

SERVLETS AND JSP

Activity Book



NIIT

R201171400011-SABHYA

Servlets and JSP

Activity Book

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution and decompilation. Please view "Mark of Authenticity" label to establish proper licensed usage. No part of this document may be copied, reproduced, printed, distributed, modified, removed, amended in any form by any means whether electronic, mechanical, digital, optical, photographic or otherwise without prior written authorization of NIIT and its authorized licensors, if any.

Information in this document is subject to change by NIIT without notice. The names of companies, products, people, characters, and/or data mentioned herein are fictitious and are in no way intended to represent any real individual, company, entity, services, product or event, unless otherwise noted.

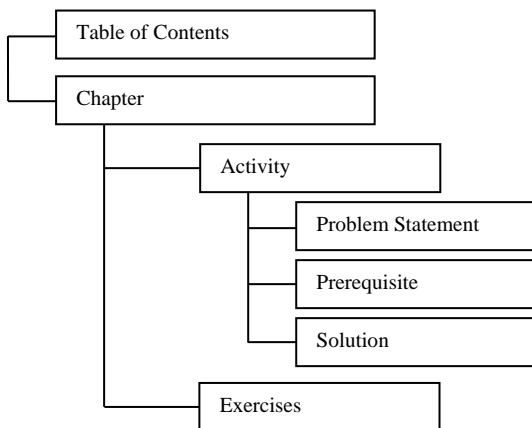
All products are registered trademarks of their respective organizations.
All software is used for educational purposes only.

Disclaimer: The documents and graphics on this courseware could include technical inaccuracies or typographical errors/translation errors. Changes are periodically added to the information herein. NIIT may make improvements and/or changes herein at any time. NIIT makes no representations about the accuracy of the information contained in the courseware and graphics in this courseware for any purpose. All documents and graphics are provided "as is". NIIT hereby disclaims all warranties and conditions with regard to this information, including all implied warranties and conditions of merchantability, fitness for any particular purpose, title and non-infringement. In any event, NIIT and/or its licensor(s)/supplier(s) shall not be liable to any party for any direct, indirect, special or other consequential damages for any use of the courseware/translated courseware, the information, or on any other hyper linked web site, including, without limitation, any lost profits, business interruption, loss of programs or other data on your information handling system or otherwise, even if NIIT is expressly advised of the possibility of such damages.

Due to the dynamic nature of the internet, the URLs and web references mentioned in this document may be (are) subject to changes, for which NIIT shall not hold any responsibility.

Servlets and JSP/AB/2018-M08-V01
Copyright ©NIIT. All rights reserved.

COURSE DESIGN - ACTIVITY BOOK



R201171400011-SABHYA

Table of Contents

Chapter 1 – Introducing Web Application Development

Introduction to Web Architecture-----	1.3
Identifying the Components of the Web Architecture -----	1.4
Understanding the HTTP Protocol -----	1.6
Introduction to Web Application Architecture and Technologies -----	1.12
Identifying the Various Web Application Architectures-----	1.13
Practice Questions -----	1.18
Summary -----	1.19

Chapter 2 – Exploring the Java Servlet Technology

Activity 2.1: Creating a Servlet -----	2.2
Problem Statement-----	2.2
Solution -----	2.2
Exercises-----	2.12
Exercise 1 -----	2.12

Chapter 3 – Exploring JavaServer Pages Technology

Activity 3.1: Creating JSP Pages -----	3.2
Problem Statement-----	3.2
Solution -----	3.2

The background of the entire page is a light blue network diagram. It consists of numerous circular nodes of varying sizes, some of which are highlighted in a slightly darker blue. These nodes are interconnected by a web of thin, light blue lines, creating a complex, interconnected pattern that suggests a network or data structure.

NIIT

R201171400011-SABHYA

Introducing Web Application Development

CHAPTER 1

This chapter discusses the various components of Web. In addition, it discusses the Web application architecture.

Objectives

In this chapter, you will learn to:

- Understand the Web architecture

R201171400011-SABHYA

Introduction to Web Architecture

The Internet has been the fastest growing technology. It has changed the way business is conducted. It is a comprehensive system of interconnected networks and is a huge storehouse of information. This is made possible by the means of the World Wide Web (WWW). The following figure depicts the WWW.



The World Wide Web

The WWW, popularly known as the Web, is a collection of several Web pages. A Web page is an electronic document containing text, images, audio, or videos. These are interlinked with each other. This interconnection among Web pages is achieved by using hypertexts. A hypertext is the highlighted or underlined text on a Web page. A hypertext connects the content on one Web page to the content on another Web page. For this reason, the hypertexts are also known as hypertext links or hyperlinks. Clicking a hyperlink opens the Web page that the hyperlink is linked to.

A set of interconnected Web pages displaying related information on a particular subject is called a website. Each website has a unique address on the Internet. This address is known as the *Uniform Resource Locator (URL)*. Websites are hosted on a Web server and are accessed by using client applications, such as a Web browser. A Web browser lets you specify the URL of a website and opens the home page of the website. The home page of a website is the first page of the website that contains links to all other pages.

The following figure shows a Web page with hyperlinks.



A Web Page with Hyperlinks

To understand how the Web functions, you need to understand its components and how they communicate with each other.

Identifying the Components of the Web Architecture

The Web architecture comprises of the following basic components:

- The client
- The Web server
- Protocol
- URL

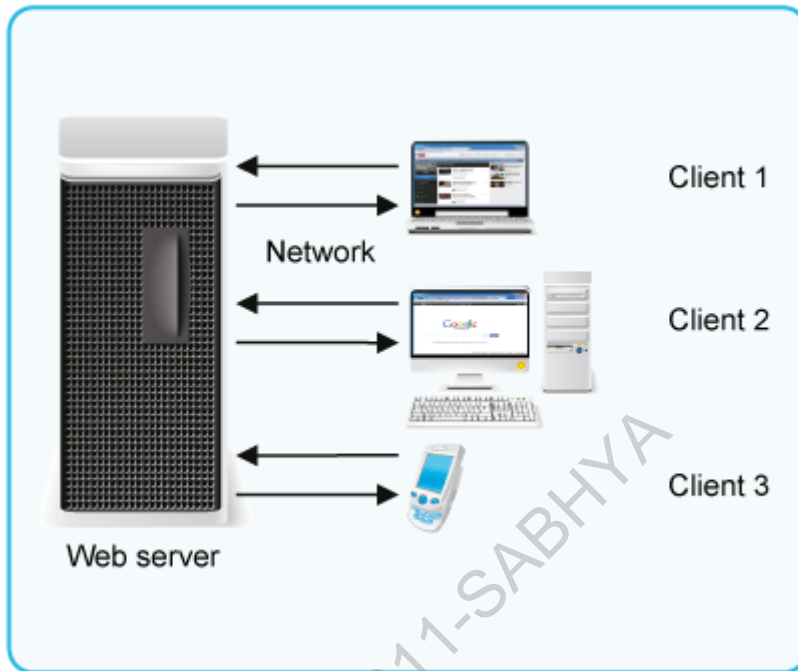
The Client

The *client* is any application that enables you to access the contents on the Web. These applications are popularly known as Web browsers. Internet Explorer, Chrome, and Mozilla Firefox are some of the popular Web browsers. By using any of these Web browsers, the client sends a request for the resources, such as a Web page, an image, an audio file, a video file, or any other document stored on the Web server.

The Web Server

A server that delivers Web pages, images, or other resources is known as a *Web server*. It is either a computer or software running on a computer that is responsible for handling the client requests. When a Web server receives a request from the client, it interprets the request and responds with the requested resource.

The following figure depicts the client-server interaction.



The Client-server Interaction

The preceding figure displays three devices; a laptop, a desktop, and a mobile phone accessing and sending a request to the Web server. The Web server processes these requests and sends the response to the respective clients. This communication between the client and the Web server is done using a protocol.

Protocol

A protocol is a set of rules that defines how computers must communicate with each other over a network. These rules define common guidelines, such as the data format and the speed of data transfer. However, there are different types of protocols, which can be classified based on their use, such as browsing the Internet, and sending and receiving e-mails. For example, the Hyper Text Transfer Protocol (HTTP) protocol enables exchange of information across the Internet.

URL

A URL is a string that refers to a Web resource. A client initiates a request for a resource by typing the URL into the Web browser. For example, if you want to access the website of a university named, GrantUniversity, you can initiate the request by typing the following URL in the Web browser:

- `http://www.GrantUniversity.com`

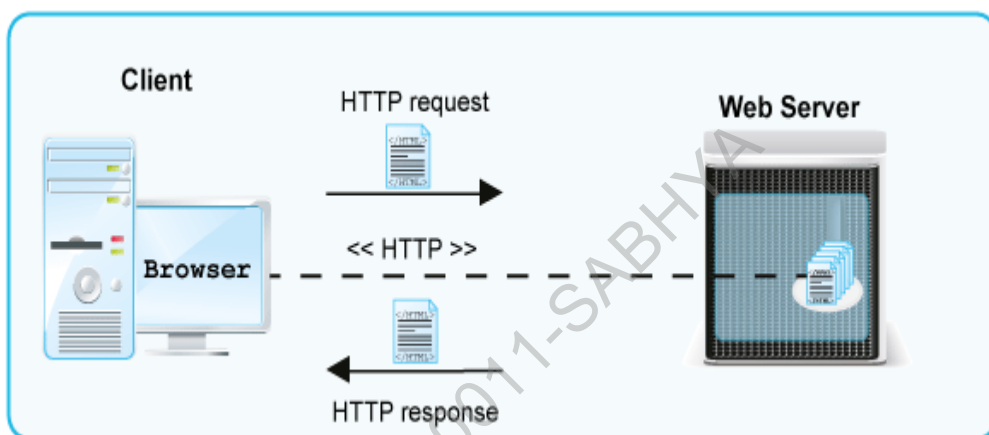
In the preceding example:

- `http` is the protocol, which enables a client and a server to communicate with each other.
- `www.GrantUniversity.com` is the part of the URL that specifies the name of the website, which is requested by the user.

Understanding the HTTP Protocol

The HTTP protocol enables communication between a client and a server over the Web. It is a stateless protocol. This means that when a client sends a request, the server processes the request and sends the response back to the client. Now, when the client sends another request, the server considers the client as a new client.

The client is primarily a Web browser that sends an HTTP request to the Web server. The Web server locates the requested resource and sends the response as an HTTP response to the client, as shown in the following figure.

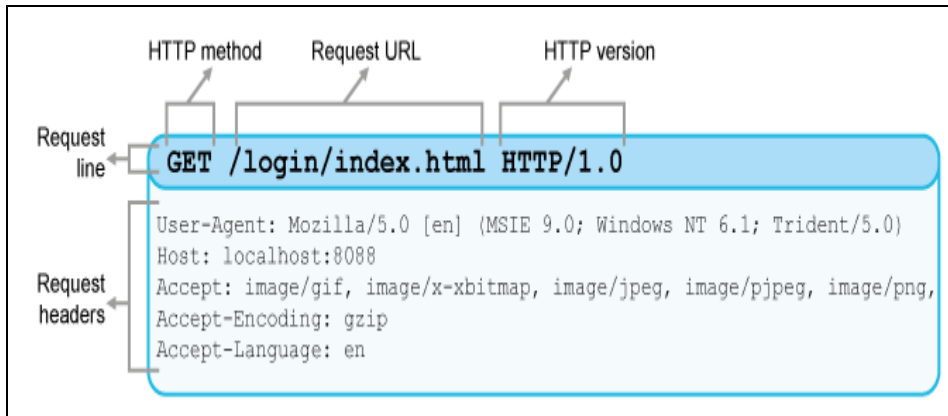


The HTTP Request and Response Cycle

The HTTP request consists of:

- **A request line:** It consists of:
 - **Request method:** It defines the method that is used by the client to request for a resource, such as GET, POST, and HEAD.
 - **Request URL:** It defines the location of the requested resource in the Web server.
 - **Version:** It defines the version of the HTTP protocol used for communication.
- **Request header:** Contains additional information about the client, such as the name and version of the Web browser.
- **Message body:** It is an optional part of the HTTP request. It contains the data entered by a user that needs to be sent to the Web server for processing. For example, the username and password information entered by a user on a login page are sent as a part of the message body to the Web server for authentication, when the submit button is clicked by the user. However, if the user requests for a resource by clicking a hyperlink or hypertext, the message body will be empty.

The following figure displays what an HTTP request consists of.



The HTTP Request Sample

Request Methods

HTTP provides various methods to request a resource stored in a Web server. The following table lists some of the request methods and their corresponding operations performed by the Web server.

<i>Request Methods</i>	<i>Operations</i>
<i>GET</i>	<i>It retrieves the information specified in the request URL. In addition, it submits the user information encoded as a part of the request URL.</i>
<i>POST</i>	<i>It sends a block of data to the Web server for processing. The data is sent as a part of the message body.</i>
<i>HEAD</i>	<i>It retrieves only the header information of the response and not the information of the message body.</i>
<i>PUT</i>	<i>It uploads the information provided in the request identified by the Request-URI on the Web server.</i>
<i>DELETE</i>	<i>It requests to delete the resource specified by the Request-URI.</i>

The Request Methods

Request Header

A request header contains additional information about the client request to the Web server. The following table describes some of the common HTTP request headers.

<i>Headers</i>	<i>Description</i>
<i>Accept</i>	<i>It specifies the Multipurpose Internet Mail Extensions (MIME) types the client receives, such as image/gif or image/jpeg.</i>
<i>Host</i>	<i>It specifies the Internet host and the port number of the requested resource.</i>
<i>User-Agent</i>	<i>It specifies the information about the client initiating the request, such as the name and version of the Web browser.</i>
<i>Accept-Language</i>	<i>It specifies the set of languages that are accepted by the client, such as en is used for English.</i>
<i>Referer</i>	<i>It specifies the URI/address of the client from which the Request-URI is received.</i>

The HTTP Request Header Fields

The HTTP response consists of:

- **Status line:** It consists of the following information:
 - **Status code and reason phrase:** The status code is a three-digit integer value and the reason phrase is a text message that describes the status code.
 - **Version:** It defines the version of the HTTP protocol.
- **Response header:** It contains additional information about the Web server and the response sent to the client, such as age and location.
- **Message body:** It consists of the response in the form of Hyper Text Markup Language (HTML) format.

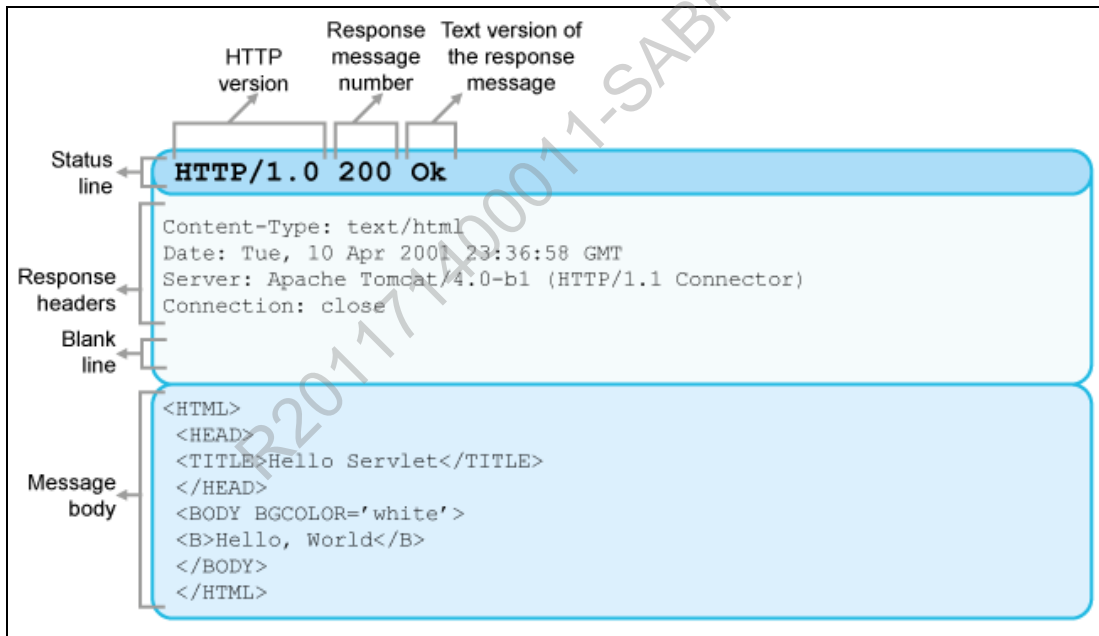
Response Headers

The following table describes some of the HTTP response headers.

<i>HTTP Response Headers</i>	<i>Description</i>
<i>Age</i>	<i>It specifies the duration in seconds since when the response is created on the Web server.</i>
<i>Retry-After</i>	<i>It specifies how long the service will be unavailable to the client. Used with a response with status code 503 (Service Unavailable).</i>
<i>Server</i>	<i>It specifies the information about the server.</i>

The HTTP Response Headers

The following figure displays what an HTTP response consists of.



The HTTP Response Sample

Status Codes

The following table lists some of the HTTP status codes and their corresponding reason phrases.

<i>Code</i>	<i>Reason Phrases</i>
200	OK
201	Created
202	Accepted
301	Moved Permanently
304	Not Modified
400	Bad request
401	Unauthorized
402	Payment required
404	Not found
500	Internal Server Error
503	Service Unavailable
505	HTTP Version Not Supported

The HTTP Status Codes

The components of the Web architecture, such as the client, URL, the HTTP protocol, and the Web server, together make the Web a powerful information sharing system. However, these days, the Web is not only used for information sharing, but also for creating and modifying information. This has led to the development of the dynamic Web pages.

A dynamic Web page is a Web page that responds to user actions or is dynamically created by a Web program on the basis of a user's request. Consider the example of a dynamic Web page of a banking website that allows you to enter your account number and credentials. After validating and authenticating the details entered by you, your account details are fetched from a database and displayed on the Web page. Such dynamic Web pages can be created by developing a Web application.



Just a minute:

Which of the following components of the Web architecture delivers Web pages, images, or other resources over the Web?

1. *Client*
2. *Server*
3. *URL*
4. *Protocol*

Answer:

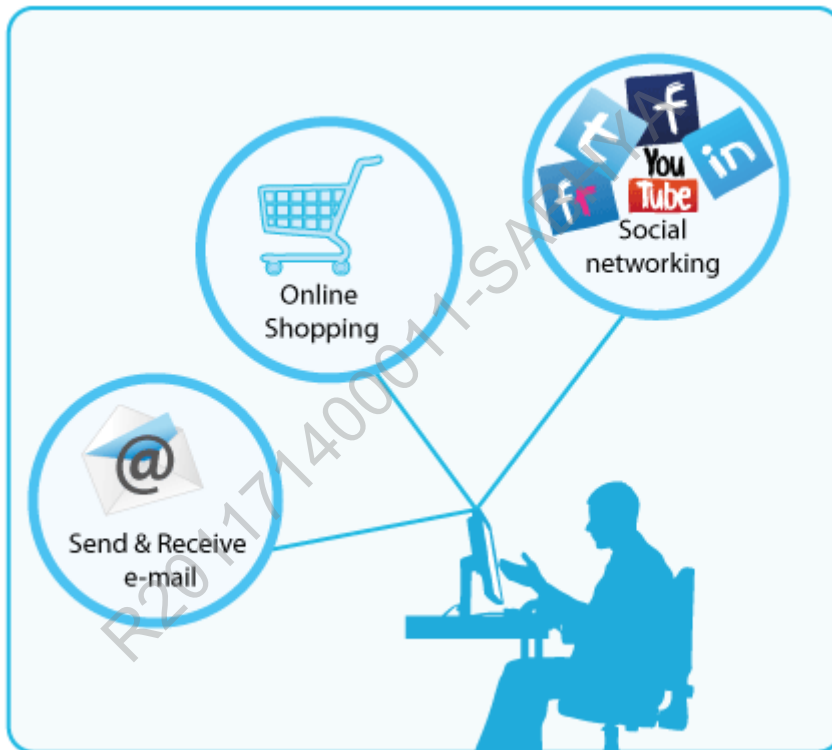
2. *Server*

R201171400011-SABHYA

Introduction to Web Application Architecture and Technologies

Web applications are programs that are executed on a server and accessed from a Web browser. These applications enable organizations to share and access information on the Internet. This information can be accessed from anywhere and at any time. In addition, Web applications can support online commercial transactions, popularly known as e-commerce. An online store, an Internet/Web mail, and social networking sites are examples of the Web applications.

The following figure showcases the example of Web applications.

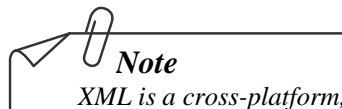


The Examples of the Web Applications

A Web application is usually broken into logical chunks known as tiers. Each tier is responsible for handling a specific role. For example, a simple Web application can be divided, based on the User Interface (UI) and business logic. However, a complex Web application can be divided based on the UI, business, and database logic.

Therefore, you can divide the Web application into the following tiers depending on the complexity of the application:

- **Presentation:** It is the top-most tier in the Web application architecture. This layer provides a UI that allows the users to interact and communicate with the applications.
- **Business logic:** It is the middle tier in the Web application architecture. It contains the logic related to the business decisions of the organization. For example, in an online banking application, the interest calculation of personal loan is one of the business decisions that can be implemented by the business logic tier. Any changes made to this tier do not affect the other two tiers.
- **Data access:** It is the last tier in the Web application architecture. It contains the logic that enables you to access the data stored in various data sources, such as Relational Database Management Systems (RDBMS), Extensible Markup Language (XML), and text files. Any changes made to this tier do not affect the business or presentation tier.



Note

XML is a cross-platform, hardware and software independent markup language. It enables computers to transfer structured data between heterogeneous systems. XML is used as a common data interchange format in a number of applications.

These logical chunks of a Web application may be contained at one place or segregated into multiple tiers, based on the complexity. In other words, a Web application follows a specific Web application architecture based on the complexity of the application.

Identifying the Various Web Application Architectures

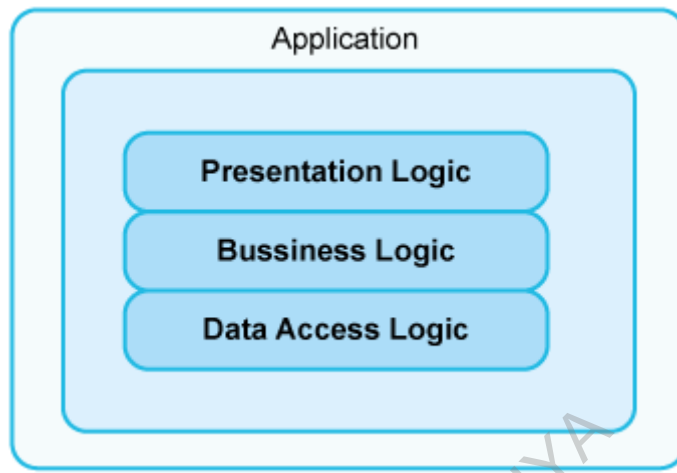
Based on the complexity of the Web application, it might follow any of the following Web application architecture:

- Single-tier architecture
- Two-tier architecture
- Three/n-tier architecture

Single-tier Architecture

The single-tier architecture is also known as the monolithic architecture. In this architecture, the presentation logic, the business logic, and the data access logic are packaged on the same computer. These layers are tightly coupled. This means that if you make changes to any one of them, you also need to modify others.

The following figure depicts the single-tier application architecture.



The Single-tier Architecture

The single-tier architecture does not support concurrent multiple users and is not flexible. Therefore, a single-tier architecture application cannot be distributed over a network. This led to the evolution of the two-tier architecture.

Two-tier Architecture

In the two-tier architecture, the presentation logic resides on the client. The business logic may reside either on the client or on the server. The data access logic resides on the server. Depending on the business requirements, the two-tier application architecture can be of the following types:

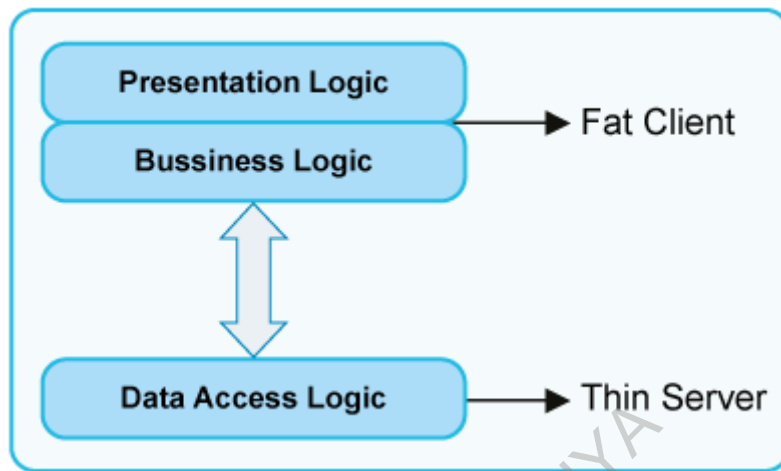
- A fat client and thin server
- A fat server and thin client

A Fat Client and Thin Server

In this type, the presentation logic and the business logic reside on the client, and the data access logic resides on the server. Such application architecture helps in reducing the load on the server because the server only stores the data and does not participate in business logic processing, which is done on the client. However, this makes the client heavy.

This type of application architecture is generally used in organizations that either cannot afford a reliable network infrastructure, or have a small user base. This is because if any modification is done either on the presentation logic or the business logic, it would require the client application to be redeployed in all the users' machine. This can be a time-consuming process if the user base is big.

The following figure depicts a fat client and thin server application architecture.



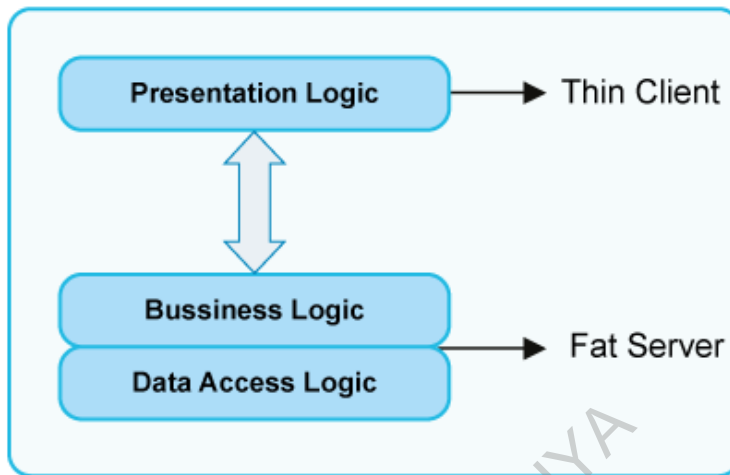
A Fat Client and Thin Server Application Architecture

A Fat Server and Thin Client

In this type, the client contains only the presentation logic. The business logic and the data access logic reside on the server. Such architecture is easily managed, less prone to security risks, and offers low maintenance and licensing costs. This is because it is easy to manage a single server instead of managing each client computer. Similarly, securing and maintaining a single computer (server) is easier than managing several computers (clients).

This type of application architecture is primarily used in organizations where the business and managerial operations that need to be performed are complex and large in number and the number of users who require access to the server is high in number. This is because in a fat server and thin client application architecture, all the business logic resides on the server. Therefore, the processing is done on the server. This saves cost as only the server needs to have a high hardware configuration. The clients do not require high configuration and can even be dumb computers. In addition, this architecture is less prone to security risks because the security is required only at the server.

The following figure displays the layout of the layers in a fat server and thin client application architecture.

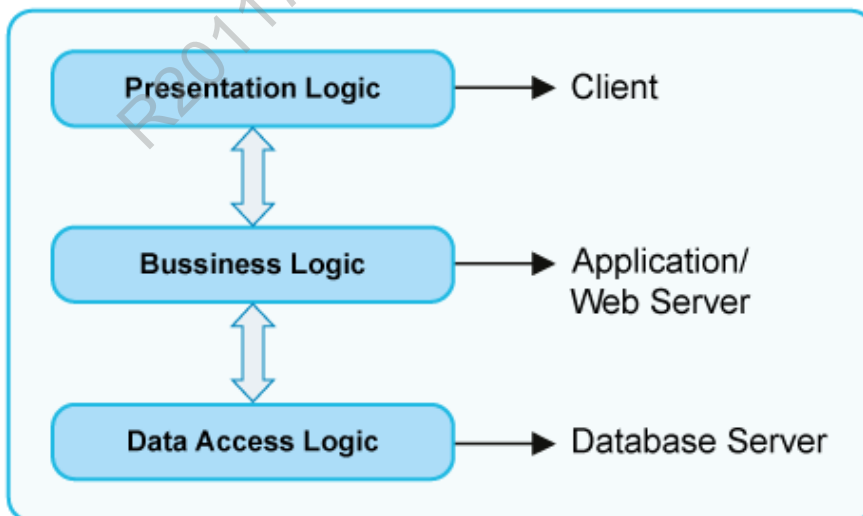


A Fat Server and Thin Client Application Architecture

However, with an increase in number of clients, the performance of the fat server and thin client application architecture may deteriorate. This is because the server may become overloaded when there is a substantial increase in the number of requests it receives. To achieve scalability and robustness, the application-related logic was further segregated in the three-tier/n-tier architecture.

Three-tier/N-tier Architecture

In the three-tier architecture, the presentation logic, business logic, and data access logic are separated into different layers, as shown in the following figure.



The Three-tier Architecture

In the three-tier architecture, every tier has its own set of processing power to service client requests. This helps in distributing the user request load resulting in high performance and scalability. If the application-related logic is segregated into more than three tiers, it is termed as n-tier architecture.

A Web application may follow any one of the preceding mentioned application architecture depending upon the requirement and complexity of the application. One or more technologies, such as HTML and Java Servlets, are used to develop these applications.

R201171400011-SABHYA

Practice Questions

1. Which one of the following components of the Web architecture is an application that enables you to access the contents on the Web?
 - a. Client
 - b. Server
 - c. URL
 - d. Protocol
2. Which one of the following HTTP request methods retrieves the information specified by the request URL in the HTTP request?
 - a. GET
 - b. POST
 - c. PUT
 - d. TRACE
3. Which one of the following status codes indicates that the client's request was successfully accepted by the server?
 - a. 404
 - b. 302
 - c. 202
 - d. 200
4. In which one of the following Web application architectures, the presentation logic and the business logic reside on the client and the data access logic resides on the server?
 - a. N-tier
 - b. Single-tier
 - c. Fat client and thin server
 - d. Fat server and thin client

Summary

In this chapter, you learned that:

- WWW, popularly known as the Web, is a collection of several Web pages.
- The Web comprises of the following basic components:
 - The client
 - The Web server
 - Protocol
 - URL
- The client is any application that enables you to access the contents on the Web.
- A server that delivers Web pages, images, or other resources is known as a Web server.
- A protocol is a set of rules that defines how computers must communicate with each other over a network.
- A URL is a string that refers to a Web resource.
- The HTTP protocol enables communication between a client and a server over the Web.
- Web applications are programs that are executed on a server and accessed from a Web browser.
- The Web application architectures are:
 - Single-tier
 - Two-tier
 - Three/n-tier

The background of the entire page is a light blue network diagram. It consists of numerous small circular nodes connected by thin, light blue lines, forming a complex web-like structure. The nodes are distributed across the page, with some clusters being denser than others. The overall effect is a technical and digital aesthetic.

NIIT

R201171400011-SABHYA

Exploring the Java Servlet Technology

CHAPTER 2



Activity 2.1: Creating a Servlet

Problem Statement

The project development team of HighTech Inc. has started the development of the Exotica Travels website. Being a part of the development team, you have been asked to create the home page of the website, as shown in the following figure.

Exotica Travels

Exotica Travels is a travel management company established by Jordan Desilva in Colombo, Sri Lanka. Today under the chairmanship of Jim Henry, the company has spread across the country. It provides online air tickets booking. In addition, it provides hotel suite booking in various exotic locations all over the world. Moreover, it provides rental car bookings. To avail the travel package services kindly log on to the website.

LOGIN AS:

☒ Existing User
☐ New User

Submit

The Home Page of the Website

Prerequisite

Ensure that the **Input** folder is copied to **C:\Chapter02\Activity1** location.

Solution

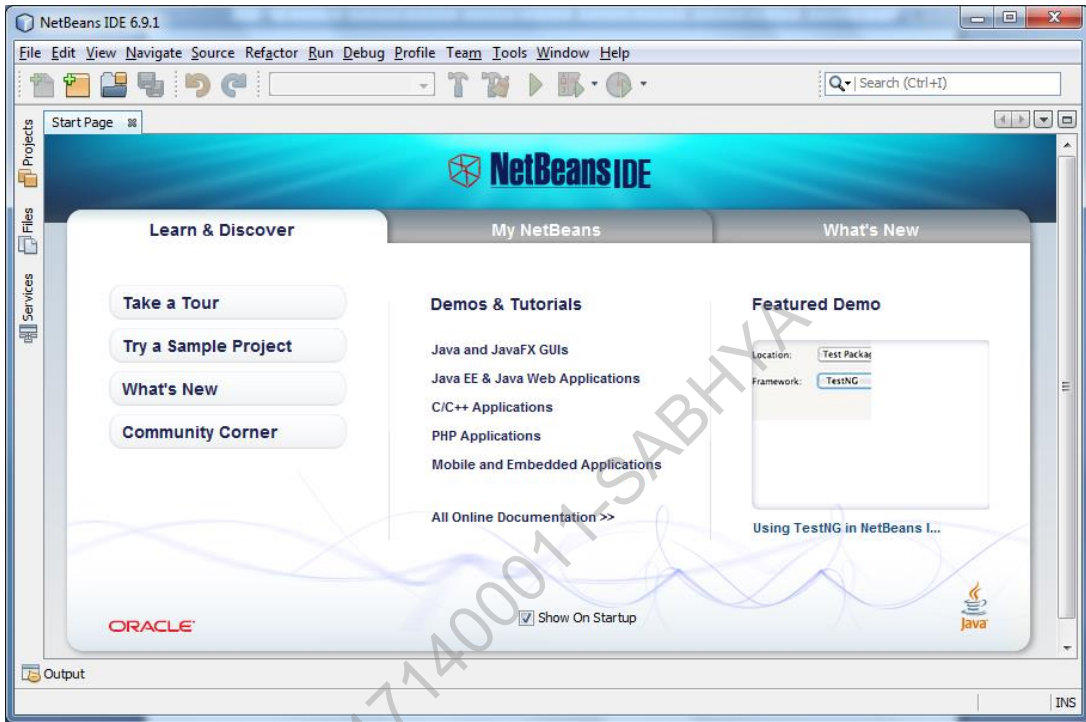
To develop the home page for the Exotica Travels website, you need to perform the following tasks:

1. Create a new project.
2. Create the **Images** folder.
3. Create the home page.
4. Verify the output.
5. Close the project.

Task 1: Creating a New Project

To create a new project, you need to perform the following steps:

1. Select **Start→All Programs→NetBeans→NetBeans IDE 6.9.1**. The **NetBeans IDE 6.9.1** window is displayed, as shown in the following figure.



The NetBeans IDE 6.9.1 Window

Note

The view of the **NetBeans IDE 6.9.1** window may differ depending upon the earlier usage of **NetBeans IDE 6.9.1**.

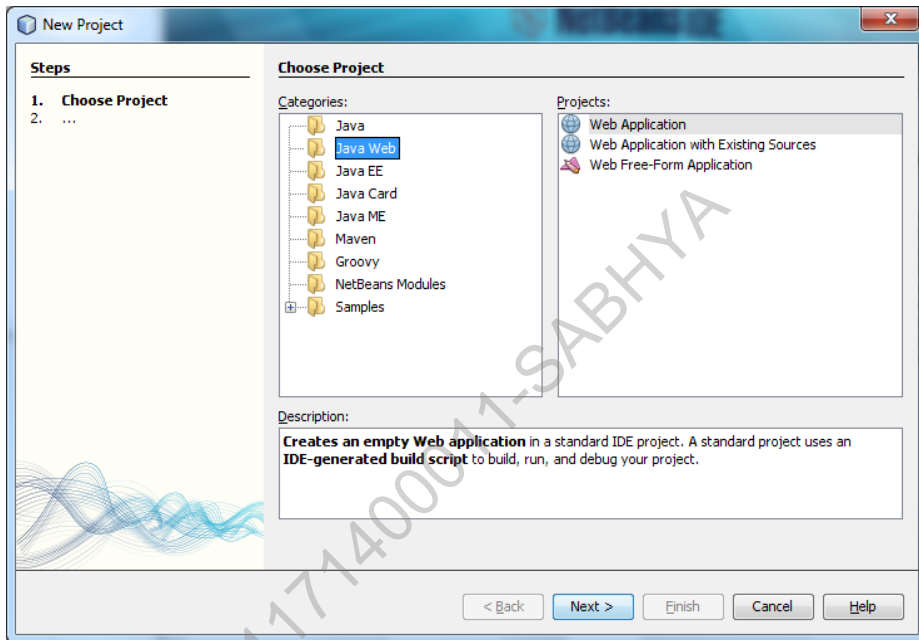
Note

The **Featured Demo** section in the **Learn & Discover** tab in the preceding figure displays the content after connecting to the Internet. If NetBeans is not able to connect to the Internet, then '**Cannot connect to internet**' message is displayed in the **Featured Demo** section.

Note

If the **Usage Statistics** dialog box is displayed after opening the NetBeans IDE, click the **No, Thank You** button.

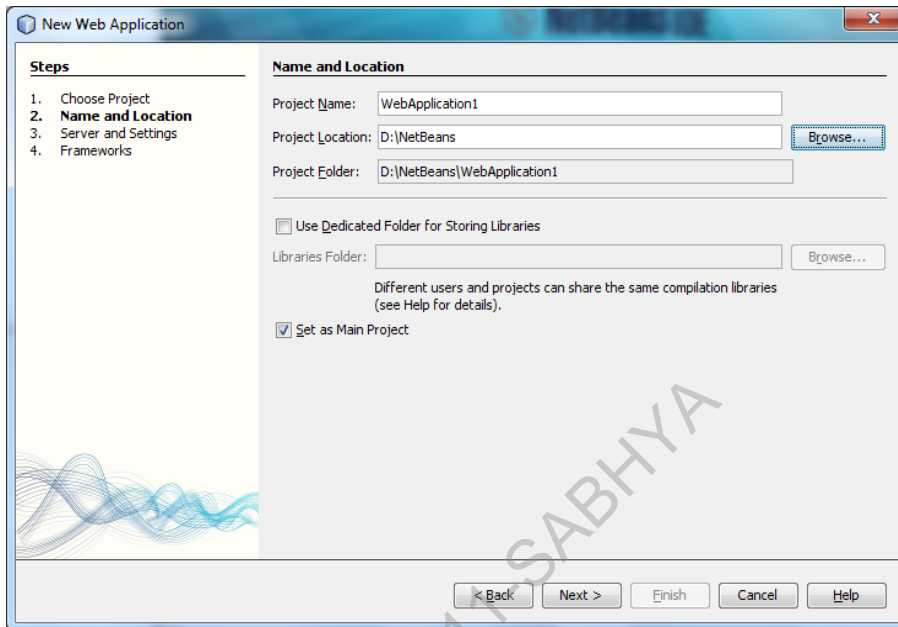
2. Select **File→New Project**. The **New Project** dialog box is displayed, as shown in the following figure.



The New Project Dialog Box

3. Ensure that **Java Web** is selected in the **Categories** section.
4. Ensure that **Web Application** is selected in the **Projects** section.

5. Click the **Next** button. The **New Web Application** wizard is displayed, as shown in the following figure.



The New Web Application Wizard

Note

If you are creating the project for the first time, the **Finding Features** page appears for a few moments before the **New Web Application** wizard is displayed.

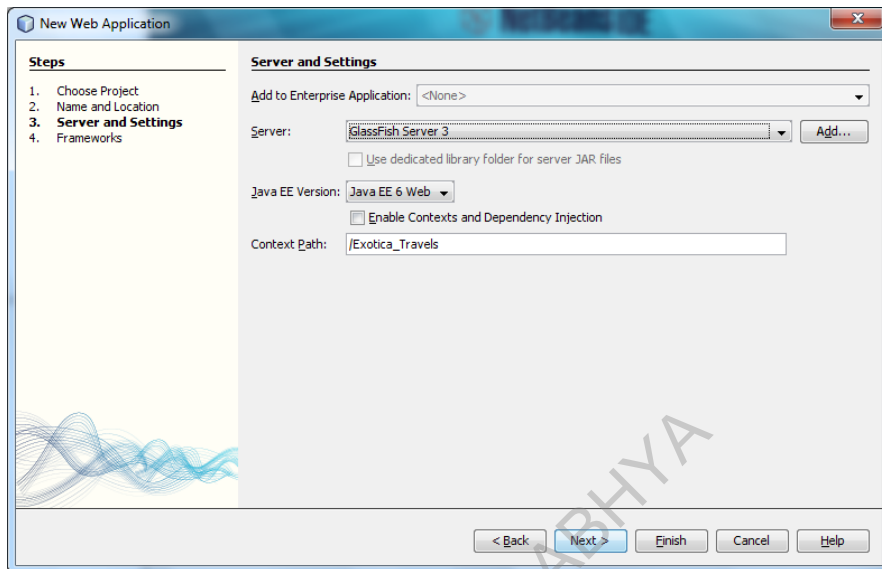
6. Replace the existing text with **ExoticaTravels** as the name of the Web application in the **Project Name** text box.
7. Replace the existing text with **C:\Chapter02\Activity1\Solution** in the **Project Location** text box to specify the directory, where you want to store the Web application.

Note

You can also click the **Browse** button to find the location of the Web application. You can also store the Web application at a location different from the one specified in the preceding step.

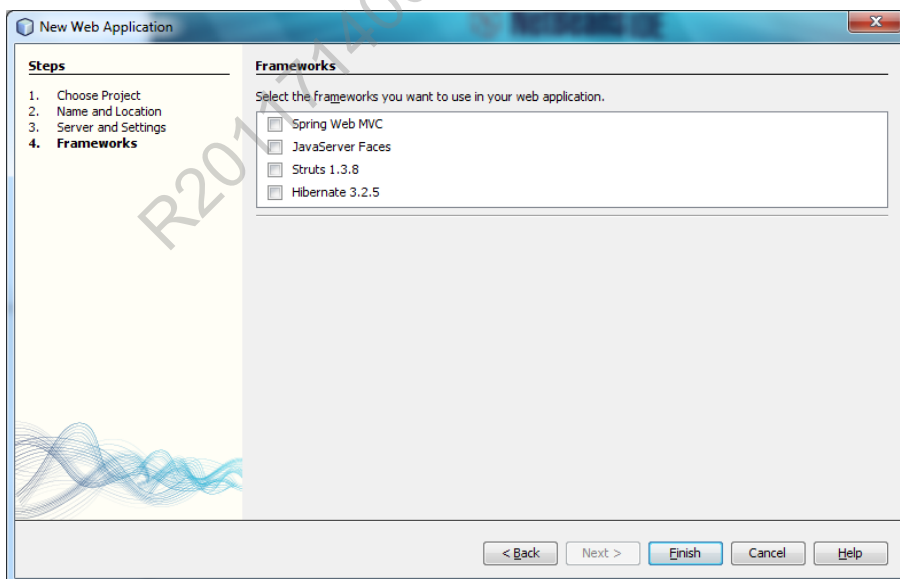
8. Ensure that the **Set as Main Project** check box is selected.

9. Click the **Next** button. The **Server and Settings** page is displayed, as shown in the following figure.



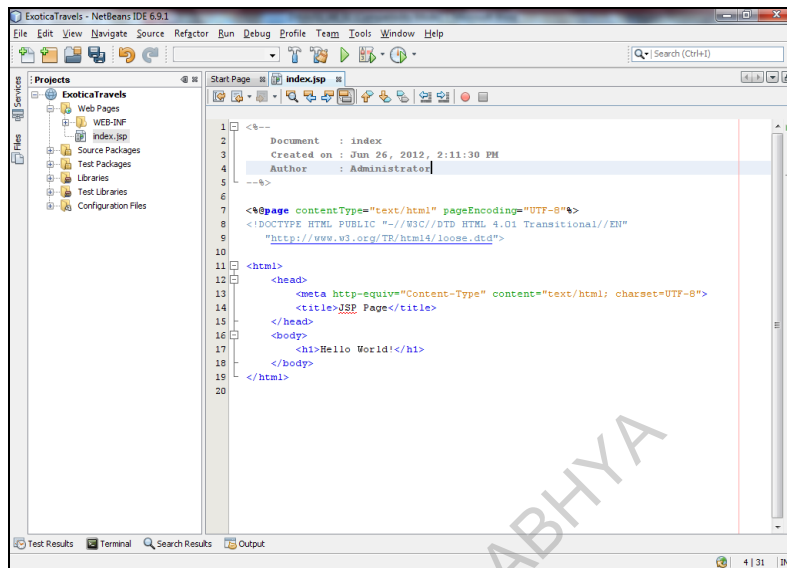
The Server and Settings Page

10. Ensure that **GlassFish Server 3** is selected in the **Server** drop-down list.
11. Ensure that **Java EE 6 Web** is selected in the **Java EE Version** drop-down list.
12. Click the **Next** button. The **Frameworks** page is displayed, as shown in the following figure.



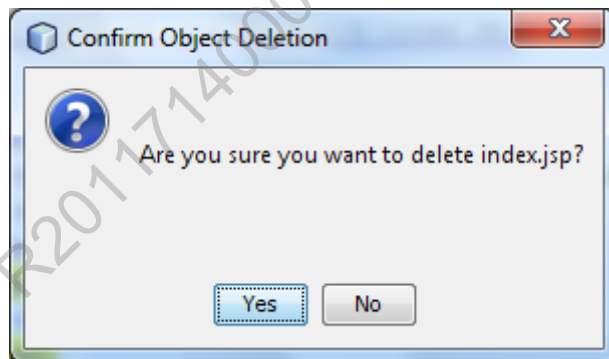
The Frameworks Page

13. Click the **Finish** button. The content of the **index.jsp** file is displayed, as shown in the following figure.



The Content of the index.jsp File

14. Right-click the **index.jsp** file in the **Projects** window, and then select **Delete**. The **Confirm Object Deletion** dialog box appears, as shown in the following figure.



The Confirm Object Deletion Dialog Box

15. Click the **Yes** button.

Task 2: Creating the Images Folder

To create the **Images** folder, you need to perform the following steps:

1. Right-click the **Web Pages** node in the **Projects** window, and then select **New**→**Other**. The **New File** dialog box appears.
2. Select **Other** in the **Categories** section.
3. Select **Folder** in the **File Types** section.
4. Click the **Next** button. The **New Folder** wizard appears.

5. Replace the existing text with **Images** in the **Folder Name** text box.
6. Click the **Finish** button. The **Images** node is created under the **Web Pages** node in the **Projects** window.
7. Open the Windows Explorer window.



Note

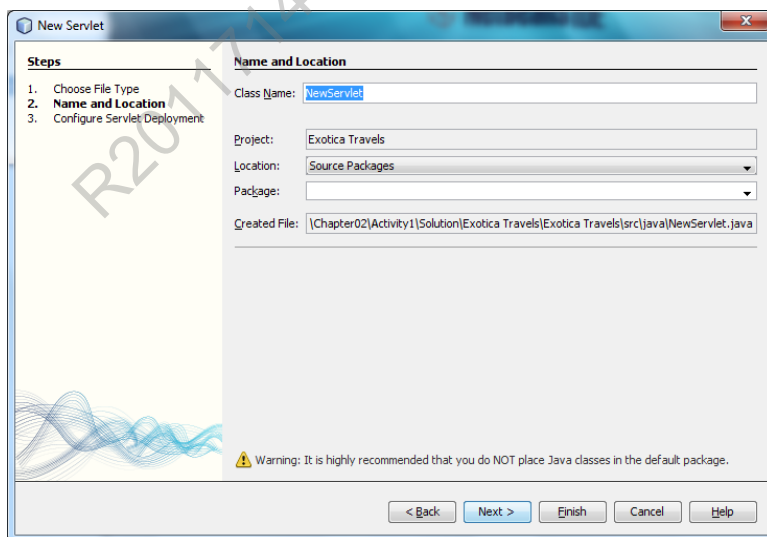
Select **Start** → **All Programs** → **Accessories** → **Windows Explorer** to open the Windows Explorer window.

8. Browse to the **C:\Chapter02\Activity1\Input\Images** folder.
9. Copy the **CompanyLogo.png** image.
10. Close the Windows Explorer window.
11. Right-click the **Images** node in the **Projects** window, and then select **Paste**.

Task 3: Creating the Home Page

To create the home page, you need to perform the following steps:

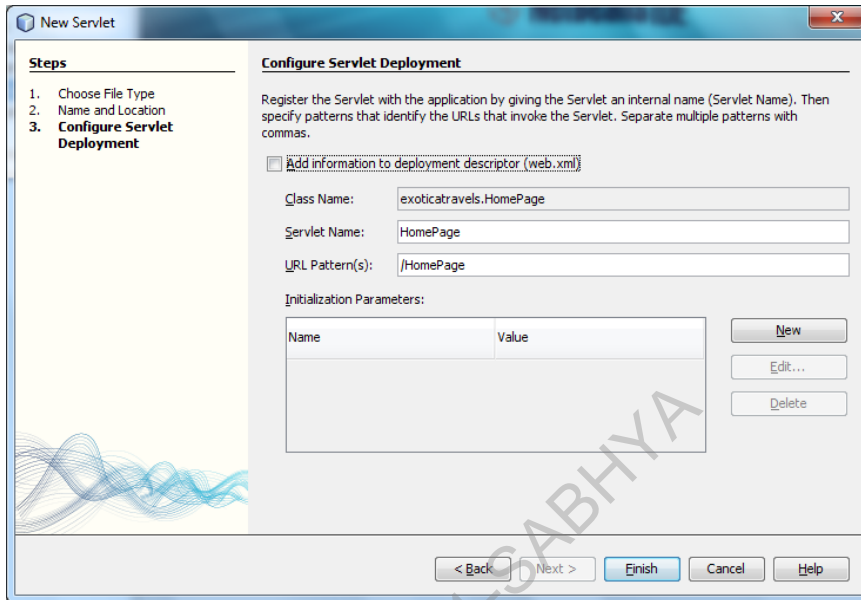
1. Right-click the **Source Packages** node in the **Projects** window, and then select **New** → **Other**. The **New File** dialog box appears.
2. Select **Web** in the **Categories** section.
3. Select **Servlet** in the **File Types** section.
4. Click the **Next** button. The **New Servlet** wizard appears, as shown in the following figure.



The New Servlet Wizard

5. Replace the existing text with **HomePage** in the **Class Name** text box.
6. Type **exoticatravels** in the **Package** combo box.

7. Click the **Next** button. The **Configure Servlet Deployment** page appears, as shown in the following figure.



The Configure Servlet Deployment Page

8. Click the **Finish** button. The content of the **HomePage.java** file is displayed.
9. Replace the existing code within the `try` block of the **HomePage.java** file with the following code snippet:

```
out.println("<html>");
out.println("<head>");
out.println("<title>Home Page</title>");
out.println("</head>");
out.println("<body>");
out.println("<table border='0' bgcolor='#000080' align='top' width='100%'
style='height:100px'>");
    out.println("<tr>");
    out.println("<td bgcolor='#000080' align='center'>");
    out.println("<font style='font-family: 'Arial Rounded MT Bold',
Gadget, sans-serif;' size='+4' color='#FFE4B5'>Exotica Travels</font>");
    out.println("</td>");
    out.println("<td bgcolor='#000080' align='left'
width='150'>");
        out.println("<img src='Images/CompanyLogo.png' width='200'
height='120' align='right'>");
    out.println("</td>");
    out.println("</tr>");
    out.println("</table>");
    out.println("<br/>");
    out.println("<B><I><font style='font-family: 'Brush Script MT
Italics', Gadget, sans-serif;' size='+1' color='darkblue'>Exotica Travels is
a travel management company established by Jordan Desilva in Colombo, Sri
```

```

Lanka. Today under the chairmanship of Jim Henry, the company has spread
across the country. It provides online air tickets booking. In addition, it
provides hotel suite booking in various exotic locations all over the world.
Moreover, it provides rental car bookings. To avail the travel package
services kindly log on to the website.</B></I>");
    out.println("<br/>");
    out.println("<form method='Post' action='SubmitServlet'>");
    out.println("<table cellpadding='10' align='center'>");
    out.println("<tr>");
    out.println("<td><font color='darkblue' size='+2'>LOGIN AS:");
    out.println("</tr>");
    out.println("</td>");
    out.println("</table>");
    out.println("<table cellpadding='10'
align='center' border='2' bordercolor='black'>");
    out.println("<tr><td bordercolor='white'><input type='Radio'
value='existinguser' name='r2' CHECKED>Existing User");
    out.println("<tr><td bordercolor='white'><input
type='Radio' value='newuser' name='r2'>New User");
    out.println("<tr><td bordercolor='white'><input
type='Submit' value='Submit'>");
    out.println("</table>");
    out.println("</form>");
    out.println("</body>");
    out.println("</html>");

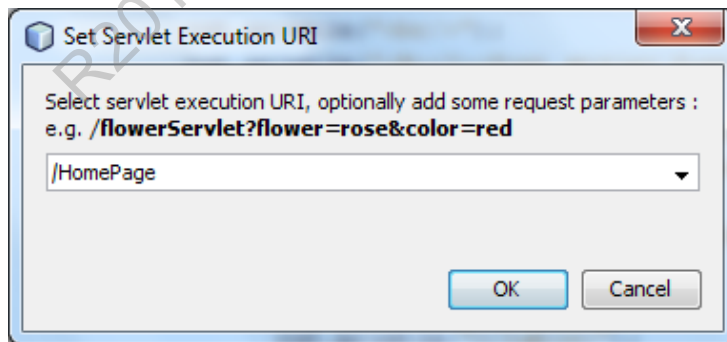
```

10. Select **File**→**Save** to save the **HomePage.java** file.

Task 4: Verifying the Output

To verify the output, you need to perform the following steps:

1. Right-click the **HomePage.java** file in the **Projects** window, and then select **Run File**. The **Set Servlet Execution URI** dialog box appears, as shown in the following figure.



The Set Servlet Execution URI Dialog Box

2. Click the **OK** button.

3. The home page appears in the Web browser, as shown in the following figure.



The Home Page

4. Close the browser window.

Task 5: Closing the Project

To close the project, you need to perform the following steps:

1. Right-click **ExoticaTravels** in the **Projects** window, and then select **Close**.
2. Exit **NetBeans IDE 6.9.1**.

Exercises

Exercise 1

The project development team of DevSoft Corporation has started the development of the TechWrite Web application. You, being the part of the team, have been asked to create the home page of the Web application.

Prerequisite: You need to use the images under the **Input → Images** folder to complete the exercise.

R201171400011-SABHYA

The background of the entire page is a light blue network diagram. It consists of numerous small circular nodes connected by thin, light blue lines, forming a complex web-like structure. The nodes are distributed across the page, with some clusters being denser than others. The overall effect is a technical, digital aesthetic.

NIIT

R201171400011-SABHYA

Exploring JavaServer Pages Technology

CHAPTER 3



Activity 3.1: Creating JSP Pages

Problem Statement

While evaluating the various Web pages of the Exotica Travels website, you found that every page has extensive coding pertaining to the UI of the page. In addition, the page contains the presentation and business logic code. This makes the page heavy and difficult to modify, debug, and manage. Therefore, you have decided to rewrite the UI-related code in JSP and retain the business logic in servlets.

In addition, the development team has decided to enhance the UI of every page of the website. They have suggested adding a sidebar image to each page of the website, as shown in the following figure.



The Sidebar Image

Prerequisite

You will need the **Input** folder to perform the activity. The **Input** folder contains the starter files needed to complete the activity.

Solution

To rewrite the UI-related code in JSP, you need to perform the following tasks:

1. Open the **ExoticaTravels** project.
2. Rewrite the UI using JSP.
3. Verify the output.
4. Close the project.

Task 1: Opening the ExoticaTravels Project

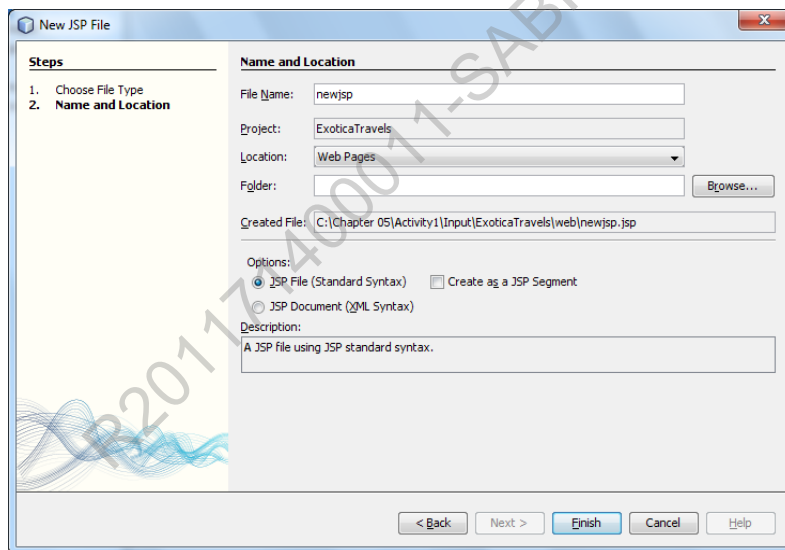
To open the **ExoticaTravels** project, you need to perform the following steps:

1. Open **NetBeans IDE 6.9.1**. The **NetBeans IDE 6.9.1** window is displayed.
2. Select **File→Open Project**. The **Open Project** dialog box is displayed.
3. Browse to the **C:\Chapter05\Activity1\Input** folder.
4. Select the **ExoticaTravels** project, and then click the **Open Project** button. The **ExoticaTravels** project is displayed in the **Projects** window.

Task 2: Rewriting the UI Using JSP

To rewrite the UI using JSP, you need to perform the following steps:

1. Right-click the **Web Pages** node in the **Projects** window, and then select **New→Other**. The **New File** dialog box appears.
2. Select **Web** in the **Categories** section.
3. Select **JSP** in the **File Types** section.
4. Click the **Next** button. The **New JSP File** wizard appears, as shown in the following figure.



The New JSP File Wizard

5. Replace the existing text with **Template** in the **File Name** text box.
6. Ensure that the **JSP File (Standard Syntax)** option is selected in the **Options** section.
7. Click the **Finish** button. The content of the **Template.jsp** file is displayed.
8. Similarly, create the following files:
 - HomePage.jsp
 - LoginPage.jsp
 - RegistrationPage.jsp
 - WelcomeAdminPage.jsp

- WelcomeCustomerPage.jsp
- GetLogDetails.jsp
- ErrorPage.jsp
- TourPackage.jsp
- Hotels.jsp
- Flights.jsp
- Cart.jsp
- ConstructionPage.jsp

9. Open Windows Explorer.

10. Browse to the **C:\Chapter05\Activity1\Input** folder.

11. Copy the **Sidebar.png** image.

12. Close the Windows Explorer window.

13. Right-click the **Images** node in the **Projects** window, and then select **Paste**.

14. Double-click the **Template.jsp** file in the **Projects** window. The content of the **Template.jsp** file is displayed.

15. Replace the existing code after the `<!DOCTYPE>` tag of the **Template.jsp** file with the following code snippet:

```
<table border='0' bgcolor='#000080' align='top' width='100%'
style='height:100px'>
<tr>
<td bgcolor='#000080' align='center'>
<font style='font-family: 'Arial Rounded MT Bold', Gadget,
sans-serif;' size='+4' color='#FFE4B5'>Exotica Travels</font>
</td>
<td bgcolor='#000080' align='left' width='150'>
<img src='Images/CompanyLogo.png' width='200' height='120' align='right' />
</td>
</tr>
</table>
<table bgcolor="white" border="0" align="top" width="100%"
style="height:470px">
<tr>
<td style="width: 216px;" colspan="1" rowspan="4"></td>
```

16. Double-click the **HomePage.jsp** file in the **Projects** window. The content of the **HomePage.jsp** file is displayed.

17. Replace the existing code within the `<html>...</html>` tags of the **HomePage.jsp** file with the following code snippet:

```
<head>
<title>Home Page</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top"> <br/><br/><B><I><font style='font-family: 'Brush Script MT
Italics', Gadget, sans-serif;' size='+1' color='darkblue'>Exotica Travels is
a travel management company established by Jordan Desilva in Colombo, Sri
Lanka. Today under the chairmanship of Jim Henry, the company has spread
across the country. It provides online air tickets booking. In addition, it
provides hotel suite booking in various exotic locations all over the world.
Moreover, it provides rental car bookings. To avail the travel package
services kindly log on to the website.</B></I>
<br/>
<br/>
<form method='Post' action='SubmitServlet'>
<table cellpadding='10' align='center'>
<tr>
<td><font color='darkblue' size='+2'>LOGIN AS:
</tr>
</td>
</table>
<table cellpadding='10' align='center' border='2' bordercolor='black'>
<tr><td bordercolor='white'><input type='Radio' value='existinguser'
name='r2' CHECKED>Existing User
<tr><td bordercolor='white'><input type='Radio' value='newuser' name='r2'>New
User
<tr><td bordercolor='white'><input type='Submit' value='Submit'>
</table>
</form>
</tr>
</table>
<body>
```

18. Double-click the **LoginPage.jsp** file in the **Projects** window. The content of the **LoginPage.jsp** file is displayed.
19. Replace the existing code within the `<html>...</html>` tags of the **LoginPage.jsp** file with the following code snippet:

```
<head>
<title>Login Page</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top">
<form action='ValidationServlet' method='Post'>
<table cellpadding='10' align='center'>
<tr>
<td><font color='darkblue' size='+2'>Login Form
</tr>
```

```

</td>
</table>
<table cellpadding='10' align='center' border='2' bordercolor='black'>
<tr><td bordercolor='white'>User Id:<td COLSPAN='2' bordercolor='white'><input
type='text' name='uid'>
<tr><td bordercolor='white'>Password:<td COLSPAN='2'
bordercolor='white'><input type='password' name='pwd'>
<tr><td bordercolor='white'>Role:<td bordercolor='white'><input
type='Radio' value='administrator' name='r1'>Administrator
<td bordercolor='white'><input type='Radio' value='customer' name='r1'>Customer
<tr><td bordercolor='white'></br>
<tr><td bordercolor='white'><input type='Submit' value='Submit'>
<td bordercolor='white'><input type='RESET' text='Reset'>
</table>
</form>
</body>

```

20. Double-click the **RegistrationPage.jsp** file in the **Projects** window. The content of the **RegistrationPage.jsp** file is displayed.
21. Replace the existing code within the `<html>...</html>` tags of the **RegistrationPage.jsp** file with the following code snippet:

```

<head>
<title>Registration Page</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top">
<form method='Post' action='submitServlet'>
<table cellpadding='10' align='center'>
<tr>
<td><font color='darkblue' size='+2'>New User Registration Form
</tr>
</td>
</table>
<table cellpadding='10' align='center' border='2' bordercolor='black'>
<tr><td bordercolor='white'>User Name:<td
COLSPAN='2' bordercolor='white'><input type='text' name='tt1'>
<tr><td bordercolor='white'>User Id:<td COLSPAN='2' bordercolor='white'>
<input type='text' name='tt2'>
</tr>
<tr><td bordercolor='white'>Password:<td COLSPAN='2' bordercolor='white'>
<input type='password' name='tt3'>
</tr>
<tr><td bordercolor='white'>Re-enter Password:<td COLSPAN='2'
bordercolor='white'><input type='password' name='tt4'>
<tr><td bordercolor='white'>Address:<td COLSPAN='2'
bordercolor='white'><input type='textArea' name='tt5'>
<tr><td bordercolor='white'>State:<td COLSPAN='2' bordercolor='white'><input
type='text' name='tt6'>
<tr><td bordercolor='white'>Country:<td COLSPAN='2'
bordercolor='white'><input type='text' name='tt7'>
<tr><td bordercolor='white'></br>

```

```

<tr><td bordercolor='white'><input type='Submit' value='Submit'>
<td bordercolor='white'><input type='RESET' text='Reset'>
</table>
</form>
</body>

```

22. Double-click the **WelcomeAdminPage.jsp** file in the **Projects** window. The content of the **WelcomeAdminPage.jsp** file is displayed.
23. Replace the existing code within the `<html>...</html>` tags of the **WelcomeAdminPage.jsp** file with the following code snippet:

```

<head>
<title>Admin Page</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top" rowspan="200" colspan="80">
<table>
<tr><td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Domestic Flights</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>International Flights</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Hotels</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Car Rentals</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("TourPackage.jsp").toString());%>'>Tou
r Packages</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("GetLogDetails.jsp").toString());%>'>G
et Log Details</a></td>
</tr>
<tr>
<td><br></td>
<tr>
<td rowspan="200" colspan="80">
<font color='darkblue' size='+2'>Welcome to the Exotica Travels Website!!
</table>
</body>

```

24. Double-click the **WelcomeCustomerPage.jsp** file in the **Projects** window. The content of the **WelcomeCustomerPage.jsp** file is displayed.

25. Replace the existing code within the `<html>...</html>` tags of the **WelcomeCustomerPage.jsp** file with the following code snippet:

```
<head>
<title>Customer Page</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top" rowspan="200" colspan="80">
<table><tr>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Domestic Flights</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>International Flights</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Hotels</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Car Rentals</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("TourPackage.jsp").toString());%>'>Tou
r Packages</a></td>
</tr>
<tr>
<td><br></td>
</tr>
<td rowspan="200" colspan="80">
<font color='darkblue' size='+2'>Welcome to the Exotica Travels Website!!
</table>
</body>
```

26. Double-click the **GetLogDetails.jsp** file in the **Projects** window. The content of the **GetLogDetails.jsp** file is displayed.
27. Replace the existing code within the `<html>...</html>` tags of the **GetLogDetails.jsp** file with the following code snippet:

```
<head>
<title>GetLogDetailsPage</title>
</head>
<body>
<jsp:include page="Template.jsp"></jsp:include>
<td valign="top" rowspan="200" colspan="80">
<table>
<tr>
<td><jsp:include page="/GetLogDetails">
<jsp:param name="title" value="GetLogDetails"/>
</jsp:include>
</td>
</tr>
</table>
</body>
```

28. Double-click the **ErrorPage.jsp** file in the **Projects** window. The content of the **ErrorPage.jsp** file is displayed.
29. Replace the existing code within the `<html>...</html>` tags of the **ErrorPage.jsp** file with the following code snippet:

```
<head>
<title>Error Page</title>
</head>
<body>
    <jsp:include page="Template.jsp"></jsp:include>
    <td valign="top" rowspan="200" colspan="80">
        <table>
            <tr>
<td>
<font color='red' size='+2'>Sorry!! Blocked Access for this IP.
            </table>
        </body>
```

30. Double-click the **TourPackage.jsp** file in the **Projects** window. The content of the **TourPackage.jsp** file is displayed.
31. Replace the existing code within the `<html>...</html>` tags of the **TourPackage.jsp** file with the following code snippet:

```
<head>
<title>TourPackage Page</title>
</head>
<body>
    <jsp:include page="Template.jsp"></jsp:include>
    <td valign="top" rowspan="200" colspan="80">
        <table>
            <tr>
<td><font color='darkblue' size='+1'><a href='ConstructionPage.jsp '>Domestic
Flights</a></td>
<td><font color='darkblue' size='+1'><a href='ConstructionPage.jsp
'>International Flights</a></td>
<td><font color='darkblue' size='+1'><a href='ConstructionPage.jsp
'>Hotels</a></td>
<td><font color='darkblue' size='+1'><a href='ConstructionPage.jsp'>Car
Rentals</a></td>
<td><font color='darkblue' size='+1'><a href='TourPackage.jsp'>Tour
Packages</a></td>
</tr>
</table>
<br>
<font align='center' color='darkblue' size='+1'>Please select the destination
where you would like to go:
<br>
<br>
<form
action='<%out.println(response.encodeURL("SessionServlet").toString());%>'
name='MyForm' Method='POST'>
<table cellpadding='10' align='center' border='2' bordercolor='black'>
```


[illegible]

34. Double-click the **Flights.jsp** file in the **Projects** window. The content of the **Flights.jsp** file is displayed.
35. Replace the existing code within the `<html>...</html>` tags of the **Flights.jsp** file with the following code snippet:

```
<head>
<title>Flights Page</title>
</head>
<body>
    <jsp:include page="Template.jsp"></jsp:include>
<td valign="top" rowspan="200" colspan="80">
    <table>
        <tr>
<tr>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Domestic Flights</a></td>
<td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>International Flights</a></td>
```

[illegible]

36. Double-click the **Cart.jsp** file in the **Projects** window. The content of the **Cart.jsp** file is displayed.

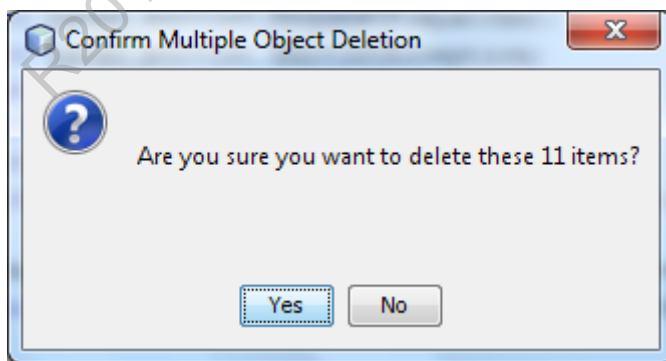
37. Replace the existing code within the `<html>...</html>` tags of the **Cart.jsp** file with the following code snippet:

```
<head>
<title>Cart Page</title>
</head>
<body>
    <jsp:include page="Template.jsp"></jsp:include>
    <td valign="top" rowspan="200" colspan="80">
        <table>
            <tr>
                <td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Domestic Flights</a></td>
                <td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>International Flights</a></td>
                <td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Hotels</a></td>
                <td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("ConstructionPage.jsp").toString());%>
'>Car Rentals</a></td>
                <td><font color='darkblue' size='+1'><a
href='<%out.println(response.encodeURL("TourPackage.jsp").toString());%>'>Tou
r Packages</a></td>
            </tr>
        </table>
        <br>
        <font align='center' color='darkblue' size='+1'>You have added the following
contents to your cart:
        </font>
        <br>
        <br>
        <br>
        <table cellpadding='10' align='center' border='2' bordercolor='black'>
        <tr><td bordercolor='white'>Destination:<td
COLSPAN='2' bordercolor='white'><%out.println(session.getAttribute("selDestina
tion").toString()); %>
        <tr><td bordercolor='white'>Hotel:<td COLSPAN='2'
bordercolor='white'><%out.println(session.getAttribute("selHotel").toString()
); %>
        <tr><td bordercolor='white'>Total number of days for stay:<td
bordercolor='white'><%out.println(session.getAttribute("numDays").toString()
); %>
        <tr><td bordercolor='white'>Airline:<td
bordercolor='white'><%out.println(session.getAttribute("selAirline").toString
()); %>
        </table>
    </body>
```

38. Double-click the **ConstructionPage.jsp** file in the **Projects** window. The content of the **ConstructionPage.jsp** file is displayed.
39. Replace the existing code within the `<html>...</html>` tags of the **ConstructionPage.jsp** file with the following code snippet:

```
<head>
<title>Under Construction Page</title>
</head>
<body>
<h1>Page under construction</h1>
</body>
```

40. Expand the **Source Packages**→**exoticatravels** node in the **Projects** window.
41. Select the **CartServlet.java** file.
42. Press and hold the **Ctrl** key, and then select the following files:
- ConstructionServlet.java
 - ErrorServlet.java
 - FlightsServlet.java
 - HomePage.java
 - HotelServlet.java
 - LoginServlet.java
 - RegistrationServlet.java
 - TourPackageServlet.java
 - WelcomeAdminServlet.java
 - WelcomeCustomerServlet.java
43. Right-click the selected files, and then select **Delete**. The **Confirm Multiple Object Deletion** dialog box appears, as shown in the following figure.



The Confirm Multiple Object Deletion Dialog Box

44. Click the **Yes** button.
45. Double-click the **GetLogDetails.java** file in the **Projects** window. The content of the **GetLogDetails.java** file is displayed.

46. Select the code between the `PrintWriter out = response.getWriter();` and `File file = new File("log.txt");` lines in the `processRequest()` method.
47. Press the **Delete** key to delete the selected code.
48. Double-click the **IPFilter.java** file in the **Projects** window. The content of the **IPFilter.java** file is displayed.
49. Replace the existing code within the `if{...}` construct in the `doFilter()` method with the following code snippet:

```
RequestDispatcher rd=request.getRequestDispatcher("GetLogDetails.jsp");
rd.forward(request,response);
```

50. Replace the existing code within the `else{...}` construct with the following code snippet:

```
RequestDispatcher rd=request.getRequestDispatcher("ErrorPage.jsp");

rd.forward(request,response);
```

51. Double-click the **SessionServlet.java** file in the **Projects** window. The content of the **SessionServlet.java** file is displayed.
52. Replace the line, `RequestDispatcher dispatch = getServletContext().getRequestDispatcher("/HotelServlet");` in the `if{...}` construct with the following code snippet:

```
RequestDispatcher dispatch =
    request.getRequestDispatcher("Hotels.jsp");
```

53. Replace the line, `RequestDispatcher dispatch = getServletContext().getRequestDispatcher("/FlightsServlet");`, in the first `else if{...}` construct with the following code snippet:

```
RequestDispatcher dispatch =
    request.getRequestDispatcher("Flights.jsp");
```

54. Replace the line, `RequestDispatcher dispatch = getServletContext().getRequestDispatcher("/CartServlet");` in the second `else if{...}` construct with the following code snippet:

```
RequestDispatcher dispatch =
    request.getRequestDispatcher("Cart.jsp");
```

55. Double-click the **SubmitServlet.java** file in the **Projects** window. The content of the **SubmitServlet.java** file is displayed.

56. Replace the existing code snippet in the `if{...}` construct in the `processRequest()` method with the following code snippet:

```
RequestDispatcher dispatch =  
request.getRequestDispatcher("RegistrationPage.jsp");  
dispatch.forward(request, response);
```

57. Replace the existing code snippet in the `else if{...}` construct with the following code snippet:

```
RequestDispatcher dispatch = request.getRequestDispatcher("LoginPage.jsp");  
dispatch.forward(request, response);
```

58. Double-click the **ValidationServlet.java** file in the **Projects** window. The content of the **ValidationServlet.java** file is displayed.

59. Replace the line, `RequestDispatcher dispatch =`
`getServletContext().getRequestDispatcher("/WelcomeAdminServlet");`, in the `if{...}`
construct with the following code snippet:

```
RequestDispatcher dispatch =  
request.getRequestDispatcher("WelcomeAdminPage.jsp");
```

60. Replace the line, `RequestDispatcher dispatch =`
`getServletContext().getRequestDispatcher("/WelcomeCustomerServlet");`, within the
`else if{...}` construct with the following code snippet:

```
RequestDispatcher dispatch =  
request.getRequestDispatcher("WelcomeCustomerPage.jsp");
```

61. Select **File**→**Save All** to save the modified files.

Task 3: Verifying the Output

To verify the output, you need to perform the following steps:

1. Right-click the **ExoticaTravels** node in the **Projects** window, and then select **Properties**. The **Project Properties - ExoticaTravels** dialog box appears.
2. Select the **Run** node in the **Categories** section.
3. Type **/HomePage.jsp** in the Relative URL text box.
4. Click the **OK** button.

5. Right-click the **ExoticaTravels** node in the **Projects** window, and then select **Run**. The **Home Page** page appears in the Web browser, as shown in the following figure.



The Home Page

6. Ensure that the **Existing User** option is selected under the **LOGIN AS** section.
7. Click the **Submit** button. The **Login Page** page is displayed, as shown in the following figure.



The Login Page

8. Type **user1** and **user1@123** in the **User Id** and **Password** text boxes, respectively.
9. Select the **Customer** option.

10. Click the **Submit** button. The **Customer Page** is displayed, as shown in the following figure.



The Customer Page

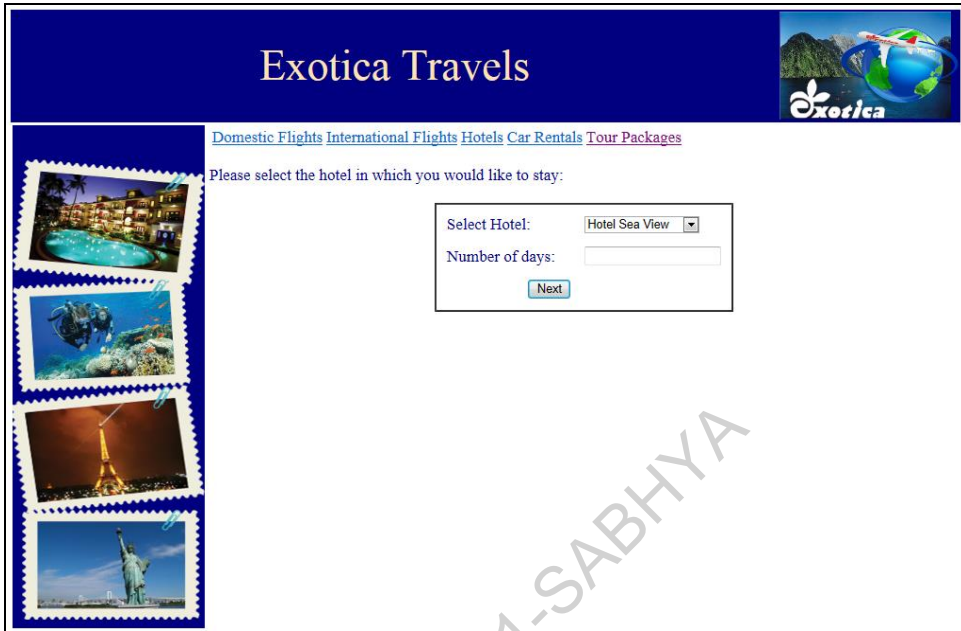
11. Click the **Tour Packages** link. The **TourPackage Page** is displayed, as shown in the following figure.



The TourPackage Page

12. Select **Bahamas** from the **Select Destination** drop-down list.

13. Click the **Next** button. The **Hotel Page** page is displayed, as shown in the following figure.



The screenshot shows the 'Exotica Travels' website. The header is dark blue with the company name in white. To the right is a logo featuring a globe and a plane. Below the header, there are navigation links: [Domestic Flights](#), [International Flights](#), [Hotels](#), [Car Rentals](#), and [Tour Packages](#). The main content area has a blue background on the left with four travel-related images: a resort at night, a scuba diver, a lighthouse, and a statue. The right side is white and contains the text 'Please select the hotel in which you would like to stay:'. Below this is a form with a 'Select Hotel:' label, a dropdown menu showing 'Hotel Sea View', a 'Number of days:' label, a text input field, and a 'Next' button.

Exotica Travels

[Domestic Flights](#) [International Flights](#) [Hotels](#) [Car Rentals](#) [Tour Packages](#)

Please select the hotel in which you would like to stay:

Select Hotel:

Number of days:

The Hotel Page

14. Select **Hotel Sea Breeze** from the **Select Hotel** drop-down list.
15. Type **2** in the **Number of days** text box.

16. Click the **Next** button. The **Flights Page** page is displayed, as shown in the following figure.



Exotica Travels

[Domestic Flights](#) [International Flights](#) [Hotels](#) [Car Rentals](#) [Tour Packages](#)

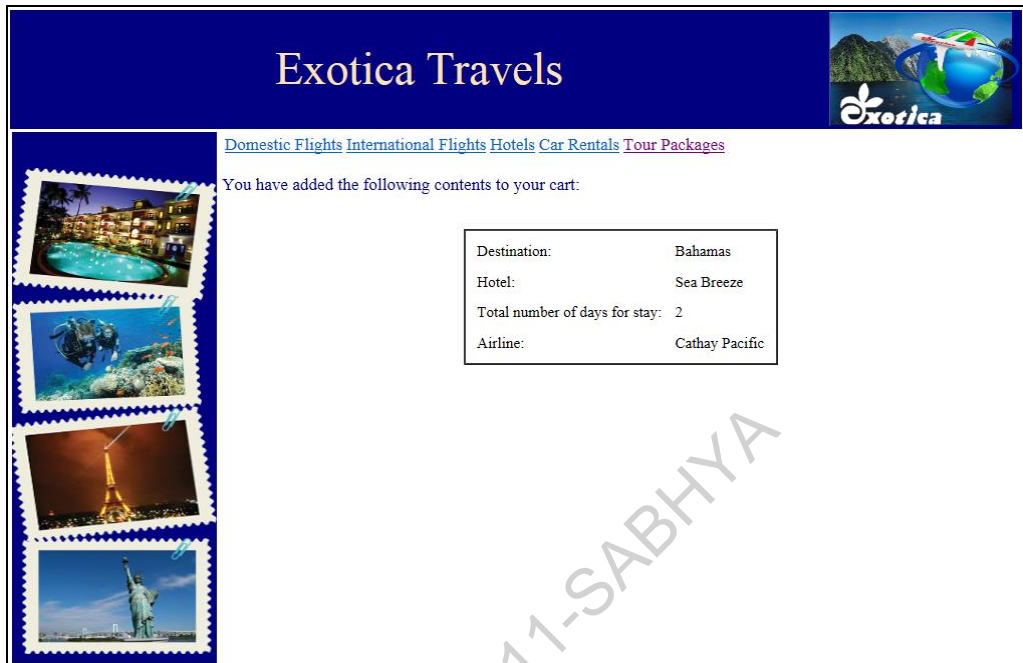
Please select the airline by which you want to go to your destination:

Select Airline:

The Flights Page

17. Select **Cathay Pacific** from the **Select Airline** drop-down list.

18. Click the **Next** button. The **Cart Page** page is displayed, as shown in the following figure.



The Cart Page

19. Close the browser window.

Task 4: Closing the Project

To close the project, you need to perform the following steps:

1. Right-click the **ExoticaTravels** node in the **Projects** window, and then select **Close**.
2. Close **NetBeans IDE 6.9.1**.

R201171400011-SABHYA

NIIT

This book is a personal copy of SABHYA of NIIT Noida Sector 62 Centre , issued on 07/12/2019.

No Part of this book may be distributed or transferred to anyone in any form by any means.