobbies);
26
27
     stRemark.setText("Remarks are " + taRemarks.getText());
28
     return null;
29 }
```

Clicking the *Register* button displays **form2**. **form1**'s **rendered** property is set to **false** (line 3) in order to hide it. **form2**'s **rendered** property is set **true** to make it visible (line 4). **form2**'s **visible** property is set **true** to ensure that the form itself is visible (line 5). For a component to be displayed, both **visible** and **rendered** properties must be **true**.

All the JSF UI components are JavaBeans components. Their properties have the associated get and set methods. You can obtain the text in a text field and text area using the **getText()** method (lines 6–8). You can obtain the selected items from the a radio button group, check box group, drop down list, or list box using the **getSelected()** method (lines 11, 13, 15, 21).

Note

The second form is displayed when the *Register* button is clicked. When **form2** is displayed, **form1** will not be shown. This is achieved by setting **form1**'s **rendered** property to **false**. Note in the Design pane, you are free to move **form2** to a desired location even if it overlaps **form1**.

41.4 JSF UI Containers

Like Swing, JSF provides containers that can be used to group components to achieve a desired layout. Three containers are available under the Layout node in the Palette: *GridPane1*, *GroupPane1*, and *LayoutPane1*. GridPane1 is like the Swing GridLayout. GroupPane1 is similar to the Swing FlowLayout. LayoutPane1 organizes child components using flow or absolute positioning.

Consider the following example that computes loan payments. The example lets the user enter a loan amount, number of years, and annual interest rate, as shown in Figure 41.16.

getText()

getSelected()

GridPanel GroupPanel LayoutPanel

41–12 Chapter 41 JSF and Visual Web Development

Clicking the *Compute Loan* button displays the monthly payment and total payment, as shown in Figure 41.17.

Ø١	lozilla	Firefo	¢				_ 🗆 🗙
<u>F</u> ile	<u>E</u> dit	View	History	<u>B</u> ookmarks	Tools	<u>H</u> elp	0 ⁴ 0 0 ₀ 0
B	http://k	ocalhos	t:8080/j	sfdemo/faces/@	Comput	eLoan.jsp	-
							
L	oan Am	ount:	10	000			
N	lumber o	fYears:	5				
4	Annual In	terest Ra	te: 5.5				
	Co	mpute Lo	an				-
Don	e						

FIGURE 41.16 The form lets the user enter loan information.

🕲 Mozilla Firefox		_ 🗆 ×				
<u>F</u> ile <u>E</u> dit <u>V</u> iew H	i <u>s</u> tory <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp	0				
http://localhost:8	3080/jsfdemo/faces/ComputeLoan.jsp	• 🔊				
Loan Amount:	10000					
Number of Years:	5					
Annual Interest Rate:	5.5					
						
Compute Loan						
Monthly Payment	Monthly Payment 191.01162171781937					
Total Payment	11460.697303069162					
Done						

FIGURE 41.17 The loan payment is computed and displayed.

Here are the steps to create the UI:

- Right-click the jsfdemo node in the Project window to display a context menu, choose New > Visual Web JSF Page to display the New Page dialog box. Enter ComputeLoan in the File Name field. Click *Finish* to create ComputeLoan.jsp. Set ComputeLoan.jsp as the start page for the project.
- 2. Drop a GridPanel to the Design pane and set its id to gridPanel1 and columns to 2.
- 3. Drop a Static Text, a Text Field, a Static Text, a Text Field, a Static Text, and a Text Field to the grid panel in this order. Set the static text to Loan Amount, Number of Years, and Annual Interest Rate. Change the text fields id to tfLoanAmount, tfNumberOfYears, and tfAnnualInterestRate. Add binding attributes for the text fields.
- 4. Drop a Button to the grid panel and set its id to btComputeLoan and text to *Compute Loan*.
- 5. Drop another **GridPanel** to the Design pane below the preceding grid panel and set its id to gridPanel2 and columns to 2.
- 6. Drop a Static Text, a Text Field, a Static Text, and a Text Field to gridPanel2 in this order. Set the static text to Monthly Payment and Total Payment. Change the id of the text fields to tfMonthlyPayment and tfTotalPayment. Add binding attributes for tfMonthlyPayment, tfTotalPayment, and gridPanel2.

The user interface is created as shown in Figure 41.18.

create a grid panel

create a new page

create a grid panel

41.4 JSF UI Containers 41–13

jsfdemo - NetBeans IDE 6.7		- x
File Edit View Navigate Source Refactor R	un Debug Profile Team Tools Window Help	
1 1 1 1 1 4 5 7 7 1	→ 📅 📸 🕨 🏧 • 🗇 • 🔍 Q• Search (Ctrl+I)	
Pro 4 % Files Services	🗗 ComputeLoan * 📽 🐼 Loan.java 📽 🕢 🖛 💌 🗖 🗄 Palette	₽ %
⊡…⊕ jsfdemo ▲	Design JSP Java 🙀 🥸 🖽 Any Size 🗆 Woodstock Basic	*
🖶 🖟 Web Pages		
🕀 🖓 WEB-INF 📰	A Static Text	
⊕ ··· ↓) resources	Loan Amount:	
ComputeLoan.jsp	Number of Years	
		-
🕀 🐻 Configuration Files	Annual Interest Rate:	ıt: D ≋
Server Resources	General	
staticText5:Total Payment: - Navig 18	id staticTex	ct5 🛄 🗌
, otatior oxisi rotari agni ona inarigini o to	Appearance	
	style	
	Monthly Payment:	
	Total Payment:	•
	staticText5 (StaticText)	
Filters:		
	2	206:1 INS

FIGURE 41.18 The panels are created to group UI components.

```
🍯 Tip
```

If the order of the components in a panel is not correct, you can reorder them in the Navigator reorder components pane using the mouse.

Double-click the *Compute Loan* button in the Design pane to generate the method **btComputeLoan_action()**. Implement it as follows:

```
1 public String btComputeLoan_action() {
     double loanAmount =
 2
                                                                              get loanAmount
 3
       Double.parseDouble(tfLoanAmount.getText().toString().trim())
 4
     int numberOfYears =
                                                                              get numberOfYears
       Integer.parseInt(tfNumberOfYears.getText().toString().trim());
 5
 6
     double annualInterestRate = Double.parseDouble(
                                                                              get annualInterestRate
 7
       tfAnnualInterestRate.getText().toString().trim());
 8
9
     chapter41.Loan loan = new chapter41.Loan(
                                                                              create a loan
10
       annualInterestRate, numberOfYears, loanAmount);
11
     gridPanel2.setRendered(true);
12
                                                                              gridPanel2 rendered true
     tfMonthlyPayment.setText(loan.getMonthlyPayment() + "");
13
                                                                              display monthly payment
14
     tfTotalPayment.setText(loan.getTotalPayment() + "");
                                                                              display total payment
15
16
     return null;
17 }
```

The handler obtains **loanAmount**, **numberOfYears**, and **annualInterestRate** from the input text fields (lines 2–7). The program creates a **Loan** object (lines 9–10). You need to create a **Loan** class for this method to work. To create the **Loan** class, choose **File** > **New** > **Java Class** to display the New Java Class dialog box. Enter **Loan** in the Class Name field and **chapter41** in the package field. The **Loan** class was given in Listing 10.2.

Set gridPanel2's rendered property to false. Run the program using the URL http://localhost:8080/jsfdemo/faces/ComputeLoan.jsp. You will see the user interface, as shown in

create Loan class

41-14 Chapter 41 JSF and Visual Web Development

run ComputeLoan.jsp

Figure 41.16. When you click the *Compute Loan* button, **gridPanel2**'s **rendered** property to **true** (line 12). You will see the monthly payment and total payment displayed in Figure 41.17.

41.5 Binding Data with UI Components

data-aware components

You can bind data from a SQL query to a JSF UI component. This is a powerful feature and makes database programming easy. These JSF UI components are known as *data-aware components*.

Consider the following example that lets the user choose a course, as shown in Figure 41.19. After a course is selected in the drop down list, the students enrolled in the course are displayed in the table, as shown in Figure 41.20. In this example, all the course titles in the **Course** table are bound to the drop down list and the query result for the students enrolled in the course is bound to the table.



FIGURE 41.19 You need to choose a course and display the students enrolled in the course.

http://local	host:8080/jsfo http://localhos	demo/1 t:8080/	faces/Displays jsfdemo/faces/I	Student.jsp;js Dis 🔻 🍫 🗙	essionid = Google	=dc3089c19	93b 💶 🗖
r 🏠 🗷 h	ttp://localhost:8	3080/jsf	demo/f	☆ - ₪	-	<u>:</u> } <u>P</u> age ▼	Tools V
Choose a course: Intro to Java II							
Table		_			_	_	
Table ssn t ₊	firstName ≁,	mi ₁.	lastName ≁,	phone 🛧	Birth Date 🛧	deptid ቱ	
Table ssn ↑↓ 444111111	firstName ↑↓ John	mi ₁↓ K	lastName ↑↓ Stevenson	phone ↑↓ 9129219434	Birth Date 🔩	deptid 1. BIOL	
Table ssn ♣ 444111111 444111118	firstName ↑↓ John Toni	mi ₁₊ K R	lastName № Stevenson Peterson	phone 14 9129219434 9129229434	Birth Date ↑↓ Apr 29, 1964	deptid ↑↓ BIOL MATH	-

FIGURE 41.20 The table displays the students enrolled in the course.

create database connection

To develop this example, first you have to create a new database connection. In the Services pane, right-click the Database node and choose *New Connection* in the context menu (Figure 41.21(a) to display the New Database Connection dialog box, as shown in Figure 41.21(b)).

Use the same MySQL database we have been using since Chapter 37. Enter the database information, as shown in Figure 41.21(b). Click *OK* to create the database connection. You will see the database under the Databases node in the Services tab, as shown in Figure 41.22.

41.5 Binding Data with UI Components 41–15



FIGURE 41.21 You can create a database connection from the Services pane.

File Édit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Files Files Files Files Files Services Services WiSQL Command 1 m WiSQL Server at localhost: 3306 [agsuser] (disconnect WiSQL Server at localhost: 3306 [agsuser] (disconnect WiSQL Server at localhost: 1527/mample [app on APP] Set Set * from javabook.course Set Set * from javabook.course Set Set * from javabook.course Set * * * * * * * * * * * * * * * * * * *	NetBeans IDE	6.7								x
Projects Files Projects <td><u>File</u> <u>Edit</u> <u>V</u>iew</td> <td><u>N</u>avigate <u>S</u>ource Ref<u>a</u>ct</td> <td>or <u>R</u>un <u>D</u>ebug <u>P</u>rofil</td> <td>e</td> <td>Tea<u>m</u></td> <td>ools <u>W</u>i</td> <td>ndow <u>H</u>elp</td> <td></td> <td></td> <td></td>	<u>File</u> <u>Edit</u> <u>V</u> iew	<u>N</u> avigate <u>S</u> ource Ref <u>a</u> ct	or <u>R</u> un <u>D</u> ebug <u>P</u> rofil	e	Tea <u>m</u>	ools <u>W</u> i	ndow <u>H</u> elp			
Projects Files Services Image: Services Image: Projects Files Services Image: Projects Image: Projects Image: Projects Image: Projects <td>te te ce</td> <td>5 7 C</td> <td>T</td> <td>•</td> <td>1</td> <td></td> <td>B•O•</td> <td>- Q- Search</td> <td>(Ctrl+I)</td> <td></td>	te te ce	5 7 C	T	•	1		B •O•	- Q- Search	(Ctrl+I)	
Databases MySQL Server at localhost:3306 [agsuser] (dsconnet) MySQL Server at localhost:3306 [agsuser] (dsconnet) MySQL Server at localhost:3306 [agsuser] (dsconnet) Divers	Projects	Files	Services 🖉 🕷	1	📄 sql c	ommand 1	. %		4 >	
Image: MySQL Server at localhost: 3306 [agsuser] (disconnection of the second of th	Database	s	<u>^</u>		6 루	9 19 1	K) 🗟 - 🔯	- 5.5	98899	⊒ £2 (
Java D8 Joint Viers Joint Viers Joint Carbox: 1527/sample [app on APP] Joint State State Administration Joint State Application Joint State Applicat Joint State App	🗎 🕀 📓 MySC	L Server at localhost:3306	agsuser] (disconnecte	lf	Bel	ect *	from java	book cours	2	
Intersection Intersection Intersectin Intersection	Java	DB								-+
<pre>ide::defby:://ocalhost:1527/wrevel [Travel on TRAVE] ide::defby:://ocalhost:1527/wrevel [Travel on TRAVE] ide::defby::de</pre>	⊕ Static	derby://localbost:1527/sam	ble [app on APP]							
<pre>idbc:derby://localhost:1527/vir [vir on VIR] idbc:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/cpplivelab2e?autoReconnect=tr idfight:mysql://localhost/pplivelab2e?autoReconnect=tr idfight:mysql://localhost?autoReconnect=tr idfight:mysql://l</pre>	i ⊕ 🕄 jdbc:	derby://localhost:1527/trav	el [travel on TRAVEL]							
Jøbc:mysql://localhost/cpplivelab2e?autoReconnect=tr jobc:mysql://localhost/javabook (scott on Default sche jinzabook Tables account address course	idbc:	derby://localhost:1527/vir [v	vir on VIR]							
Importantion schema Importantion sch	iller 🖳 jdbc:	mysql://localhost/cpplivelab2	e?autoReconnect=tri							
<pre> information_schema information</pre>	jdbc:	mysql://localhost/javabook [scott on Default schei							
i javabook i accunt address colege course course <td< td=""><td></td><td>formation_schema</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td<>		formation_schema								
		avabook J. Tables								
account account <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td<>										
college course course csi1301 csi1302 csi4990 csi4990 department <td></td> <td>address</td> <td>-</td> <td>lf</td> <td>select * f</td> <td>om iavab</td> <td>ook.co %</td> <td></td> <td></td> <td></td>		address	-	lf	select * f	om iavab	ook.co %			
Course Course Course Course Course Course Course Course CourseId SubjectId CourseId SubjectId CourseId SubjectId CourseId SubjectId CourseId SubjectId CourseId Subject CourseId Subject CourseId Subject Subject CourseId Subject Subject CourseId Subject Subject CourseId Subject		college	-							
<pre># courseId subjectid courseNumber title csci301 csci4990 csci</pre>		course			4	×		< > >	Page Size: 20	I Toti
<pre>csdi302 csdi990 csdi90 csdi9</pre>	. e	csci1301			#	courseId	subjectId	courseNumber	title	R
2 11112 CSCI 1302 Intro to Java II 4 11113 CSCI 4720 Database Systems 6 errolment 11113 CSCI 4720 Database Systems 9 faculty 11114 CSCI 4720 Database Systems 9 faculty 11115 MATH 2750 Calculus I 6 11116 MATH 3750 Calculus I 6 11116 MATH 3750 Calculus I 7 11117 EDUC 1111 Redding 8 11118 TEC 1344 Database Administration 8 11118 TEC 1344 Database Administration 9 subject Intervention Intervention 9 Forcedures isfdemo (run) SQL Command 1 execution * Intervention 9 Intil SUCCESSFUL (total time: 0 seconds) Intervention Intervention Intervention		csci1302			1	11111	CSCI	1301	Intro to Java I	^
3 11113 CSCI 4720 Database Systems 4 11114 CSCI 4750 Rapid Java Application 5 11115 MATH 2750 Calculus I 6 11116 MATH 3750 Calculus II 7 11117 EDUC 1111Reading 8 11118 ITEC 1344 Database Administration 9 student 9 Procedures 9 Procedures 9 Wews 9 Procedures 9 Will SUCCESSFUL (total time: 0 seconds)		CSCI4990			2	11112	CSCI	1302	Intro to Java II	
<pre>4 iiii4 CSCI 4750 Rajd Java Application 5 iiii5 MATH 2750 Calculus I 6 iii16 MATH 3750 Calculus I 7 iii17 EDUC 1111 Reading 8 iii18 TEC 1344 Database Administration 8 iii18 TEC 1344 Database Administration 9 subject 9 subject 9 Student 9 Stu</pre>					3	11113	CSCI	4720	Database Systems	
S a 1113 MATH 2750 Calculus I quiz quiz quiz staff statecapital student student subject wews wews wews weys		- faculty			4	11114	CSCI	4750	Rapid Java Applicatio	n
quiz quiz 1110 Print 0		person			5	11115	MATH	2/50	Calculus I Calculus II	_
Image: Staff 8 11118 ITEC 1344 Database Administration Image: Statecapital Image: Statecapital Image: Statecapital Image: Statecapital Image: Statecapital Image: Statecapital Image: Statecapital Image: State	G.	quiz			7	11117	EDUC	1111	Reading	_
Image: State capital Image: State capital capital Image: State capital ca	e de la centra de	- staff			8	11118	ITEC	1344	Database Administra	tion
Image: student Image: student Image: student Image: studen		statecapital								-
Image: subject Image: subjec		student				•	m			•
GlassFish V3 * X SQL Command 1 execution * X GlassFish V3 * X SQL Command 1 execution * X jsfdemo (run) X Java DB Database Process * X run: BUILD SUCCESSFUL (total time: 0 seconds)		men subject			Output					9 %
Image: State		Views		lfi		GlassFieb	V3* %	SOL Com	and 1 execution *	
Image: Second	III	Procedures		$\ $		jsfdemo (r	run) %	Java DB D	atabase Process *	88
III		iysql	-		an run					-
1:1 INS	< <u> </u>	I	Þ		VI BUI	D SUCC	ESSFUL (tot	cal time: 0	seconds)	-
									E 1:	1 INS

FIGURE 41.22 You can access the database from NetBeans.

Create a new Visual Web JSF page named **DisplayStudent** and create the user interface by dropping a Drop Down List, and a Table to the Design pane, as shown in Figure 41.23. Name the drop down list and table as **ddCourse** and **tbStudent**. Set the **label** property for **ddCourse** to **Choose a course:**.

create new page

create UI

41–16 Chapter 41 JSF and Visual Web Development

Image: State State State State Image: State State State Image: State State State Image: State State Image: State State Image: State					
IPro IFiles J Services Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Service state Image: Se	DisplayStudent * × Design JSP Java	ණ ලා Any Size		J Palette ▶ × Image: Checkbox Group	
WEB-INF resources ComputeLoan.jsp CurrentTime.jsp DisplayStudent.jsp	Choose a Course: Item 1 🔽			 ⊙ Radio Button ⇒ Radio Button Grou; ⇒ Image ⇒ Table 	
	column1 14	column2 *+	column3 t+]tbStudent - Proper I ×	
	row1_column1	row1_column2	row1_column3	⊡General ▲	
	row2_column1	row2_column2	row2_column3	id tbSt 🛄 🖵	
	row3_column1	row3_column2	row3_column3	Appearance	
Him heads	row4_column1	row4_column2	row4_column3	di Ciudanta 🛛 🔘	
i i i i i i i i i i i i i i i i i i i		rows_column2	rows_coumn3	table1 (Table)	

FIGURE 41.23 The user interface is created in the Design pane.



FIGURE 41.24 A data provider is created.

create data provider bind data

The drop down list should be bound to the course titles from the **Course** table. Click the **Course** table in the database connection under the Database node in the Services pane, as shown in Figure 41.22. Drop the **Course** table to the table component in the Design pane. You will see **courseDataProvider** created in the Navigator pane, as shown in Figure 41.24.

To bind the data for **ddCourse**, right-click on **ddCourse** to display a context menu in the Design and choose *Bind to Data* to display the Bind to Data dialog box, as shown in Figure 41.25. Select **course.courseId** in the Value field and **course.title** in the Display field. Click *OK* to close the dialog box. Add binding attribute for **ddCourse**.

41.5 Binding Data with UI Components 41–17

The table should be bound to a query result for the students enrolled in a selected course. Click the **Student** table in the database connection under the Database node in the Services pane. Drop the table to the table component **tbStudent** in the Design pane. You will see **studentDataProvider** created in the Navigator pane, as shown in Figure 41.26.

Each data provider has a **cachedRowSet** property that specifies a **RowSet**. **studentDataProvider** uses **studentRowSet**, as shown in Figure 41.27. Double-click **studentRowSet** under the **SessionBean1** node in the Navigator pane to display the row set in the content pane, as shown in Figure 41.28. create data provider

display query visually

Bind to Data - ddCourse	×
Current Items property setting	
#{DisplayStudent.courseDataProvider.options[course.courseId,course.	title']}
Bind to Data Provider Bind to an Object	
Choose a Data Provider to bind to ddCourse:	
courseDataProvider (DisplayStudent)	
Value field:	Digplay field:
<none></none>	<use field="" value=""></use>
course.courseId String	course.courseId String
course.subjectId String	course.subjectId String
course.courseNumber Integer	course.courseNumber Integer
course.title String	course.title String
course.numOfCredits Integer	course.numOfCredits Integer
ОК	Cancel Apply Help

FIGURE 41.25 You can set value and display name in a drop down box.

jsfdemo - NetBeans IDE 6.7						
File Edit View Navigate Source Refa	actor <u>R</u> un <u>D</u> ebug <u>P</u> re	ofile Tea <u>m</u> <u>T</u> ools <u>W</u> in	dow <u>H</u> elp			
		- 7 7 >	B·O·		Q- Search (C	Ctrl+I)
: Projects : Files : 4 %	📄 SQL Command 1 🛛 🕸	🚽 DisplayStudent * 🛚 🕷]			: Palette 🗈 🛚
faculty A	Design JSP Jav	/a 🕢 🖓 🖓 Any S	lize	•		Checkbox Group
ti in the second secon					· · · · · · · · · · · · · · · · · · ·	Radio Button
terrendaria tattatatatatatatatatatatatatatatatata	Choose a Course:	abc				Radio Button Group
terent subject	Table					Table
taunhthy		5				Table Column
studentDataProvider - Na 4		tirstname 4	mi +	lastname '+	pnone +	: studentDataP IP 8
	abc	abc	abc	abc	abc	id st
	abc	abc	abc	abc	abc	⊡Data ≡
tbStuder	··· abc	abc	abc	abc	abc	cachedRows▼
courseDataProvider		· · · · · · · · · · · · · · · · · · ·	· · · · · ·	·····		dataListener
terrestBean1						
GessionBean1 ApplicationBean1 ▼						studentDataProvi W hedRowSetDataProvider1
	•	m			4	chedRowSetDataProvider)
						1:1 INS

FIGURE 41.26 The studentDataProvider is created.

Modify the query as follows:

select all Student.ssn, Student.firstName, Student.mi, modify query

41–18 Chapter 41 JSF and Visual Web Development

NetBeans IDE 6.7						
<u>File Edit View Navigate Source Refactor Run</u>	<u>D</u> ebug <u>P</u> rofile Tea	<u>m T</u> ools <u>W</u> indow <u>H</u> el	Þ			
n 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	- F	18 D B- (1) -	(🔍 – Search (C	(trl+I)
Projects Files Se 4 %	SQL Command 1 🕺	DisplayStudent * 🕷	jdbc:mysql://k	ocalhost/javabook [s		Palette
taculty ▲ De	sign JSP Jav	a 🕢 🕲 🗗 Any Si	ze	•		Checkbox Group
					· · · · ·	Dadia Ruttan
					••••••	
terministan	Choose a Course:	abc				Radio Button Group
statecapital						🐏 Image
subject	Table					🛄 Table
taughtby						The second secon
The Views	ssn ⁺₊	firstName 🔩	mi ⁺∔	lastName ↑↓	phon	studentRowS ₪ 🕷
						General
studentRowSet:"SELECT ALL stud 🐠 🕷 🚺	abc	abc	abc	abc	abc E	id st
🖃 🗊 DisplayStudent	abc	abc	abc	abc	abc	command SE
page1						concurrency C
	abc	abc	abc	abc	abc	dataSourcen Javab
studentDataProvider						
🗄 🖷 🚳 RequestBean1						naneSize 0
SessionBean1						password
courseRowSet: SELECT * FROM cours	: : : : :				1 1	
StudentRowSet: SELECT ALL student.						studentRowSet:" 🛞
					• • • • •	cachedRowSetXImpl2
×		III			+	(CachedRowSetXImpl)
						Đ

FIGURE 41.27 You can define a SQL statement in studentRowSet.

			Q	Search (Ctrl+I)
Projects : Files : Se Projects : Files : Se Projects : Projects : Projects Projects : Projects : Projects	SQL Command 1 **	DisplayStudent * 😰 🕑 jdbc:mys [Table] me	oqi://localhost/javabook [scott on De	fault schema) 🕫 🕧 💌 🗖
Views Navigator	♦ If is stName If is stName	ent.ssn, ent.firstName, ent.lastName, ent.lastName, ent.phone,		
<no available="" view=""></no>	Stud Stud Stud	ent.birthDate, ent.street, ent.zipCode,		

FIGURE 41.28 The query is visually displayed in the content pane.

Student.lastName, Student.phone, Student.birthDate, Student.street, Student.zipCode,

41.5 Binding Data with UI Components 41–19

Student.deptId from Student, Enrollment, Course
where course.courseId = ?
 and student.ssn = enrollment.ssn
 and enrollment.courseId = course.courseId

10:18 AM Page 41-19

M41_LIAN0807_08_SE_C41.QXD 11/16/09

Note that the ? mark is a placeholder for the prepared statement. Close and reopen prepared statement **studentRowSet**. You will see the new data diagram, as shown in Figure 41.29. If you don't see it, go back to the Design pane and refresh the project, then reopen **studentRowSet**.

NetBeans IDE 6.7				
<u>File Edit View Navigate Source Refacto</u>	or <u>R</u> un <u>D</u> ebug <u>P</u> rofile Tea <u>m</u>	<u>T</u> ools <u>W</u> indow <u>H</u> elp		
17 11 12 4 19 (* [- fr f			Q. ► Search (Ctrl+I)
; Projects ; Files ; Se @	3 🕱 📑 SQL Command 1 🕱 🚰	DisplayStudent * 🛚 🗐 jdbr	c:mysql://localhost/javabook [so	cott on Default schema] 🗱 🚺 🗖 🗖
e→faculty e→person e→guiz e→staff e→statecapital e→student e→subject e→faculty	Student [Table Sen Sen FirstName I astName		nroliment [Table] ssn courseId dateRegistered grade	Course [Table]
: - Navigator 🛛 🖓	il 🏁 Column	Alias Table	Output	Sort Type Sort Order Crit
	ssn	student		^
	firstName	student		
		student	V	
<no available="" view=""></no>	ent.ssn, student.	`firstName`, student	t.mi, student.`last	Name`, student phone, studer

FIGURE 41.29 The tables and their relationship in the SQL query are visually displayed.

You can customize the table's layout by right-clicking the table in the Design pane and choosing *Table Layout* to display the Table Layout dialog box, as shown in Figure 41.30. Choose the selected columns and set the header for birthDate to Birth Date, as shown in Figure 41.30. Click *OK* to close the dialog box.

Drop a Message Group component to the upper right corner of the page. This component is useful to display diagnostic errors.

Double-click the drop down list to generate the **ddCourse_processValueChange-**(ValueChangeEvent event) handler and implement it as follows:

```
1 public void ddCourse_processValueChange(ValueChangeEvent e) {
 2
     try {
 3
       getSessionBean1().getStudentRowSet().setObject(
                                                                             set courseId
 4
         1, ddCourse.getSelected());
 5
       studentDataProvider.refresh();
 6
     }
 7
     catch (Exception ex) {
 8
       error(ex.toString());
 9
     }
10 }
```

The **studentRowSet** object can be obtained using **setSessionBean1().getStudentRowSet()** (line 3). **studentRowSet** object is an instance of JDBC **RowSet**. The bind data Message Group

41–20 Chapter 41 JSF and Visual Web Development

🗍 Table Layout - tb	Student
Columns Options	
Get Data From: stu	vdentDataProvider (DisplayStudent)
Available	Selected
student.zipCode student.street	> student.ssn Up student.firstName Down <
Column Details	
Header Text:	Birth Date
Footer Text:	
Component Type:	Static Text
Value Expression:	#{currentRow.value["student.birthDate"]}
Width:	
Horizontal Align:	Left → Vertical Align: <pre><ru></ru></pre> (<not set=""> →</not>
🔽 Sortable	
	OK Cancel Apply Help

FIGURE 41.30 You can customize the layout of the table.

setObject method sets the parameter value (line 4). The **refresh()** method executes the query for the row set.

The **error** method displays the error message in the Message Group component (line 8). The Message Group component is not displayed if the **error** method is not called.

Replace the body of the **prerender** method with the following code to ensure that the table is properly displayed initially.

```
public void prerender() {
    try {
        if (ddCourse.getSelected() == null) {
            courseDataProvider.cursorFirst();
            getSessionBean1().getStudentRowSet().setObject(
                1, courseDataProvider.getValue("Course.courseId"));
            studentDataProvider.refresh();
        }
    }
    catch (Exception ex) {
        error(ex.toString());
    }
}
```

auto-submit

Right-click on the drop down list in the Design pane to choose *Auto-Submit on Change*. In the Properties window, the following code appears in the **onchange** property:

webui.suntheme.common.timeoutSubmitForm(this.form, 'ddCourse');

Now when the user changes a value in the drop down list, the Web browser automatically submits the change.

Run the program using the URL http://localhost:8080/jsfdemo/faces/DisplayStudent.jsp. You will see the user interface, as shown in Figure 41.19.

override prerender

41.6 Session Tracking

Chapter 40, "JSP," introduced session tracking using JavaBeans by sharing the JavaBeans objects among different pages. You can specify the JavaBeans objects at the application scope, session scope, page scope, or request scope. JSF supports session tracking using JavaBeans at the application scope, session scope, and request scope.

Consider the following example that prompts the user to guess a number. When the page starts, the program randomly generates a number between 0 and 99. This number is stored in the session. When the user enters a guess, the program checks the guess with the random number in the session and tells the user whether the guess is too high, too low, or just right, as shown in Figure 41.31.

🕲 Mozilla Firefox	_ 🗆 🗙
<u>F</u> ile <u>E</u> dit <u>V</u> iew Hi <u>s</u> tory <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp	$\langle \rangle$
	.jsp 🔹 🔛
Enter your guess: 50 Guess	
Too high	<u> </u>
Done	11.
	1-1-1-1
V Mozilla Firefox	
<u>File Edit View History Bookmarks Tools H</u> elp	
	r.jsp 🔻 ⋗
Enter your guess: 25	
Too low	-
Done	//.
🕲 Mozilla Firefox	<u>_ ×</u>
<u>F</u> ile <u>E</u> dit <u>V</u> iew Hi <u>s</u> tory <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp	()
The second seco	r.jsp 💌 ⋗
Enter your guess: 40 Guess	
You got #	
Done	1

FIGURE 41.31 The user enters a guess and the program displays the result.

Create a new page named **GuessNumber** and create the user interface by dropping a text field with **id tfGuess** and **label** "Enter your guess: " and a Button with **id btGuess**, and a Static Text with **id stResponse** for displaying the program's response to the user's guess, as shown in Figure 41.32. Add binding attributes for **tfGuess** and **stResponse**.

To store the random number in the session, create a property named **number** in **SessionBean1. java** with get and set methods as follows:

create GuessNumber page

create UI

session bean property

int number;
public int getNumber() {
 return number;

}

41–22 Chapter 41 JSF and Visual Web Development

```
public void setNumber(int number) {
   this.number = number;
}
```

SessionBean1.java is in the Source Packages folder in the jsfdemo project.

Add the following highlighted code in the constructor of SessionBean1.java, as shown in Figure 41.33:

public SessionBean1() {
 number = (int)(Math.random() * 100);
}

create random number



FIGURE 41.32 The user interface is created in the Design pane.



FIGURE 41.33 You can modify the code in SessionBean1.java.

41.7 Validating Input 41–23

Double-click the Guess button in the Design to generate the handler **btGuess_action()** implement button handler and implement the method as follows:

```
1 public String btGuess_action() {
     int guess = Integer.parseInt(tfGuess.getText().toString().trim());
 2
                                                                               get guess
 3
 4
     if (guess < getSessionBean1().getNumber())</pre>
 5
       stResponse.setText("Too low");
                                                                               too low
 6
     else if (guess > getSessionBean1().getNumber())
 7
       stResponse.setText("Too high");
                                                                               too high
 8
     else
9
       stResponse.setText("You got it!");
10
11
     return null;
12 }
```

A session bean object is automatically created when a session starts. The **getSession- getSessionBean1() Bean1()** method (lines 4, 6) returns the session bean object.

41.7 Validating Input

M41_LIAN0807_08_SE_C41.QXD 11/16/09

In the preceding **GuessNumber** page, an error would occur if you did not enter a guess or you entered a string rather than an integer before clicking the *Guess* button. A simple way to fix the problem is to check the text field before processing any event. JSF provides several convenient and powerful ways for input validation. You can perform validation to ensure that the required field has been filled and the input is an integer for the preceding example.

10:18 AM Page 41-23

Consider the following example that displays a form for collecting user input as shown in Figure 41.34. All text fields in the form must be filled. If not, error messages are displayed. The SSN must be formatted corrected. If not, an error is displayed. If all input is correct, clicking *Submit* displays the result in a static text, as shown in Figure 41.35.

🕲 Mozilk	a Firefox	<
<u>F</u> ile <u>E</u> dit	t <u>V</u> iew Hi <u>s</u> tory <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp	
<	• C ValidateForm.jsp •	
Name; SSN: Age: Height:	form1:tfName: Validation Error: Value is required. Enter a valid SSN, e.g., 123-44-5678 form1:tfAge: Validation Error: Value is required. form1:tfHeight: Validation Error: Value is required.	
Subr	mt	•

FIGURE 41.34 The error messages are displayed if input is invalid.

Create a new page named **ValidateForm** and create the user interface as follows:

1. Drop a GridPanel to the Design and set its **columns** to **2**. Add four Static Texts, four Text Fields, and a Button to the grid panel, as shown in Figure 41.36. Set the **id**

create ValidateForm page create UI create a GridPanel

41–24 Chapter 41 JSF and Visual Web Development

property of the text fields to **tfName**, **tfSSN**, **tfAge**, and **tfHeight**, respectively. Set the button's **id** to **btSubmit** and **label** to Submit.

escape property

- 2. Drop a Static Text below the grid panel and set its **id** to **stResult**. Set its **escape** property to **false** so you can display HTML contents in the static text.
- add binding attribute
- 3. Add binding attributes for tfName, tfSSN, tfAge, tfHeight, and stResult.

🕑 Mozilla	Firefo	ĸ					×
<u>F</u> ile <u>E</u> dit	View	History	<u>B</u> ooki	marks	Tools	Help 3	0
<	- @		a [http:	://localh	ost:8080/jsfdemo/faces/ValidateForm.jsp 💌 🛛	>
Name; SSN:	John Sm 333-34-5	ith 5678					
Age:	41						
Height: Subn The input Name: Jo SSN: 333 Age: 41 Height: 5	5.7 nit s you ente hn Smith -34-5678 7	ered are					
Done							//.



) jsfdemo - NetBeans IDE 6.7 <u>File <u>E</u>dit <u>V</u>iew <u>N</u>avigate <u>S</u>ource Ref<u>a</u>ct</u>	or <u>R</u> un <u>D</u> ebug <u>P</u> rofile Tea <u>m</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp
17) († 12 1 2 1 5 († 1	- 🔐 🎲 🕨 🌇 - 🖓 - 🔍 Search (Ctrl+1)
Image: Pr Image: Pr <td< th=""><th>ava ValidateForm * X Image: Constraint of the system Design JSP Java Java</th></td<>	ava ValidateForm * X Image: Constraint of the system Design JSP Java Java
⊕-A staticText3:A ⊕-E ffAge ⊕-A staticText4:F ⊕-E tfHeight ⊕-A staticText4:F ⊕-E tfHeight ⊕-A staticText4:F ⊕-A staticText4:F ⊕-E tfHeight ⊕-A staticText4:F	converter escape Behavior A-min escape Escape (poolean) Escape the num text so it won't be interpreted by the browser as HTML 170:19 IN

FIGURE 41.36 The user interface is created in the Design pane.

You can use the Message component to display input errors. Drop four Message components next to each text field, as shown in Figure 41.37. To associate a Message component with a text field, choose an appropriate value in the Message's **for** property, as shown in Figure 41.37. Message will automatically display an error message when input validation fails in the associated text field.

display error in Message **for** property

M41_LIAN0807_08_SE_C41.QXD	11/16/09	10:18 AM	Page 41-25	
			$-\Phi$	-

jsfdemo - NetBeans IDE 6.7		
File Edit View Navigate Source Refactor R	In Debug Profile Team Tools Window Help	
¹¹ 11 12 13 19 (°	T T T T I I I I I I I I I I I I I I	Q, ▼ Search (Ctrl+I)
Proj 4 % ; Files ; Services	ber 🙆 SessionBean 1. java 🕱 🔂 ValidateForm * 🕱 🛛 🚺 🗖	Palette
E - D resources	Design JSP Java 🚮 🚱 🗇 Any Size 🗸	Calendar
ComputeLoan.jsp		Ele Upload
CurrentTime.jsp	Name: Message summary for the ame	Tree
GuessNumber.jsp	SSN: Message summary for trSSN ;	🗆 Tree Node
Registrations.jsp	Age: Message summary for tfAge	· · · · · · · · · · · · · · · · · · ·
ValidateForm.jsp	Height: Message summary for tfHeight	message1 - Properties
imessage1 - Navigator		style height: 24px;
in i	Button	styleClass
⊕ e btGuess:Button		for tillame v
⊕ A stResult E	Static Text	visible
⊕ ⊕ message1 ⊕ ⊕ message2		for
⊕ 🕢 message3		the component assetfName
🖾 👘 🐨 message4 🔽		message componentfSSN
		E 214:1 INS

FIGURE 41.37 Four Message components are created in the Design.

For each text field, set its **required** property **true** to indicate that an input value is **requi**red for the field.

required property

LengthValidator

You can validate the length of the input using the LengthValidator. Suppose the name must have minimum 1 character and maximum 10 characters. To set its length validator, choose new LengthValidator() in the validatorExpression property for tfName. You will see lengthValidator1 appearing in the Navigator pane, as shown in Figure 41.38. Set the minimum and maximum properties for lengthValidator1 to 1 and 10.

🔘 jsfdemo - NetBeans IDE 6.7		
<u>File E</u> dit <u>V</u> iew <u>N</u> avigate <u>S</u> ource Ref <u>a</u> ctor <u>R</u>	un <u>D</u> ebug <u>P</u> rofile Tea <u>m</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp	
1 1 1 1 1 1 1 1 1 1	- T B > B. D .	Q ► Search (Ctrl+I)
Proj 👁 📽 🕴 Files 🕴 Services	ber 🗟 SessionBean1.java 🕺 🗗 ValidateForm * 🕺 🚺 💌 🗖	: Palette 🗈 🕷
DisplayStudent.jsp	Design JSP Java 🕢 🖓 🖓 🖓 🗛	🖁 Anchor
Registrations.jsp		(i) Message
ValidateForm.jsp	Name: Message summary for tfAge	P Message Group
Configuration Files	SSN: Message summary for tfSSN	<u>W</u> oodstock Layout
🕀 🚡 Source Packages	Age: Message summary for tfAge	; lengthValidator1 - Properties D 🕷
Test Dackanes	Height: Message summary for tfHeight	General
ilengthValidator1 - Navigator		id lengthValidator 1 🛄
		maximum 10
⊕ ⊕ message2 ⊕ ⊕ message3		minimum 1
⊕ 🕢 message4	Static Taxt	
🦾 💞 lengthValidator 1		
🕀 🗠 🔐 RequestBean1		minimum 🥨
🖶 🖻 SessionBean1		minimum (int) minimum
🗄 🗠 🐼 ApplicationBean1 📼	i	
		INS INS

FIGURE 41.38 You can specify the maximum and minimum properties for a lengthValidator.

41–26 Chapter 41 JSF and Visual Web Development

regular expression	The format for SSN is ddd-dd-dddd , where d is a digit. You need to create a custom val- idator and use a regular expression to validate a SSN. Right-click tfSSN in the Design, and choose Edit Event Handler > validate to generate the method named tfSSN_validate in the Java source page. Implement the method as follows:
regular expression throw ValidatorException	<pre>1 public void tfSSN_validate(FacesContext context, 2 UIComponent component, Object value) { 3 String ssn = String.valueOf(value); 4 5 if (!ssn.matches("[\\d]{3}-[\\d]{2}-[\\d]{4}")) { 6 throw new ValidatorException (new FacesMessage(7 "Enter a valid SSN, e.g., 123-44-5678")); 8 } 9 }</pre>
ValidatorException	The program checks whether the ssn is correctly formatted. If not, a ValidatorException is thrown in line 6.
	Tip Right-click on ValidatorException in the Editor pane to display a context menu and choose <i>Fix Imports</i> to import javax.faces.validator.ValidatorException.
implement Submit button	Similarly, you can write the code to ensure that age is an integer between 1 and 135 and height is a double value between 2.0 and 10.0. The <i>Submit</i> button can be implemented as follows:
	<pre>1 public String btSubmit_action() { 2 stResult.setText("The inputs you entered are " + 3 "Name: " + tfName.getText() + " " + 4 "SSN: " + tfSSN.getText() + " " + 5 "Age: " + tfAge.getText() + " " + 6 "Height: " + tfHeight.getText() + " "); 7 8 return null; 9 } </pre>
	41.8 Virtual Forms
	When you submit a form, all input components in the form participate in the submission. However, not all these components are needed for a particular submission. For example, when you click the <i>Compute Loan</i> button in the form in Figure 41.39, only the text fields for loan amount, interest rate, and number of years need to be submitted. When you click the <i>Compute Body Mass Index</i> button, only the text fields for weight and height need to be submitted.
virtual form	To improve efficiency, you may define a virtual form to group input components with a submission component (e.g., a button or a drop down list with autosubmit enabled). The input components participate in the virtual form. When the user interacts with a submission component that submits a virtual form, the input components in the virtual form are submitted while the other input components not defined in the virtual form on the page are ignored.
create UI	VirtualFormDemo and create the user interface, as shown in Figure 41.40. The text fields are named tfLoanAmount, tfAnnualInterestRate, tfNumberOfYears, tfWeight, and tfHeight. Two buttons are named btComputeLoan and btComputeBMI. Two static

We are going to create two virtual forms: one for grouping the components for computing loan and the other for grouping the components for computing body mass index.

fields and static texts.

texts are named **stLoanResult** and **stBMIResult**. Add binding attributes for all text

41.8 Virtual Forms **41–27**

③ Mozilla Firefox	
<u>File Edit View History B</u> ookmarks <u>T</u> ools	; <u>H</u> elp
http://localhost:8080/jsfdemo/faces/Virt	ualFormDemo.jsp 🏠 🕇
Loan Amount: 100000 Annual Interest Rate: 7.15	Weight in Pounds: 145
Number of Years: 5	Height in Inches: 72
Compute Loan	E Compute Body Mass Index
Monthly payment is 1987.2045267470833 Total payment is 119232.27160482499	BMI is 19.67
Done	C McAfee SiteAdvisor -

FIGURE 41.39 The application lets you compute loan and body mass index.

jsfdemo - NetBeans IDE 6.7		
<u>File Edit View Navigate Source Refacto</u>	r <u>R</u> un <u>D</u> ebug <u>P</u> rofile Tea <u>m</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp	
1412 4 577	- T B > B O -	Q, - Search (Ctrl+I)
Pr 🕼 📽 ; Files 🕴 ; Services	ava 🗗 ValidateForm 📽 🗗 VirtualFormDemo * 📽 🖏 sun-web.xml 📽	
StBMIResult - Navigator StBonnesult StBMIResult - Navigator StBMIResult - Navigator StBonnesult	Design JSP Java Java Any Size Loan Amount:	
Compute BMI C	Static Text	style position: ab style vertication: ab style position: ab style vertication: ab st

FIGURE 41.40 The user interface is created for computing loan and body mass index.

 Ctrl-click to select tfLoanAmount, tfAnnualInterestRate, tfNumberOfYears, and btComputeLoan. Right-click one of the selected components to display a context menu and choose Configure Virtual Forms to display the Configure Virtual Form dialog box, as shown in Figure 41.41(a). You should see tfLoanAmount, tfAnnualInterestRate,

Configure Virt	ual Forms			×	Configure Virt	ual Forms			×
tfLoanAmount tfNumberOfYears btComputeLoan tfAnnualInterestRat	e				tfLoanAmount tfNumberOfYears btComputeLoan tfAnnualInterestRate	3			
These components	participate in and si	ubmit virtual forms a	s follows:		These components p	participate in and su	ubmit virtual forms a	is follows:	
Color	Name	Participate	Submit	New	Color	Name	Participate	Submit	New
				Delete		virtualForm1	Yes	No 💌 Yes No	Delete
		OK Can	cel <u>A</u> pply	Help			OK Can	icel <u>A</u> pply	Help
		(a)					(b)		



41–28 Chapter 41 JSF and Visual Web Development

tfNumberOfYears, and **btComputeLoan** in the upper left corner in the dialog box. Click the *New* button to create a virtual form named **virtualForm1**. Choose *Yes* in both the Participate and Submit columns, as shown in Figure 41.41(b).

2. Ctrl-click to select tfWeight, tfHeight, and btComputeBMI. Right-click one of the selected components to display a context menu and choose Configure Virtual Forms to display the Configure Virtual Form dialog box. You should see tfWeight, tfHeight, and btComputeBMI in the upper left corner in the dialog box. Click the New button to create a virtual form named virtualForm2. Choose Yes in both the Participate and Submit columns, as shown in Figure 41.42.



FIGURE 41.42 Two virtual forms are created.

Implement the handlers for the two buttons as follows:

```
public String btComputeLoan_action() {
 double loanAmount =
    Double.parseDouble(tfLoanAmount.getText().toString().trim());
  int numberOfYears =
    Integer.parseInt(tfNumberOfYears.getText().toString().trim());
  double annualInterestRate = Double.parseDouble(
    tfAnnualInterestRate.getText().toString().trim());
  chapter41.Loan loan = new chapter41.Loan(
    annualInterestRate, numberOfYears, loanAmount);
  stLoanResult.setText("Monthly payment is " +
    loan.getMonthlyPayment() + " Total payment is " +
    loan.getTotalPayment());
  return null;
}
public String btBMI_action() {
  final double KILOGRAMS_PER_POUND = 0.45359237;
  final double METERS_PER_INCH = 0.0254;
 double weight = Double.parseDouble(tfWeight.getText().toString());
 double height = Double.parseDouble(tfHeight.getText().toString());
 double bmi = weight * KILOGRAMS_PER_POUND /
    ((height * METERS_PER_INCH) * (height * METERS_PER_INCH));
 stBMIResult.setText("BMI is " + (int)Math.round(bmi * 100) / 100.0);
  return null;
}
```

Now you can run the program (see Figure 41.39). Clicking the *Compute Loan* button will submit the text fields for loan amount, annual interest rate, and number of years. Clicking *Compute BMI* will submit the text fields for weight and height.

Review Questions 41–29

🍯 Note

Supplement V.F, "More on JSF and Visual Web Development," provides additional examples on visual Web development, such as performing database inserts, updates, and deletes, and working with Ajax.

more supplemental examples

Key Terms

init 41-5
preprocess 41-5
prerender 41-5
destroy 41-6

data binding 41–14 palette window 41–3 UI container 41–11

CHAPTER SUMMARY

- **I.** JSF enables you to use a visual design tool to develop UI for Web applications.
- 2. A JSF page has three tabs: Design, JSP, and Java. The Design tab enables you to visually design UI. The JSP tab enables you to view and modify the JSP script for the page. The Java tab enables you to view and modify the page bean Java source code.
- **3.** JSF uses the life-cycle methods **init**, **preprocess**, **prerender**, and **destroy** to control the execution of a JSF page.
- **4.** JSF contains Swinglike UI components. You can click and drop these components to the Design pane to create UI.
- JSF contains Grid Panel, Group Panel, and Layout Panel containers for grouping components.
- You can create a database connection from the Databases node in the Services pane and bind database queries with JSF UI components.
- 7. You can perform session tracking using the **SessionBean**.
- **8.** You can validate input to ensure that the required field has been filled and the input is formatted correctly.
- 9. You can use virtual forms to group the participants of a form to improve efficiency.

REVIEW QUESTIONS

Sections 41.1-41.2

- **41.1** What is JSF?
- **41.2** How do you create a JSF project in NetBeans?
- **41.3** How do you create a JSF page in a JSF project?
- **41.4** Describe the life-cycle methods **init**, **preprocess**, **prerender**, and **destroy**.
- **41.5** Each JSF page has a .jsp file. What is in the JSP file? Can you modify it directly?
- **41.6** Each JSF page has a page bean file. What is in the page bean file? Can you modify it directly?

Section 41.3

- **41.7** How do you set the initial value and display names in a drop down list and in a list box?
- **41.8** How do you specify radio buttons and their names in a radio button group?
- **41.9** How do you specify check box buttons and their names in a check box button group?
- **41.10** How do you hide a component so that it will not be displayed?
- **41.11** What method should be used to obtain the input from a text field or a text area?

41–30 Chapter 41 JSF and Visual Web Development

- **41.12** What method should be used to obtain the selected item(s) from a drop down list or a list box?
- **41.13** Can you create two forms in one JSF page?

Section 41.4

- **41.14** What containers are available in JSF?
- **41.15** If the order of the components in a container is not correct, how do you reorder them?

Section 41.5

- **41.16** How do you create a database connection?
- **41.17** How do you create a data provider?
- **41.18** How do you modify a row set?
- **41.19** How do you create a table and bind a data provider with the table?

Section 41.6

- **41.20** How does JSP implement session tracking?
- **41.21** How do you create a JavaBeans property?
- **41.22** How do you obtain a JavaBeans property in the page bean file?

Section 41.7

- **41.23** What component can be used to automatically display the error message if invalid input is entered?
- **41.24** How do you set a Message component to associate with an input component?
- **41.25** What property for an input component should be checked to specify that an input is required?
- **41.26** How do you check the format of input using regular expressions?

PROGRAMMING EXERCISES

- **41.1*** (*Factorial table in JSF*) Rewrite Exercise 40.1 using JSF. Drop a Static Text to display the result. Set its escape property to false to display it as HTML contents.
- **41.2*** (*Multiplication table in JSP*) Rewrite Exercise 40.2 using JSF.
- **41.3*** (*Calculator*) Write a calculator to perform addition, subtraction, multiplication, and division, as shown in Figure 41.43.

http://localhost:8080/jsfdemo/faces/Exercise41_3.jsp - Windows Internet Explorer	
Google	9-
😭 🔆 @ http://ocahost:8080/jsfdemo/faces/Exer 🔤 🖓 ▼ 🔝 ▼ 🖷 ▼ 🔂 Page ▼ 🛞 Toc	ols ▼ "
Number 1: 4 Number 2: 5 Result: 9	
Add Subtract Multiply Divide	•

FIGURE 41.43 This JSF application enables you to perform addition, subtraction, multiplication, and division.

- **41.4*** (*Calculating tax in JSP*) Rewrite Exercise 40.4 using JSF.
- **41.5*** (*Multiple-question opinion poll*) Rewrite Exercise 40.13 using JSF.
- **41.6*** (*Addition quiz*) Rewrite Exercise 40.14 using JSF.
- **41.7*** (*Subtraction quiz*) Rewrite Exercise 40.15 using JSF.
- **41.8*** (*Guessing birth date*) Rewrite Exercise 40.16 using JSF.
- **41.9*** (*Guessing the capitals*) Rewrite Exercise 40.17 using JSF.