BEA WebLogic Commerce Server

Registration and User Processing Package

BEA WebLogic Commerce Server 3.2
Document Edition 3.2
December 2000

Registration and User Processing Package

Document Edition	Date	Software Version
3.2	December 2000	BEA WebLogic Commerce Server 3.2

Contents

About This Document

What You Need to Know ix
e-docs Web Site x
How to Print the Document x
Related Information xi
Contact Us! xi
Documentation Conventions xii

1. Overview of the Registration and User Processing Package

What Is the Registration and User Processing Package? 1-2
High-level Architecture 1-4
About the Database Schema 1-7
Development Roles 1-7
Next Steps 1-8

2. Customer Registration and Login Services

JavaServer Pages (JSPs) 2-2
- login.jsp Template 2-2
  - Sample Browser View 2-2
  - Location in the WebLogic Commerce Server Directory Structure 2-4
  - Tag Library Imports 2-4
  - Java Package Imports 2-4
  - Location in Default Webflow 2-4
  - Included JSP Templates 2-5
  - Events 2-6
  - Dynamic Data Display 2-6
  - Form Field Specification 2-6

badlogin.jsp Template 2-8
Sample Browser View 2-8
Location in the WebLogic Commerce Server Directory Structure 2-9
Tag Library Imports 2-10
Java Package Imports 2-10
Location in Default Webflow 2-10
Included JSP Templates 2-10
Events 2-10
Dynamic Data Display 2-11
Form Field Specification 2-11
successfullogin.jsp Template 2-12
Sample Browser View 2-12
Location in the WebLogic Commerce Server Directory Structure 2-15
Tag Library Imports 2-15
Java Package Imports 2-15
Location in Default Webflow 2-15
Included JSP Templates 2-16
Events 2-16
Dynamic Data Display 2-17
Form Field Specification 2-18
newuser.jsp Template 2-19
Sample Browser View 2-19
Location in the WebLogic Commerce Server Directory Structure 2-22
Tag Library Imports 2-22
Java Package Imports 2-22
Location in Default Webflow 2-23
Included JSP Templates 2-23
Events 2-26
Dynamic Data Display 2-26
Form Field Specification 2-27
Input Processors 2-30
CustomerProfileIP 2-30
Pipeline Components 2-33
RegisterUserPC 2-33

3. Customer Profile Services

JavaServer Pages (JSPs) 3-2
viewprofile.jsp Template 3-2
Sample Browser View 3-3
Location in the WebLogic Commerce Server Directory Structure..... 3-4
Tag Library Imports 3-5
Java Package Imports 3-5
Location in Default Webflow 3-6
Included JSP Templates 3-6
Events 3-7
Dynamic Data Display 3-8
Form Field Specification 3-12
editprofile.jsp Template 3-13
Sample Browser View 3-13
Location in the WebLogic Commerce Server Directory Structure... 3-15
Tag Library Imports 3-15
Java Package Imports 3-16
Location in Default Webflow 3-16
Included JSP Templates 3-16
Events 3-17
Dynamic Data Display 3-18
Form Field Specification 3-21
profilenewaddress.jsp Template 3-23
Sample Browser View 3-23
Location in the WebLogic Commerce Server Directory Structure... 3-24
Tag Library Imports 3-24
Java Package Imports 3-25
Location in Default Webflow 3-25
Included JSP Templates 3-25
Events 3-26
Dynamic Data Display 3-26
Form Field Specification 3-27
profileeditaddress.jsp Template 3-29
Sample Browser View 3-29
Location in WebLogic Commerce Server Directory Structure 3-30

Tag Library Imports 3-30
Java Package Imports 3-31
Location in Default Webflow 3-31
Included JSP Templates 3-31
Events 3-33
Dynamic Data Display 3-34
Form Field Specification 3-35

profilenewcc.jsp Template 3-37
Sample Browser View 3-37
Location in the WebLogic Commerce Server Directory Structure 3-38
Tag Library Imports 3-39
Java Package Imports 3-39
Location in Default Webflow 3-39
Included JSP Templates 3-40
Events 3-40
Dynamic Data Display 3-41
Form Field Specification 3-41

profileeditcc.jsp Template 3-43
Sample Browser View 3-43
Location in the WebLogic Commerce Server Directory Structure 3-44
Tag Library Imports 3-44
Java Package Imports 3-45
Location in Default Webflow 3-45
Included JSP Templates 3-45
Events 3-48
Dynamic Data Display 3-48
Form Field Specification 3-50

changepassword.jsp Template 3-53
Sample Browser View 3-53
Location in the WebLogic Commerce Server Directory Structure 3-54
Tag Library Imports 3-54
Java Package Imports 3-55
Location in Default Webflow 3-55
Included JSP Templates 3-55
Events 3-56

Dynamic Data Display 3-56
Form Field Specification 3-57
Input Processors 3-59
DeleteCreditCardIP 3-59
DeleteShippingAddressIP 3-60
UpdateAccountInfoIP 3-61
UpdateBasicInfoIP 3-62
UpdatePaymentInfoIP 3-63
UpdateShippingInfoIP 3-64
Pipeline Components 3-65
UpdateBasicInfoPC 3-65
UpdatePaymentInfoPC 3-66
UpdateShippingInfoPC 3-66
UpdatePasswordPC 3-67

4. Customer Self-Service

JavaServer Pages (JSPs) 4-2
main.jsp Template 4-2
orderhistory.jsp Template 4-4
Sample Browser View 4-4
Location in the WebLogic Commerce Server Directory Structure 4-5
Tag Library Imports 4-6
Java Package Imports 4-6
Location in Default Webflow 4-6
Included JSP Templates 4-7
Events 4-7
Dynamic Data Display 4-8
Form Field Specification 4-10
orderstatus.jsp Template 4-11
Sample Browser View 4-11
Location in WebLogic Commerce Server Directory Structure 4-13
Tag Library Imports 4-14
Java Package Imports 4-14
Location in Default Web Flow 4-14
Included JSP Templates 4-15

Events 4-15
Dynamic Data Display 4-15
Form Field Specification 4-19
paymenthistory.jsp Template 4-20
Sample Browser View 4-20
Location in the WebLogic Commerce Server Directory Structure ... 4-21
Tag Library Imports 4-21
Java Package Imports 4-22
Location in Default Webflow 4-22
Included JSP Templates 4-22
Events 4-23
Dynamic Data Display 4-23
Form Field Specification 4-26
Input Processors 4-27
SelectOrderForViewingIP 4-27
Pipeline Components 4-28
RefreshOrderHistoryPC 4-28
RefreshPaymentHistoryPC 4-29

Index

About This Document

This document explains how to use the services available within the BEA WebLogic Commerce Server Registration and User Processing package.

This document includes the following topics:

- Chapter 1, “Overview of the Registration and User Processing Package,” which describes the high-level architecture of the package and provides introductory information about its services.
- Chapter 2, “Customer Registration and Login Services,” which describes the JSP templates, input processors, and Pipelines associated with the customer registration and login Web pages.
- Chapter 3, “Customer Profile Services,” which describes the JSP templates, input processors, and Pipelines associated with the customer profile Web pages.
- Chapter 4, “Customer Self-Service,” which describes the JSP templates, input processors, and Pipelines associated with the customer self-service Web pages.

What You Need to Know

This document is intended for the following audiences:

- The commerce engineer/JSP content developer, who uses JSP templates and tag libraries to implement interactive Web pages to meet business requirements. This user also maintains simple configuration files.
- The business analyst, who defines the company’s business protocols (processes and rules) for a business-to-consumer Web site. This user may set pricing policies and discounts, and may plan promotional advertising.

- The site administrator, who uses Commerce and Personalization Server administration screens to configure the site's rules, portals, property sets, user profiles, content delivery, and product catalog.
- The Java/EJB programmer, who creates custom code to insert in the JSP files. This user may also handle complex configuration files.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.beasys.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Commerce Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Commerce Server documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

The following BEA WebLogic Commerce Server documents contain information that is relevant to using the Registration and User Processing package and understanding how to customize or extend the provided services.

- *BEA WebLogic Commerce Server Webflow and Pipeline Management*
- *BEA WebLogic Commerce Server Order Processing Package*
- *BEA WebLogic Commerce Server Product Catalog Management*
- For more information about J2EE as it relates to WebLogic Server security, see the information posted on the Sun Microsystems, Inc. Java(TM) 2 Platform, Enterprise Edition Web site at http://java.sun.com/j2ee/.

Contact Us!

Your feedback on the BEA WebLogic Commerce Server documentation is important to us. Send us e-mail at **docsupport@beasys.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Commerce Server documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Commerce Server 3.1 release.

If you have any questions about this version of BEA WebLogic Commerce Server, or if you have problems installing and running BEA WebLogic Commerce Server, contact BEA Customer Support through BEA WebSupport at **www.beasys.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address

- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Overview of the Registration and User Processing Package

The processes related to customer (user) profiles and customer self-service are necessary components of any e-business expecting return customers. To help you get to market faster than your competitors, the BEA WebLogic Commerce Server product provides you with a Registration and User Processing package. This package contains default implementations for the most common pre- and post-order processing services (registration, login, customer profile creation/updates, and customer self-service pages). Designed to be used out-of-the-box, the Registration and User Processing package also allows your site designers to customize these processes, without the need for advanced programming skills. This topic provides you with some background information about the Registration and User Processing package, and introduces you to the types of services that are available.

This topic includes the following sections:

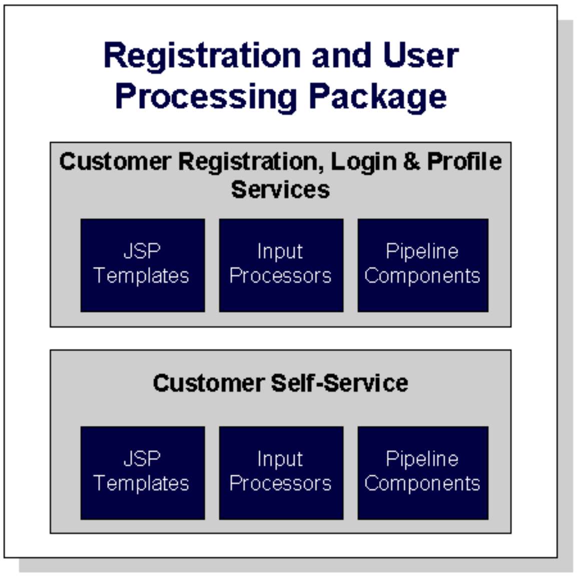
- What Is the Registration and User Processing Package?
- High-level Architecture
- About the Database Schema
- Development Roles
- Next Steps

What Is the Registration and User Processing Package?

The Registration and User Processing package is a collection of services used to facilitate the registration of customers with your e-business site and the activities customer can perform after registering. There are services for registration, login, customer profile creation/updates, and so on. Additionally, the customer self-service pages provide your customers with the ability to check the status of orders and payments.

As shown in Figure 1-1, each service in the package consists of one or more JavaServer Pages (JSPs) and the business logic associated with them. Some of these templates may collect information from your customers, while others will simply display dynamic data your customer previously supplied. Some JSPs may do both. This logic is implemented as a combination of input processors and Pipeline components, each of which can be modified to suit your needs. You can also create your own input processors and Pipeline components to plug into the Registration and User Processing package.

Figure 1-1 Structure of the Registration and User Processing Package



Because all the business logic is managed by a Pipeline and accessed within a Pipeline session, the state of your customer's experiences can be maintained. For detailed information about Pipelines (including Pipeline components and Pipeline sessions), see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

In addition to the services available in the Registration and User Processing package, the BEA WebLogic Commerce Server product also contains services for browsing the product catalog and for order processing. For information on services related to the product catalog, see *BEA WebLogic Commerce Server Product Catalog Management*. For information on services related to order processing, see *BEA WebLogic Commerce Server Order Processing Package*.

High-level Architecture

The Registration and User Processing package is essentially an application that utilizes the Webflow/Pipeline infrastructure. Before you begin to customize or extend this application, however, it is important that you have a high-level understanding of how all the JSP templates in the Registration and User Processing package work together in the default Webflow. It is also important that you understand how this package works in conjunction with JSP templates in the Order Processing package.

- For more information about the default Webflow, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.
- For more information about the Order Processing package, see *BEA WebLogic Commerce Server Order Processing Package*.

Figure 1-2 shows the ways in which your customer might move through the JSP templates in the login and registration portion of the Registration and User Processing package. It also shows where the user processing portion of the package, the Product Catalog, and the Order Processing package come into play.

- Although this diagram shows the shopping cart management piece of the Webflow, it is not discussed in this document. For more information about the shopping cart and the checkout process, see the "Shopping Cart Management Services" in the *BEA WebLogic Commerce Server Order Processing Package* documentation.

Figure 1-2 Default Webflow for Login/Registration

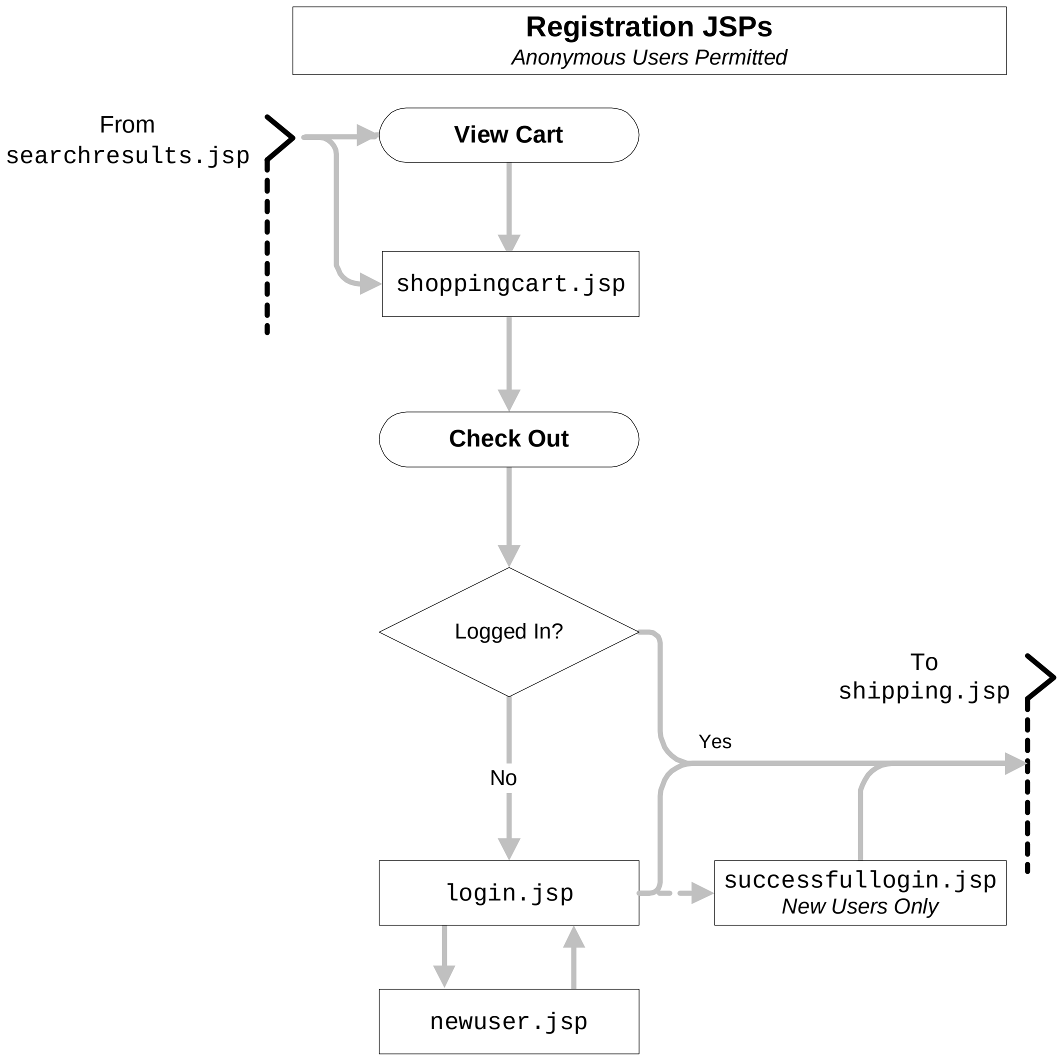
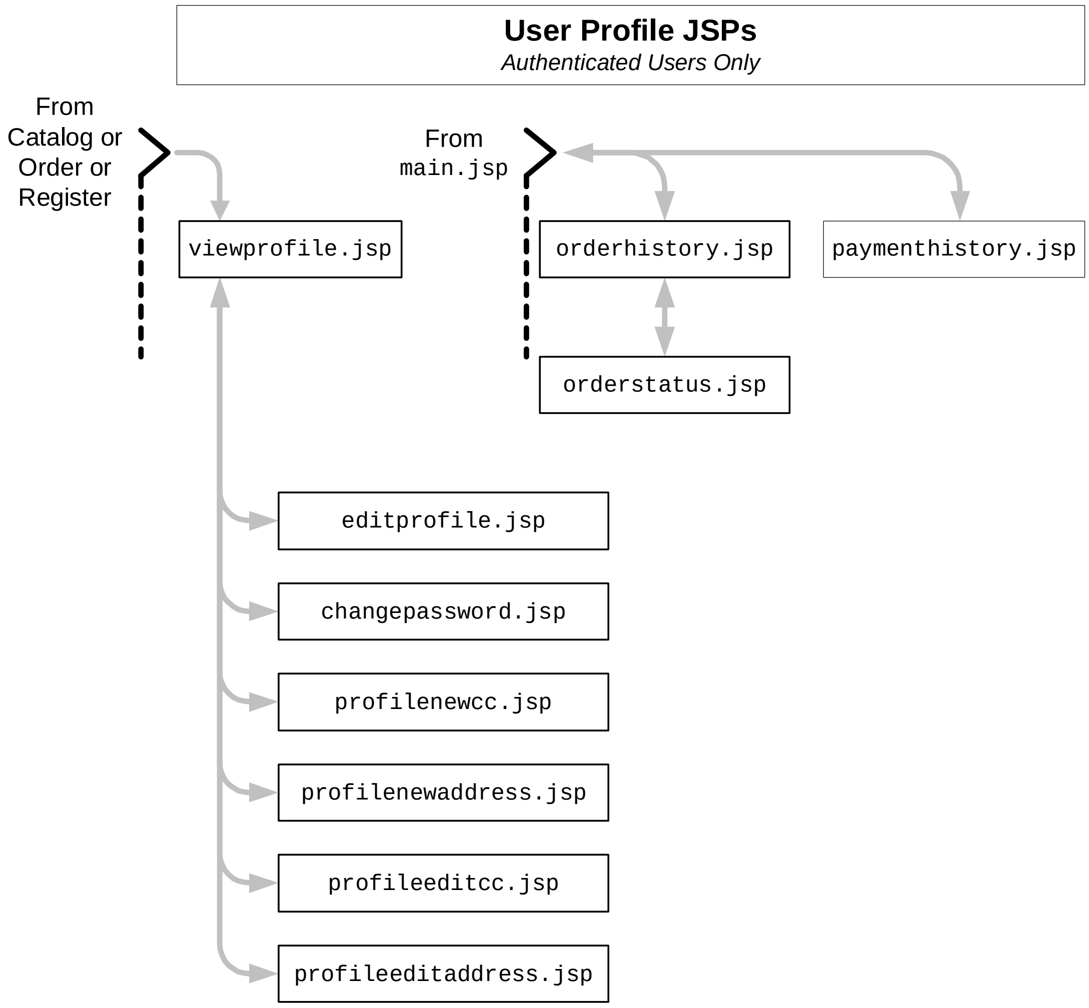


Figure 1-3 shows the ways in which your customer might move through the JSP templates in the user processing and customer self-service portions of the Registration and User Processing package. It also shows where the Product Catalog, the Order Processing package, and the login/registration portion of the package come into play.

Figure 1-3 Default Webflow for User Processing/Customer Self-Service



Note: All JSP templates include other templates, making it easy for you to create new pages with the same look and feel.

Whether you are customizing or extending this architecture, everything you need to know about the services in the Registration and User Processing package (including the JSP templates, input processors, and Pipeline components associated with them) is provided in this document. This includes detailed information about the database schema, for those advanced programmers who want to take their e-business site to the next level.

About the Database Schema

The database schema used for the Registration and User Processing package is the one used for the BEA WebLogic Personalization Server. For more information about this database schema, see *BEA WebLogic Personalization Server Developer's Guide*. Additionally, customer profiles in the BEA WebLogic Commerce Server are implemented as Unified User Profiles (UUP). For more information about UUP, see "Creating and Managing Users" in the *BEA WebLogic Personalization Server* documentation.

Development Roles

This document is intended for the following audiences:

- The commerce engineer/JSP content developer, who uses JSP templates and tag libraries to implement interactive Web pages to meet business requirements. This user also maintains simple configuration files.
- The business analyst, who defines the company's business protocols (processes and rules) for a business-to-consumer Web site. This user may set pricing policies and discounts, and may plan promotional advertising.
- The site administrator, who uses Commerce and Personalization Server administration screens to configure the site's rules, portals, property sets, user profiles, content delivery, and product catalog.
- The Java/EJB programmer, who creates custom code to insert in the JSP files. This user may also handle complex configuration files.

Next Steps

Subsequent chapters of this document describe the Registration and User Processing package in detail, and provide you with information about how to customize or extend the default implementations to meet your requirements. These chapters are as follows:

- "Customer Registration and Login Services"
- "Customer Profile Services"
- "Customer Self-Service"

CHAPTER

2 Customer Registration and Login Services

For customers who plan on frequenting your e-business on a regular basis, it is beneficial to provide a way for them to store some personal information. In doing so, the ordering process will require less time because your customer will not need to reenter their name, address, payment information, and so on. For security, privacy, and management however, this feature requires customers to log into your site with a username/password combination. This topic describes the JavaServer Pages (JSPs) and associated components that allow customers to register and log into your site by creating a customer profile.

This topic includes the following sections:

- JavaServer Pages (JSPs)
  - login.jsp Template
  - badlogin.jsp Template
  - successfullogin.jsp Template
  - newuser.jsp Template
- Input Processors
  - CustomerProfileIP
- Pipeline Components
  - RegisterUserPC

JavaServer Pages (JSPs)

The Registration and User Processing package contains a number of JavaServer Pages (JSPs) that handle customer registration (initial customer profile creation) and customer login. Remember, you can always use these templates for your Web site, or you can adapt them to meet your specific needs. This section describes each of these pages in detail.

Note: For a description of the complete set of JSPs used in the WebLogic Commerce Server Web application and a listing of their locations in the directory structure, see the Summary of JSP Templates documentation.

login.jsp Template

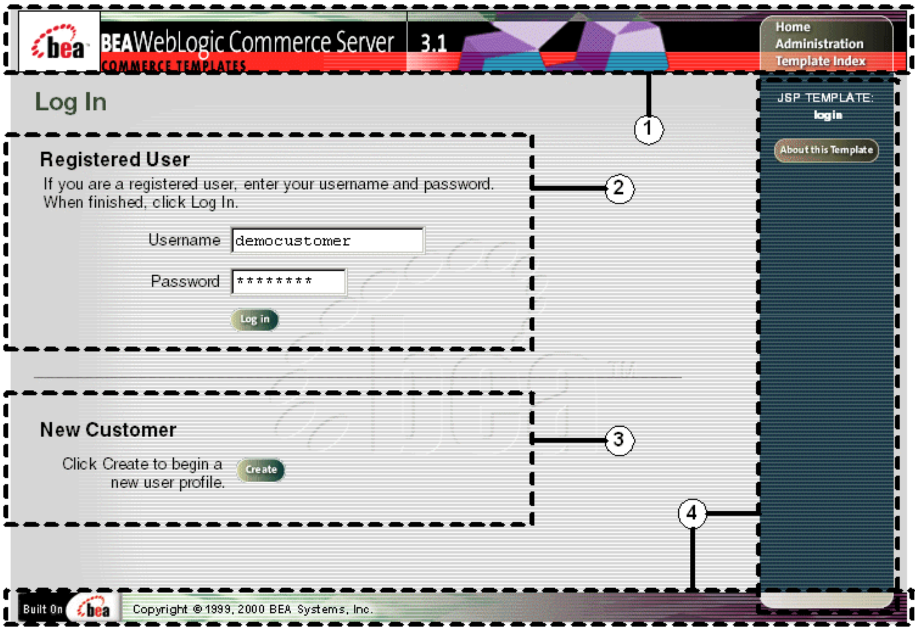
The `login.jsp` template (shown in Figure 2-1) allows a customer who has previously created a profile to log into your e-commerce site by providing a valid username/password combination. Since this page is the entry point to the checkout process, it also establishes mechanisms (such as sessions) that will allow customers to continue their shopping experience.

For customers who have not yet registered with your site, the `login.jsp` template provides customers with an entry point into a page that allows them to register (create their initial customer profile) for subsequent use on the site.

Sample Browser View

Figure 2-1 shows an annotated version of the `login.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 2-1 Annotated login.jsp Template


BEA WebLogic Commerce Server 3.1
COMMERCE TEMPLATES
Home
Administration
Template Index
JSP TEMPLATE:
login
About this Template
Log In
Registered User
If you are a registered user, enter your username and password.
When finished, click Log In.
Username
democustomer
Password
****
Log in
New Customer
Click Create to begin a new user profile.
Create
Built On bea
Copyright © 1999, 2000 BEA Systems, Inc.
1
2
3
4

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. This region provides two form fields for customers who already have a username and password combination. When the form is posted, authentication is handled by the WebLogic Server (not part of the default Webflow).

3. This region provides a link into the page that allows new customers to register with your e-commerce site. Registration involves creating an initial customer profile, and is where the customer will set their username and password for subsequent logins.

4. The `login.jsp` template's content in region 4 contains the included `innerfooter.jsp` template. The include call in `login.jsp` is:

```
<%@ include file="/commerce/includes/innerfooter.jsp" %>
```

`innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

```
<%@ include file="/commerce/includes/rightside.jsp" %>
```

Location in the WebLogic Commerce Server Directory Structure

You can find the `login.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\login.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/login.jsp` (UNIX)

Tag Library Imports

The `login.jsp` template does not use any JSP tags. Therefore, the template does not include imports of any JSP tag libraries.

Java Package Imports

The `login.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import=”javax.servlet.http.*” %>
<%@ page import=”com.beasys.commerce.webflow.*” %>
```

Location in Default Webflow

The `login.jsp` template itself is not part of the default Webflow. Rather, it is automatically loaded into the browser when a protected page is referenced by the WebLogic Server.

Note: All JSP templates in the `/order` and `/user` subdirectories are protected and are accessible only by registered and authenticated customers.

If the customer already has a username/password combination from prior registration and the customer's login is successful, the next page is the protected page the customer was attempting to access. If the customer's login is unsuccessful, a version of the `login.jsp` template is reloaded with an error message (`badlogin.jsp`).

If the customer is not yet registered and clicks on the Create button, the next page loaded will allow the customer to create a profile and obtain a username/password combination (`newuser.jsp`). After the customer has registered, the customer will be returned to `login.jsp` so they can log in. If the customer's login is then successful, the next page is the successful login page, (`successfullogin.jsp`), which allows customers to decide whether they want to proceed to their shopping cart (`shoppingcart.jsp`), proceed to checkout (`shipping.jsp`), or proceed to the main page (`main.jsp`). If the customer's login is unsuccessful, a version of the `login.jsp` template is reloaded with an error message (`badlogin.jsp`).

Notes: The option to proceed to checkout is only provided on the `successfullogin.jsp` template if there are items in the customer's shopping cart.

For a detailed description of the `main.jsp` template, see "Product Catalog JSP Templates and Tag Library" in the *BEA WebLogic Commerce Server Product Catalog Management* documentation. For a detailed description of the `shoppingcart.jsp` and `shipping.jsp` templates, see "Shopping Cart Management Services" or "Shipping Services" in the *BEA WebLogic Commerce Server Order Processing Package* documentation.

For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `login.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

The `login.jsp` template presents a customer with two buttons, only one of which is considered an event. The event triggers a particular response in the default Webflow that allows customers to continue. The other button is a standard HTML Submit button that posts the page back to the WebLogic Server for authentication. Table 2-1 provides information about the event and the business logic it invokes.

Table 2-1 login.jsp Events

Event	Webflow Response(s)
`button(createUser)`	No business logic required. Loads `newuser.jsp`.

Note: The Login button actually is not an event that would trigger a Webflow response. Rather, when a customer clicks the button, control is turned over to the WebLogic Server (specifically, the RDBMS realm of the WebLogic Personalization Server). The WebLogic Server remembers the HTTP request, determines whether the customer's username and password combination is correct, and then reinvokes the Webflow using the request. Since this authentication follows the WebLogic Server and J2EE specifications, more information on this topic can be found in documents at the *BEA WebLogic Server 5.1 Documentation Center*.

Dynamic Data Display

No dynamic data is presented on the `login.jsp` template.

Form Field Specification

The primary purpose of the `login.jsp` template is to allow customers to enter their username and password using two HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `login.jsp` template, and a description for each of these form fields are listed in Table 2-2.

Table 2-2 login.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`login.jsp`), used by the Webflow.
`“j_username”`	Textbox	The customer’s login name, passed to WebLogic Server for authentication.
`“j_password”`	Password	The customer’s login password, passed to WebLogic Server for authentication.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.USER_NAME %>`) for use in the JSP.

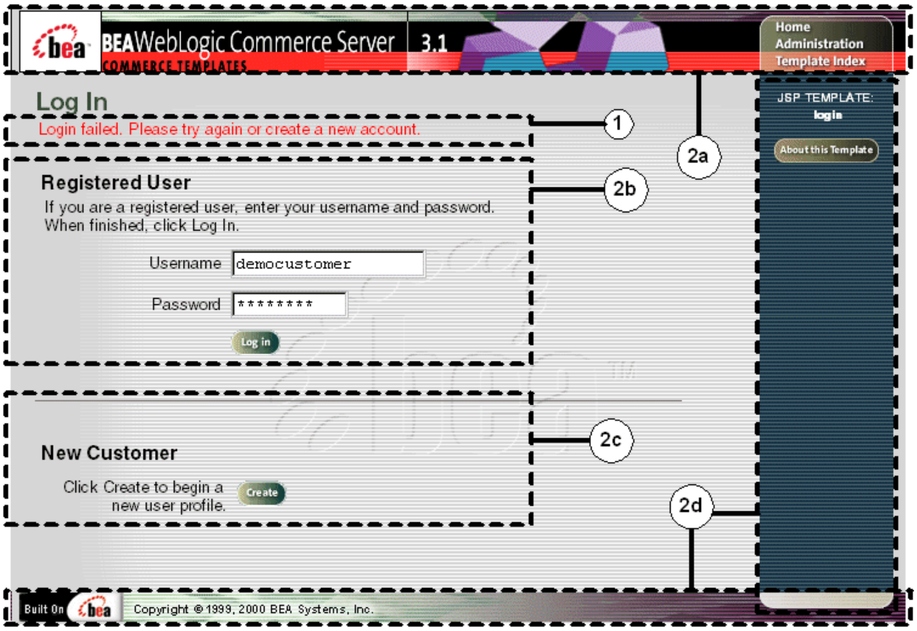
badlogin.jsp Template

The `badlogin.jsp` template (shown in Figure 2-2) informs a customer that they have entered an invalid username/password combination, and allows the customer to try logging into your e-commerce site again by providing a valid username/password combination. Except for the error message, it behaves exactly as the `login.jsp` template previously described.

Sample Browser View

Figure 2-2 shows an annotated version of the `badlogin.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 2-2 Annotated badlogin.jsp Template



Log In

Login failed. Please try again or create a new account.

Registered User

If you are a registered user, enter your username and password. When finished, click Log In.

Username democustomer

Password ********

New Customer

Click Create to begin a new user profile.

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. Region 1 displays the login error message to the customer, and prompts the customer to try logging in again or to create a new account.
2. Because the `badlogin.jsp` template includes the `login.jsp` template, the following regions are actually part of the `login.jsp` template:

   a. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by the WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

   b. This region provides two form fields for customers who already have a username and password combination. When the form is posted, authentication is handled by WebLogic Server (not part of the default Webflow).

   c. This region provides a link into the page that allows new customers to register with your e-commerce site. Registration involves creating an initial customer profile, and is where the customer will set their username and password for subsequent logins.

   d. The `login.jsp` template's content in region 2d contains the included `innerfooter.jsp` template. The include call in `login.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `badlogin.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\badlogin.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/badlogin.jsp` (UNIX)

Tag Library Imports

The `badlogin.jsp` template does not use any JSP tags. Therefore, the template does not include imports of any JSP tag libraries.

Java Package Imports

The `badlogin.jsp` template does not use any Java classes and therefore does not include any package import statements.

Location in Default Webflow

Customers arrive at the `badlogin.jsp` template when they fail to provide a valid username/password combination on the `login.jsp` template. If the customer is registered and the customer's second attempt at logging in is successful, the next page is the protected page the customer was attempting to access. If the customer's login is unsuccessful, the `badlogin.jsp` template is reloaded.

If the customer is not yet registered and clicks on the Create button, the next page loaded will allow them to create a customer profile and obtain a username/password combination (`newuser.jsp`). After the customer has registered, the customer will be returned to `login.jsp` so they can log in.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP template is included in the `badlogin.jsp` template:

- `login.jsp`, which creates the entire page, with the exception of the error message at the top.

Events

Because the `badlogin.jsp` template is essentially the same as the `login.jsp` template, the `badlogin.jsp` template makes use of the same events. For more information about these events, see "login.jsp Template" on page 2-2.

Dynamic Data Display

No dynamic data is presented on the `badlogin.jsp` template.

Form Field Specification

Because the `badlogin.jsp` template is essentially the same as the `login.jsp` template, the `badlogin.jsp` template makes use of the same form fields. For more information about these form fields, see “login.jsp Template” on page 2-2.

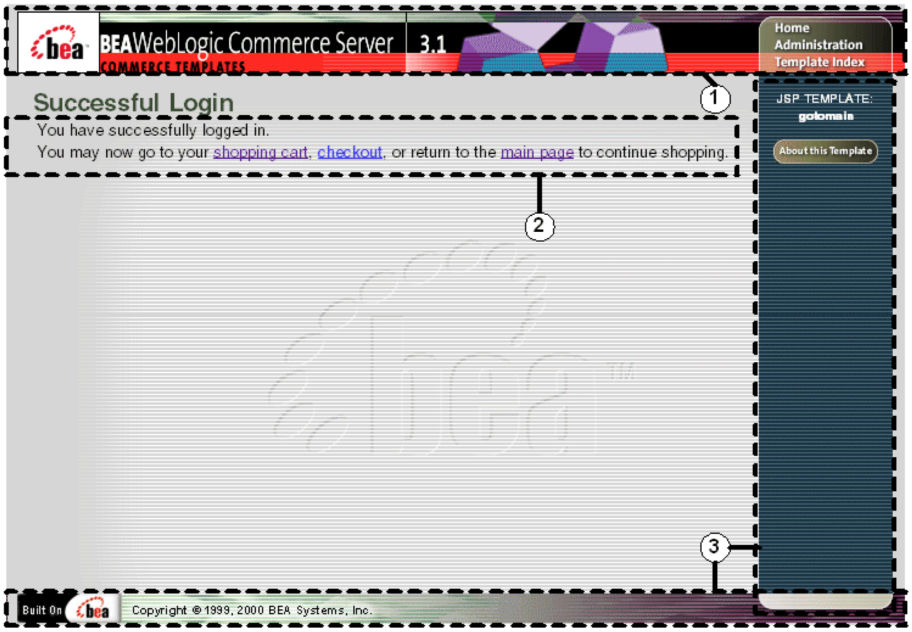
successfullogin.jsp Template

The `successfullogin.jsp` template (shown in Figure 2-3) informs a customer who has just created a user profile and logged in that the login was successful, and provides the customer with the opportunity to return to their shopping experience through several navigation options.

Sample Browser View

Figure 2-3 shows an annotated version of the `successfullogin.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 2-3 Annotated successfullogin.jsp Template - With Checkout Option



Successful Login

You have successfully logged in.

You may now go to your shopping cart, checkout, or return to the main page to continue shopping.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

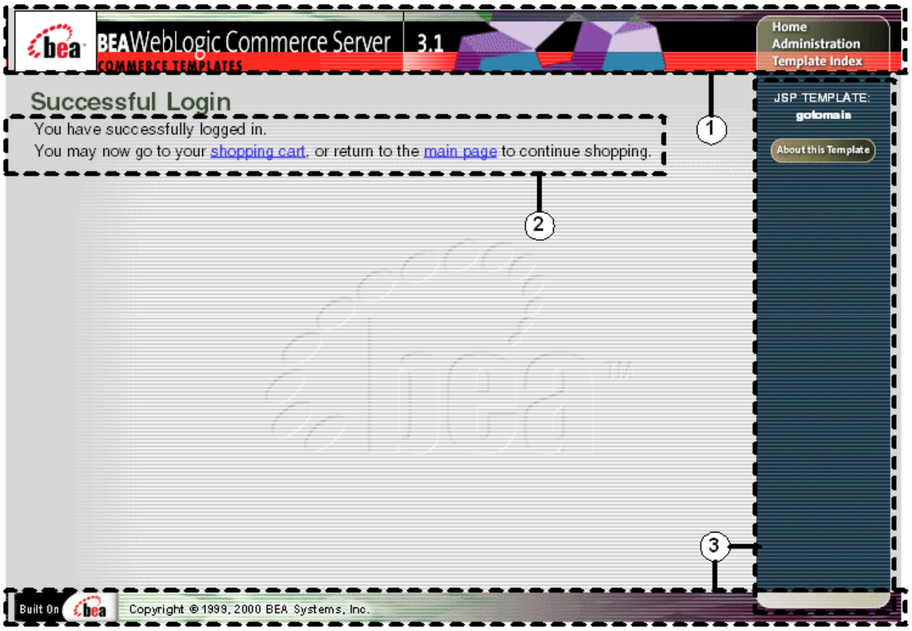
   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. This region indicates to the customer that their login (and thus registration) was successful, and provides them with links to return to their shopping cart (`shoppingcart.jsp`), to continue to the checkout process (`shipping.jsp`) or to return to the main catalog page (`main.jsp`).

Notes: For a detailed description of the `main.jsp` template, see the "Product Catalog JSP Templates and Tag Library" in the *BEA WebLogic Commerce Server Product Catalog Management* documentation. For a detailed description of the `shoppingcart.jsp` and `shipping.jsp` templates, see the "Shopping Cart Management Services" or the "Shipping Services" in the *BEA WebLogic Commerce Server Order Processing Package* documentation.

The option to proceed to checkout is only provided on the `successfullogin.jsp` template if there are items in the customer's shopping cart. Otherwise, the `successfullogin.jsp` template will leave out this option, as shown in Figure 2-4.

Figure 2-4 successfullogin.jsp - Without Checkout Option



Successful Login

You have successfully logged in.
You may now go to your shopping cart, or return to the main page to continue shopping.

3. The `successfullogin.jsp` template's content in region 3 contains the included `innerfooter.jsp` template. The include call in `successfullogin.jsp` is:

```
<%@ include file="/commerce/includes/innerfooter.jsp" %>
```

`innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

```
<%@ include file="/commerce/includes/rightside.jsp" %>
```

Location in the WebLogic Commerce Server Directory Structure

You can find the `successfullogin.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\successfullogin.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/successfullogin.jsp` (UNIX)

Tag Library Imports

The `successfullogin.jsp` template uses Pipeline JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri="pipeline.tld" prefix="pipeline" %>
```

Note: For more information about the Pipeline JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `successfullogin.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.ebusiness.shoppingcart.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
```

Location in Default Webflow

Customers arrive at the `successfullogin.jsp` template when they have successfully logged into your e-commerce site (on the `login.jsp` template) only after just having created a customer profile.

Note: If a customer had created a profile on a previous visit and logged in using the `login.jsp` template, the customer would simply be taken to the protected page the customer was trying to access.

From the `successfullogin.jsp` template, the customer can return to their shopping cart (`shoppingcart.jsp`), continue to the checkout process (`shipping.jsp`), or return to the main catalog page (`main.jsp`).

Notes: The option to proceed to checkout is only provided on the `successfullogin.jsp` template if there are items in the customer's shopping cart.

For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `successfullogin.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

Every time a customer clicks a button to view more detail about an order, it is considered an event. Each event triggers a particular response in the default Webflow that allows them to continue. While this response can be to load another JSP, it is usually the case that an input processor and/or Pipeline is invoked first. Table 2-3 provides information about these events and the business logic they invoke.

Table 2-3 successfullogin.jsp Events

Event	Webflow Response(s)
`link(shoppingcart)`	`InitShoppingCartIP`
`link(checkout)`	`InitShippingMethodListIP`

Table 2-3 successfullogin.jsp Events

Event	Webflow Response(s)
`link(home)`	`GetTopCategoriesIP` `GetTopCategories`

Note: For more information about the `GetTopCategoriesIP` and `GetTopCategories` Pipeline, see *BEA WebLogic Commerce Server Product Catalog Management*.

Dynamic Data Display

One purpose of the `successfullogin.jsp` template is to display navigation options that allow customers to continue their shopping experience after logging in. However, if there are no items in the customer's shopping cart, then checkout is not an option that should be displayed. The decision of whether or not to display this option is accomplished on `successfullogin.jsp` using a combination of Pipeline JSP tags and accessor methods/attributes.

First, the `getPipelineProperty` JSP tag retrieves the `SHOPPING_CART` attribute from the Pipeline session. Table 2-4 provides more detailed information on this attribute.

Table 2-4 successfullogin.jsp Pipeline Session Attributes

Attribute	Type	Description
`PipelineSessionConstants.SHOPPING_CART`	`com.beasys.commerce.ebusiness.shoppingcart.ShoppingCart`	The currently active shopping cart.

Listing 2-1 illustrates how this attribute is retrieved from the Pipeline session using the `getPipelineProperty` JSP tag.

Listing 2-1 Retrieving the Shopping Cart Attribute

```
<pipeline:getPipelineProperty
 propertyName="<%=PipelineSessionConstants.SHOPPING_CART%>"
 returnName="shoppingCart"
 returnType="com.beasys.commerce.ebusiness.shoppingcart.
ShoppingCart"/>
```

Note: For more information on the `getPipelineProperty` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

The data stored within the Pipeline session attribute is then accessed by using accessor methods/attributes within Java scriptlets. Table 2-5 provides more detailed information about these methods/attributes for `shoppingCart`.

Table 2-5 shoppingCart Accessor Methods/Attributes

Method/Attribute	Description
`isEmpty()`	Returns `true` if the customer's shopping cart is empty.

The presence of items in the shopping cart is evaluated using this method in a Java scriptlet, as shown in Listing 2-2.

Listing 2-2 Using Accessor Methods/Attributes Within successfullogin.jsp Java Scriptlets

```
<% if (shoppingCart != null && shoppingCart.isEmpty() == false) { %>
<a href="<%=WebflowJSPHelper.createWebflowURL(pageContext,
"successfullogin.jsp","button(checkout)", true)%>">checkout</a>
<% } %>
```

Form Field Specification

No form fields are used in the `successfullogin.jsp` template.

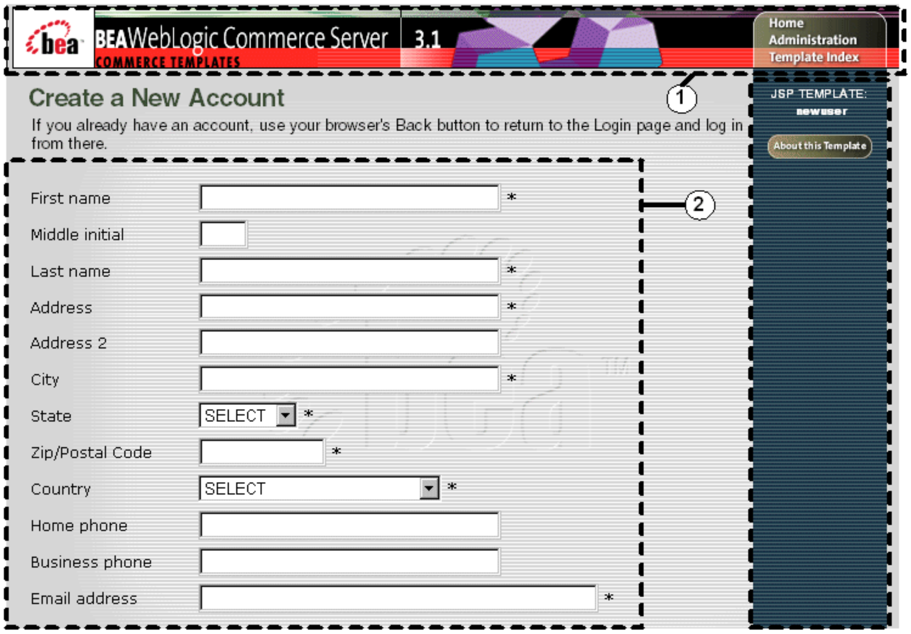
newuser.jsp Template

The `newuser.jsp` template (shown in Figure 2-5 through Figure 2-7) allows a new customer to register with your e-commerce site by creating their customer profile, which includes personal information, shipping address information, payment information (optional), and account information.

Sample Browser View

Figure 2-5 through Figure 2-7 show annotated versions of the `newuser.jsp` template. Although there are three figures, together these screen shots form the single `newuser.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shots.

Figure 2-5 Annotated newuser.jsp Template - Personal Information



Create a New Account

If you already have an account, use your browser's Back button to return to the Login page and log in from there.

First name	*
Middle initial	
Last name	*
Address	*
Address 2	
City	*
State	SELECT *
Zip/Postal Code	*
Country	SELECT *
Home phone	
Business phone	
Email address	*

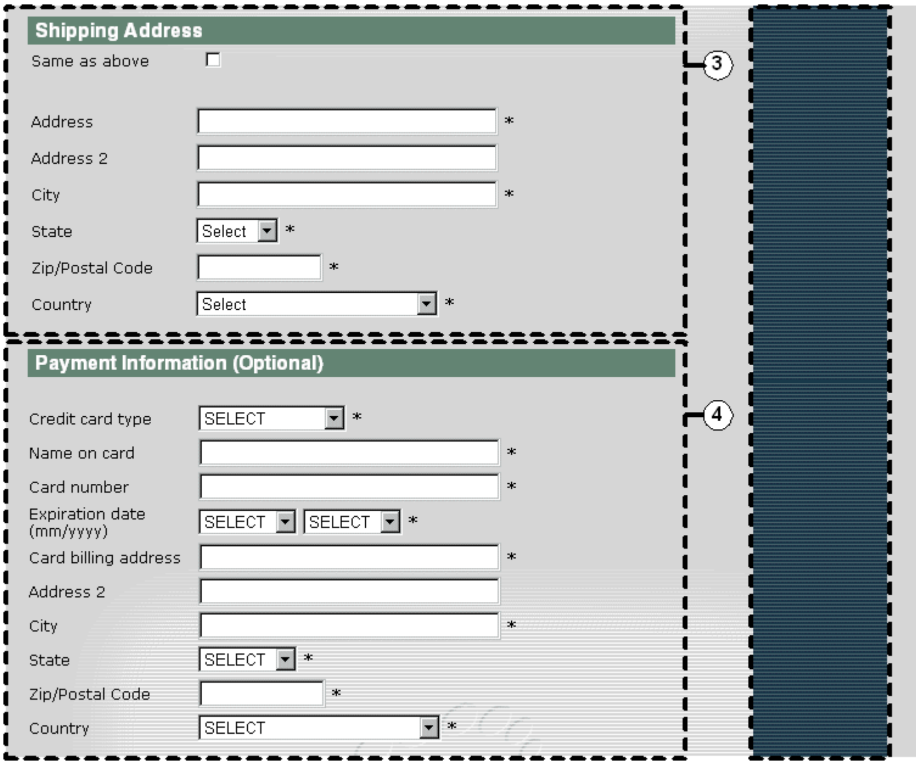
The numbers in the following list refer to the numbered regions in the figures:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. This region provides form fields for customers to enter their personal customer profile information, including their name, address, phone number(s), and email address. For the address, this region utilizes the form fields defined in the included `states.jsp` and `countries.jsp` template files.

Figure 2-6 Annotated newuser.jsp Template - Shipping Address and Payment Information (Optional)



Shipping Address

Same as above	☐
Address	*
Address 2	
City	*
State	Select *
Zip/Postal Code	*
Country	Select *

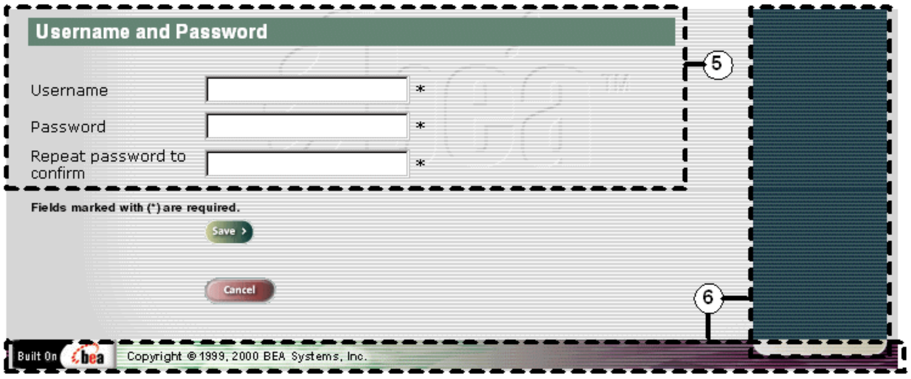
Payment Information (Optional)

Credit card type	SELECT *
Name on card	*
Card number	*
Expiration date (mm/yyyy)	SELECT SELECT *
Card billing address	*
Address 2	
City	*
State	SELECT *
Zip/Postal Code	*
Country	SELECT *

3. This region provides form fields for customers to enter a shipping address. If the customer wishes to use the address they provided in the personal information section as their shipping address, the customer can click the Same as Above checkbox instead of retyping the information. Other than the checkbox, this region consists almost entirely of the form fields defined in the included `newaddresstemplate.jsp` template file.

4. If the customer would like to provide their payment information, region 4 allows the customer to do so. This region provides form fields for the type of credit card, the credit card holder's name, the credit card number, the credit card expiration date, and an address associated with the credit card. The information requested in this region is optional, and consists entirely of the form fields defined in the included `newcctemplate.jsp` template file.

Figure 2-7 Annotated newuser.jsp Template - Account Information


Username and Password
Username *
Password *
Repeat password to confirm *
Fields marked with (*) are required.
Save
Cancel
5
6
Built On bea
Copyright © 1999, 2000 BEA Systems, Inc.

5. This region provides the customer with the opportunity to specify their username and password for use on your e-business site. If the customer decides to submit this form data, the customer's profile will be saved and the `login.jsp` template will be reloaded to allow the customer to login.

Note: The maximum number of characters allowed for usernames and passwords is set to 50, but there are no other restrictions. If you want to impose other restrictions (such as required character types, disallowed character types, or length requirements), you must set up your own field validator.

6. The `newuser.jsp` template's content in region 6 contains the included `innerfooter.jsp` template. The include call in `newuser.jsp` is:

```
<%@ include file="/commerce/includes/innerfooter.jsp" %>
```

`innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

```
<%@ include file="/commerce/includes/rightside.jsp" %>
```

Location in the WebLogic Commerce Server Directory Structure

You can find the `newuser.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\registration\newuser.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/registration/newuser.jsp` (UNIX)

Tag Library Imports

The `newuser.jsp` template makes use of the Webflow JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri="weblogic.tld" prefix="webflow" %>
```

Note: For more information about the Webflow JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `newuser.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
```

Location in Default Webflow

The page prior to `newuser.jsp` is the customer login page (`login.jsp`). If no errors are found after a customer enters their initial profile information, customers are returned to the customer login page (`login.jsp`) where they can use their account information to log in. If errors are found, the `newuser.jsp` is reloaded with an appropriate message next to the erroneous form fields.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `newuser.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `states.jsp`, which contains a list of states that are displayed when the customer is prompted to enter an address.
- `countries.jsp`, which contains a list of countries that are displayed when the customer is prompted to enter an address.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

About the Included newaddresstemplate.jsp Template

The `newaddresstemplate.jsp` template (included in most JSP templates that prompt customers for a shipping address) provides a standardized format for both the form field presentation and error handling. The form fields are organized in a table, and upon form submission, the input processors associated with the `newaddresstemplate.jsp` template will validate the form to ensure that all required fields contain values. If errors are detected, the `newaddresstemplate.jsp` template will be redisplayed, with an error message at the top and the offending field labels shown in red (as opposed to the original black) font. Further, the information your customer entered correctly will still be displayed in the form.

The behavior described above is accomplished on the `newaddresstemplate.jsp` template using the `getValidatedValue` JSP tag, as shown in Listing 2-3.

Listing 2-3 Use of the getValidatedValue JSP Tag on newaddresstemplate.jsp

```
<table>
<tr>

<!-- use the webflow:getValidatedValue to retrieve a value from the HttpRequest.
This value was placed there by the CustomerProfileIP input processor -->

  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1%>"
     fieldValue="customerShippingAddress1" fieldStatus="status"
     validColor="black" invalidColor="red" unspecifiedColor="black"
     fieldColor="fontColor" />

    <div class="tabletext">
      <font color=<%=fontColor%>>Address</font>
    </div>
  </td>

  <td>
    <input type="text"
     name="<%=HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1%>"
     value="<%=customerShippingAddress1%>" size="30" maxlength="30">
  </td>

</tr>
</table>
```

Notes: For more information about the `getValidatedValue` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

Because the `newaddresstemplate.jsp` template collects address information, this template also includes `states.jsp` and `countries.jsp` where appropriate.

About the Included newcctemplate.jsp Template

The `newcctemplate.jsp` template (included in all JSP templates that prompt customers for credit card/payment information) provides a standardized format for both the form presentation and error handling. The form fields are organized in a table, and upon form submission, the input processors associated with the `newcctemplate.jsp` template will validate the form to ensure that all required fields contain values. If errors are detected, the `newcctemplate.jsp` template will be redisplayed, with an error message at the top and the offending field labels shown in red (as opposed to the original black) font. Further, the information your customer entered correctly will still be displayed in the form.

The behavior described above is accomplished on the `newcctemplate.jsp` template using the `getValidatedValue` JSP tag, as shown in Listing 2-4.

Listing 2-4 Use of the getValidatedValue JSP Tag on newcctemplate.jsp

```
<table>
<tr>

<!-- use the webflow:getValidatedValue to retrieve a value from the HttpRequest.
This value was placed there by the CustomerProfileIP input processor -->

  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER%>"
     fieldValue="customerCreditCardHolder" fieldStatus="status"
     validColor="black" invalidColor="red" unspecifiedColor="black"
     fieldColor="fontColor" />

    <div class="tabletext">
      <font color=<%=fontColor%>>Name on card</font>
    </div>

  </td>

  <td>
    <input type="text"
     name="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER%>"
     value="<%=customerCreditCardHolder%>" size="30">
  </td>

</tr>
</table>
```

Notes: For more information about the `getValidatedValue` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

Because the `newcctemplate.jsp` template collects address information, this template also includes `states.jsp` and `countries.jsp` where appropriate.

Events

The `newuser.jsp` template presents a customer with two buttons, each of which is considered an event. These events trigger a particular response in the default Webflow that allows customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 2-6 provides information about these events and the business logic they invoke.

Table 2-6 newuser.jsp Events

Event	Webflow Response(s)
`button(cancel)`	`GetCategoryIP`
`button(save)`	`CustomerProfileIP` `CustomerProfile`

Table 2-7 briefly describes each of the Pipelines from Table 2-6, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see "Pipeline Components" on page 2-33.

Table 2-7 New User Profile Pipelines

Pipeline	Description
`CustomerProfile`	Contains `EncryptCreditCardPC` and `RegisterUserPC`, and is transactional.

Dynamic Data Display

No dynamic data is presented on the `newuser.jsp` template.

Form Field Specification

The primary purpose of the `newuser.jsp` template is to allow customers to enter their profile information using various HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `newuser.jsp` template, and a description for each of these form fields are listed in Table 2-8.

Table 2-8 newuser.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`newuser.jsp`), used by the Webflow.
`HttpRequestConstants.CUSTOMER_FIRST_NAME`	Textbox	The customer’s first name.
`HttpRequestConstants.CUSTOMER_MIDDLE_NAME`	Textbox	The customer’s middle initial.
`HttpRequestConstants.CUSTOMER_LAST_NAME`	Textbox	The customer’s last name.
`HttpRequestConstants.CUSTOMER_ADDRESS1`	Textbox	The first line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_ADDRESS2`	Textbox	The second line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_CITY`	Textbox	The city in the customer’s address.
`HttpRequestConstants.CUSTOMER_STATE`	Listbox	The state in the customer’s address.
`HttpRequestConstants.CUSTOMER_ZIPCODE`	Textbox	The zip code in the customer’s address.

Table 2-8 newuser.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.` `CUSTOMER_COUNTRY`	Listbox	The country in the customer's address.
`HttpRequestConstants.` `CUSTOMER_HOME_PHONE`	Textbox	The customer's home phone number.
`HttpRequestConstants.` `CUSTOMER_BUSINESS_PHONE`	Textbox	The customer's business phone number.
`HttpRequestConstants.` `CUSTOMER_EMAIL`	Textbox	The customer's email address.
`HttpRequestConstants.` `SAME_AS_ABOVE`	Checkbox	Indicates that the customer's shipping address is the same as the contact address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_ADDRESS1`	Textbox	The first line in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_ADDRESS2`	Textbox	The second line in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_CITY`	Textbox	The city in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_STATE`	Listbox	The state in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_ZIPCODE`	Textbox	The zip/postal code in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_SHIPPING_COUNTRY`	Listbox	The country in the customer's shipping address.
`HttpRequestConstants.` `CUSTOMER_CREDITCARD_TYPE`	Listbox	The type of the customer's credit card.
`HttpRequestConstants.` `CUSTOMER_CREDITCARD_HOLDER`	Textbox	The name on the credit card.
`HttpRequestConstants.` `CUSTOMER_CREDITCARD_NUMBER`	Textbox	The number of the customer's credit card.

Table 2-8 newuser.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH`	Listbox	The month of the customer’s credit card expiration date.
`HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR`	Listbox	The year of the customer’s credit card expiration date.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1`	Textbox	The first line in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2`	Textbox	The second line in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_CITY`	Textbox	The city in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_STATE`	Listbox	The state in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE`	Textbox	The zip/postal code in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY`	Listbox	The country in the customer’s billing address.
`HttpRequestConstants.USER_NAME`	Textbox	An identity chosen by the customer for login.
`HttpRequestConstants.PASSWORD`	Password	A password chosen by the customer for login.
`HttpRequestConstants.CONFIRM_PASSWORD`	Password	Confirmation of the password chosen by the customer for login.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.USER_NAME %>`) for use in the JSP.

Input Processors

This section provides a brief description of each input processor associated with the Customer Login and Registration Services JSP template(s).

Note: For more information about the `GetTopCategoriesIP` input processor, see *BEA WebLogic Commerce Server Product Catalog Management*.

CustomerProfileIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.CustomerProfileIP`
Description	Processes the input of `newuser.jsp` and allows the customer to store their profile. Creates and places a `CustomerValue` object into the Pipeline session.

Required `HTTPServletRequest` Parameters (Personal Information)	`HttpRequestConstants.CUSTOMER_FIRST_NAME`
	`HttpRequestConstants.CUSTOMER_MIDDLE_NAME`
	`HttpRequestConstants.CUSTOMER_LAST_NAME`
	`HttpRequestConstants.CUSTOMER_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_CITY`
	`HttpRequestConstants.CUSTOMER_STATE`
	`HttpRequestConstants.CUSTOMER_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_COUNTRY`
	`HttpRequestConstants.CUSTOMER_HOME_PHONE`
	`HttpRequestConstants.CUSTOMER_BUSINESS_PHONE`
	`HttpRequestConstants.CUSTOMER_EMAIL`

Required `HTTPServletRequest` Parameters (Shipping Information)	`HttpRequestConstants.SAME_AS_ABOVE`
	`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_SHIPPING_CITY`
	`HttpRequestConstants.CUSTOMER_SHIPPING_STATE`
	`HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY`
	`HttpRequestConstants.DEFAULT_SHIPPING_ADDRESS`

`HTTPServletRequest` Parameters (Payment Information)	`HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_CITY`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_STATE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY`

Required `HTTPServletRequest` Parameters (Account Information)	`HttpRequestConstants.USER_NAME`
	`HttpRequestConstants.PASSWORD`
	`HttpRequestConstants.CONFIRM_PASSWORD`

Required Pipeline Session Attributes	None

Updated Pipeline Sesion Attributes	`PipelineSessionConstants.CUSTOMER`
	`PipelineSessionConstants.PASSWORD`
	`PipelineSessionConstants.CREDIT_CARD_KEY` (only if customer provides a credit card update).

Removed Pipeline Session Attributes	None
Validation	Checks that the required fields contain values and checks that the credit card number is not less than 16 digits (15 digits for AMEX type). Also checks that the password and confirm password fields contain matching values.

Exceptions	`InvalidInputException`, thrown when required fields are empty or credit card number is less than 16 digits (15 digits for AMEX type).

Pipeline Components

This section provides a brief description of each Pipeline component associated with the Customer Login and Registration Services JSP template(s).

Note: Some Pipeline components extend other, base Pipeline components. For more information on the base classes, see the *Javadoc*.

RegisterUserPC

Class Name	`com.beasys.commerce.ebusiness.customer.pipeline.RegisterUserPC`
Description	Retrieves the `CustomerValue` object and password from the Pipeline session, and creates a `CUSTOMER` attribute.
Required Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER` `PipelineSessionConstants.PASSWORD`
Updated Pipeline Session Attributes	None
Removed Pipeline Session Attributes	`PipelineSessionConstants.PASSWORD`
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when the Pipeline component cannot create the user.

CHAPTER

3 Customer Profile Services

Customers who have registered with your e-commerce site may, from time to time, change the information stored in their profile. For example, customers may want to send a shipment to a different address, or use a different credit card. To help you meet your customers' needs, the Registration and User Processing Package provides you with an implementation of these Customer Profile Services. This topic describes the pages that allow registered customers to modify various aspects of their customer profile.

This topic includes the following sections:

- JavaServer Pages (JSPs)
  - viewprofile.jsp Template
  - editprofile.jsp Template
  - profilenewaddress.jsp Template
  - profileeditaddress.jsp Template
  - profilenewcc.jsp Template
  - profileeditcc.jsp Template
  - changepassword.jsp Template
- Input Processors
  - DeleteCreditCardIP
  - DeleteShippingAddressIP
  - UpdateAccountInfoIP

- UpdateBasicInfoIP
- UpdatePaymentInfoIP
- UpdateShippingInfoIP

- Pipeline Components
  - UpdateBasicInfoPC
  - UpdatePaymentInfoPC
  - UpdateShippingInfoPC
  - UpdatePasswordPC

JavaServer Pages (JSPs)

The Registration and User Processing package contains a number of JavaServer Pages (JSPs) that allow customers to view or update their stored profile. Remember, you can always use these templates for your Web site, or you can adapt them to meet your specific needs. This section describes each of these pages in detail.

Note: For a description of the complete set of JSPs used in the WebLogic Commerce Server Web application and a listing of their locations in the directory structure, see the Summary of JSP Templates documentation.

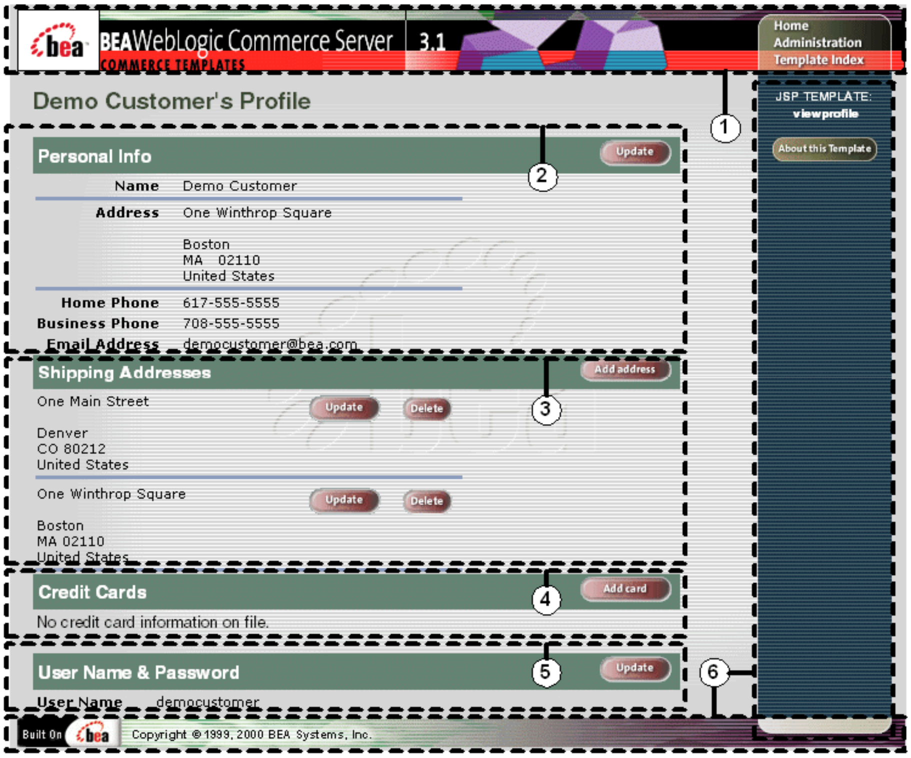
viewprofile.jsp Template

The `viewprofile.jsp` template (shown in Figure 3-1) allows a registered customer to view their existing profile information. It displays the existing information in four categories: personal information, shipping addresses, credit cards, and username and password. There are options in each category for updating, deleting, or adding information.

Sample Browser View

Figure 3-1 shows an annotated version of the `viewprofile.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-1 Annotated viewprofile.jsp Template



Demo Customer's Profile

Personal Info

Name	Demo Customer
Address	One Winthrop Square Boston MA 02110 United States
Home Phone	617-555-5555
Business Phone	708-555-5555
Email Address	democustomer@bea.com

Shipping Addresses

One Main Street

Denver
CO 80212
United States

One Winthrop Square

Boston
MA 02110
United States

Credit Cards

No credit card information on file.

User Name & Password

User Name democustomer

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. Region 2 displays the customer's existing personal information using the WebLogic Server JSP tags and the WebLogic Personalization Server's User Management JSP tags. It also provides customers with a button that will allow customers to update their existing personal information (using the `editprofile.jsp` template).

3. This region displays any shipping addresses the customer may have previously stored as part of their customer profile. This is accomplished using the WebLogic Server JSP tags and the WebLogic Personalization Server's User Management JSP tags. In this region, your customer can choose from buttons that allow them to delete an address, enter a new shipping address (using the `profilenewaddress.jsp` tempate), or update an existing address (using the `profileeditaddress.jsp` template).

4. If any exists, region 4 displays the customer's existing credit card (payment) information using a combination of the WebLogic Server JSP tags and the WebLogic Personalization Server's User Management JSP tags. For each credit card shown, your customer can decide to delete the card, enter a new credit card (using the `profilenewcc.jsp` template), or change the information associated with the card (using the `profileeditcc.jsp` template).

5. This region displays the customer's current username, and provides customers with a button that will allow them to modify their password (using the `changepassword.jsp` template).

6. The `viewprofile.jsp` template's content in region 6 contains the included `innerfooter.jsp` template. The include call in `viewprofile.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `viewprofile.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\viewprofile.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/viewprofile.jsp` (UNIX)

Tag Library Imports

The `viewprofile.jsp` template uses existing WebLogic Server JSP tags and WebLogic Personalization Server's User Management JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri="weblogic.tld" prefix="wl" %>
<%@ taglib uri="um.tld" prefix="um" %>
```

Note: For more information on the WebLogic Server JSP tags or the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `viewprofile.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ page import="com.beasys.commerce.webflow.PipelineSessionConstants" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
```

Location in Default Webflow

If the customer is not logged in, the page prior to the `viewprofile.jsp` template is the customer login page (`login.jsp`). If the customer is already logged in, the page prior to the `viewprofile.jsp` template is any page from which the customer clicks the View Profile button. Based on what the customer decides to do after viewing their profile, the next page could be any of the following:

- `editprofile.jsp`, which allows customers to edit their personal information, including their name, contact address, and phone numbers,
- `profilenewaddress.jsp`, which allows customers to add a new shipping address,
- `profileeditaddress.jsp`, which allows customers to edit a shipping address,
- `profilenewcc.jsp`, which allows customers to add a new credit card to the profile,
- `profileeditcc.jsp`, which allows customers to edit information about an existing credit card, or
- `changepassword.jsp`, which allows customers to change their account password.

Each of these pages are described in subsequent sections of this document.

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `viewprofile.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

The `viewprofile.jsp` template presents a customer with several buttons, each of which is considered an event. These events trigger a particular response in the default Webflow that allow customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-1 provides information about these events and the business logic they invoke.

Table 3-1 viewprofile.jsp Events

Event	Web Flow Response(s)
`button(updateBasicInfo)`	No business logic required. Loads `editprofile.jsp.`
`button(addNewShippingAddress)`	No business logic required. Loads `profilenewaddress.jsp.`
`button(updateShippingInfo)`	No business logic required. Loads `profileeditaddress.jsp.`
`button(deleteShippingAddress)`	`DeleteShippingAddressIP` `DeleteShippingAddressFromProfile`
`button(addNewCreditCard)`	No business logic required. Loads `profilenewcc.jsp.`
`button(updatePaymentInfo)`	No business logic required. Loads `profileeditcc.jsp.`
`button(deletePaymentInfo)`	`DeleteCreditCardIP` `DeleteCreditCard`
`button(changePassword)`	No business logic required. Loads `changepassword.jsp.`

Table 3-2 briefly describes each of the Pipelines from Table 3-1, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see “Pipeline Components” on page 3-65.

Table 3-2 View Profile Pipelines

Pipeline	Description
`DeleteShippingAddressFromProfile`	Contains `UpdateShippingInfoPC` and is transactional.
`DeleteCreditCard`	Contains `UpdatePaymentInfoPC` and is transactional.

Dynamic Data Display

One purpose of the `viewprofile.jsp` template is to display the profile information a customer had previously entered. This is accomplished on `viewprofile.jsp` using a combination of WebLogic Server JSP tags, the WebLogic Personalization Server's User Management JSP tags, and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) for which the customer information should be retrieved, as shown in Listing 3-1.

Listing 3-1 Setting the Customer Context

```
<um:getProfile profileKey="<%=request.getRemoteUser()%>"
 profileType="WLCS_Customer" />
```

Note: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Next, the `getProperty` JSP tag is used to obtain the customer's contact address, a collection of the customer's shipping addresses, and a collection of the customer's credit cards, which are then initialized with data from their corresponding objects. This is shown in Listing 3-2.

Listing 3-2 Obtaining the Customer's Profile Information

```
<um:getProperty propertyName="contactAddress"
 id="contactAddressObject" />
```

```
<um:getProperty propertyName="shippingAddressMap"
 id="shippingAddressMapObject" />

<um:getProperty propertyName="creditCardsMap"
 id="creditCardsMapObject" />

Address contactAddress = (Address) contactAddressObject;
Map shippingAddressMap = (Map) shippingAddressMapObject;
Map creditCardsMap = (Map) creditCardsMapObject;
```

The data stored within these objects can now be accessed by calling accessor methods/attributes within Java scriptlets. Table 3-3 provides more detailed information about the methods/attributes for both the contact and shipping addresses. Table 3-4 provides information about the methods/attributes for the customer's credit cards.

Table 3-3 contactAddress/shippingAddress Accessor Methods/Attributes

Method/Attribute	Description
`getStreet1()`	The first line in the customer's contact or shipping street address.
`getStreet2()`	The second line in the customer's contact or shipping street address.
`getCity()`	The city in the customer's contact or shipping address.
`getCounty()`	The county in the customer's contact or shipping address.
`getState()`	The state in the customer's contact or shipping address.
`getPostalCode()`	The zip/postal code in the customer's contact or shipping address.
`getCountry()`	The country in the customer's contact or shipping address.

Table 3-4 creditCard Accessor Methods/Attributes

Method/Attribute	Description
`creditCard`	The credit card name, consisting of the credit card type and 4 digits (for example, VISA-4111).

Listing 3-3 illustrates how these accessor methods/attributes are used within Java scriptlets.

Listing 3-3 Using Accessor Methods/Attributes Within viewprofile.jsp Java Scriptlets

```
<table>
.
.
.
<tr>
  <td><div class="tabletext"><b>Address</b></div></td>
  <td><div class="tabletext">
    <%=contactAddress.getStreet1()%><br>
    <%=contactAddress.getStreet2()%><br>
    <%=contactAddress.getCity()%><br>
    <%=contactAddress.getState()%> &nbsp;
    <%=contactAddress.getPostalCode()%><br>
    <%=contactAddress.getCountry()%></div>
  </td>
</tr>
.
.
.
<wl:repeat
 set="<%=((Map)creditCardsMapObject).keySet().iterator()%>"
 id="creditCard" type="String" count="100000">

<tr>
  <td width="55%">
  <div class="tabletext"><b><%=creditCard%></b></div>
  <td>
</tr>

</wl:repeat>

</table>
```

Notes: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

The `getPropertyAsString` JSP tag is used to directly obtain the customer's first and last name, the customer's home and business phone numbers, and the customer's email address. Listing 3-4 illustrates how to use the `getPropertyAsString` JSP tag to display ths customer's name in the welcome message at the top of the `viewprofile.jsp` template.

Listing 3-4 Obtaining the Customer's Name

```
<p class="head1">
<um:getPropertyAsString propertyName="firstName" />
<um:getPropertyAsString propertyName="lastName" />’s Profile
</p>
```

Form Field Specification

No form fields are used in the `viewprofile.jsp` template.

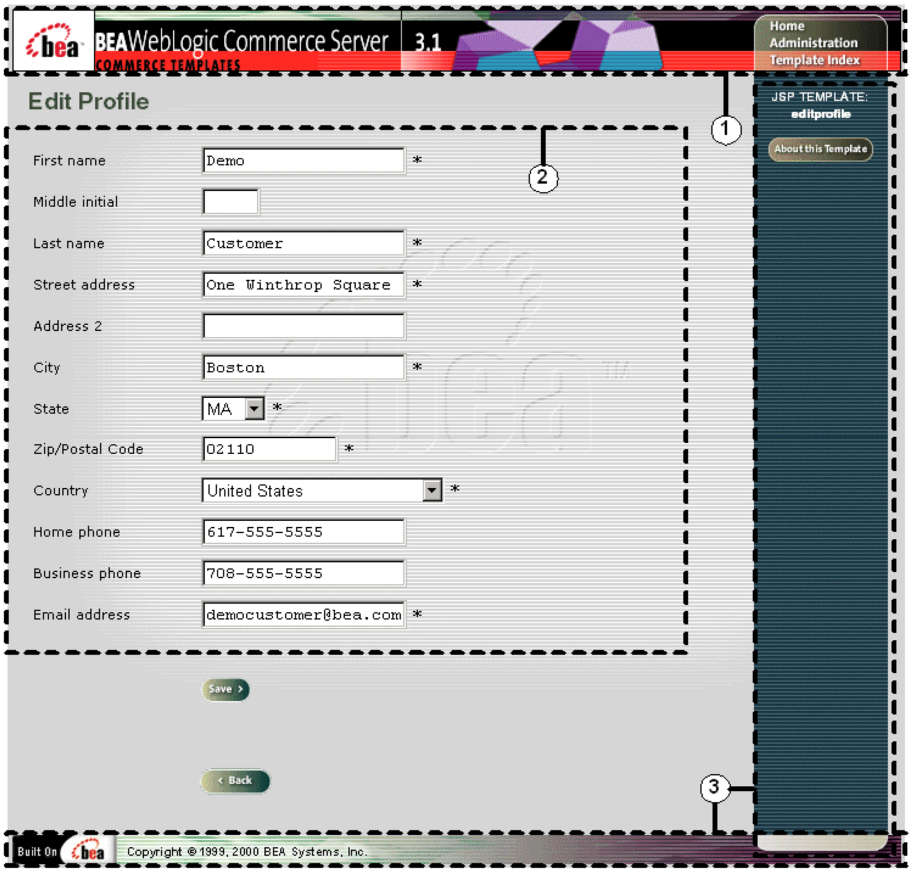
editprofile.jsp Template

The `editprofile.jsp` template (shown in Figure 3-2) allows a registered customer to update the personal information in their stored profile, which includes their name, address, home and business phone numbers, and email address.

Sample Browser View

Figure 3-2 shows an annotated version of the `editprofile.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-2 Annotated editprofile.jsp Template



Edit Profile

First name	Demo	*
Middle initial		
Last name	Customer	*
Street address	One Winthrop Square	*
Address 2		
City	Boston	*
State	MA	*
Zip/Postal Code	02110	*
Country	United States	*
Home phone	617-555-5555	
Business phone	708-555-5555	
Email address	democustomer@bea.com	*

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

```
<%@ include file="/commerce/includes/innerheader.jsp" %>
```

2. Region 2 provides customers with a series of form fields that allow customers to change their personal information. Where available, existing data for the customer is dynamically displayed in the form fields using WebLogic Server and the WebLogic Personalization Server's User Management JSP tags. For the address, this region utilizes the form fields defined in the included `states.jsp` and `countries.jsp` template files. The import calls are:

   ```
   <%@ include file="/commerce/includes/states.jsp" %>
   <%@ include file="/commerce/includes/countries.jsp" %>
   ```

3. The `editprofile.jsp` template's content in region 3 contains the included `innerfooter.jsp` template. The include call in `editprofile.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `editprofile.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`editprofile.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`editprofile.jsp` (UNIX)

Tag Library Imports

The `editprofile.jsp` template uses existing WebLogic Server JSP tags and the WebLogic Personalization Server's User Management JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”weblogic.tld” prefix=”wl” %>
<%@ taglib uri=”um.tld” prefix=”um” %>
```

Note: For more information on the WebLogic Server JSP tags or the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `editprofile.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before `editprofile.jsp` is the page on which a customer can view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, `editprofile.jsp` is reloaded with an appropriate error message.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included into the `editprofile.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `states.jsp`, which contains a list of states that are displayed when the customer is prompted to enter an address.

- `countries.jsp`, which contains a list of countries that are displayed when the customer is prompted to enter an address.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

The `editprofile.jsp` template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-5 provides information about these events and the business logic they invoke.

Table 3-5 editprofile.jsp Events

Event	Webflow Response(s)
`button(back)`	No business logic required. Loads `viewprofile.jsp.`
`button(save)`	`UpdateBasicInfoIP` `EditBasicInfo`

Table 3-6 briefly describes each of the Pipelines from Table 3-5, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see “Pipeline Components” on page 3-65.

Table 3-6 Edit Profile Pipelines

Pipeline	Description
`EditBasicInfo`	Contains `UpdateBasicInfoPC` and is transactional.

Dynamic Data Display

One purpose of the `editprofile.jsp` template is to display the profile information a customer had previously entered. This is accomplished on the `editprofile.jsp` template using a combination of WebLogic Server JSP tags, the WebLogic Personalization Server's User Management JSP tags, and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) for which the customer information should be retrieved, as shown in Listing 3-5.

Listing 3-5 Setting the Customer Context

```
<um:getProfile profileKey="<%=request.getRemoteUser()%>"
 profileType="WLCS_Customer" />
```

Note: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Next, the `getProperty` JSP tag is used to obtain the customer's contact address, which is then initialized with data from the customer object, as shown in Listing 3-6.

Listing 3-6 Obtaining the Customer's Contact Address

```
<um:getProperty propertyName="contactAddress"
 id="contactAddressObject" />
<% Address contactAddress = (Address) contactAddressObject; %>
```

The data stored within the `contactAddress` object can now be accessed by calling accessor methods/attributes within Java scriptlets. Table 3-7 provides more detailed information about the methods/attributes for the contact address.

Table 3-7 contactAddress Accessor Methods/Attributes

Method/Attribute	Description
`getStreet1()`	The first line in the customer's contact street address.
`getStreet2()`	The second line in the customer's contact street address.
`getCity()`	The city in the customer's contact address.
`getCounty()`	The county in the customer's contact address.
`getState()`	The state in the customer's contact address.
`getPostalCode()`	The zip/postal code in the customer's contact address.
`getCountry()`	The country in the customer's contact address.

Note: The `getPropertyAsString` JSP tag is used to directly obtain the customer's first and last name, the customer's home and business phone numbers, and the customer's email address. Listing 3-7 illustrates how to use the `getPropertyAsString` JSP tag to obtain the customer's last name.

Listing 3-7 Obtaining the Customer's Last Name

```
<um:getPropertyAsString propertyName="lastName" id="lastName" />
```

Listing 3-8 illustrates how these accessor methods/attributes are used within Java scriptlets to display existing data within the form fields.

Listing 3-8 Using Accessor Methods/Attributes within editprofile.jsp Java Scriptlets

```
<table>
<tr>

  <um:getPropertyAsString propertyName="lastName" id="lastName" />

  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_LAST_NAME%>"
     fieldDefaultValue="<%=(String)lastName%>"
     fieldValue="customerLastName" fieldStatus="status" validColor="black"
     invalidColor="red" unspecifiedColor="black" fieldColor="fontColor" />

    <div class="tabletext">
      <font color=<%= fontColor %>>Last name </font>
    </div>
  </td>

  <td>
    <input type="text" name="<%=HttpRequestConstants.CUSTOMER_LAST_NAME%>"
    value="<%=customerLastName%>">*
  </td>

</tr><tr>

  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_ADDRESS1%>"
     fieldDefaultValue="<%=contactAddress.getStreet1()%>"
     fieldValue="customerAddress1" fieldStatus="status" validColor="black"
     invalidColor="red" unspecifiedColor="black" fieldColor="fontColor" />

    <div class="tabletext">
      <font color=<%= fontColor %>>Street address</font>
    </div>
  </td>

  <td>
    <input type="text" name="<%=HttpRequestConstants.CUSTOMER_ADDRESS1%>"
    value="<%=customerAddress1%>">*
  </td>

</tr>
</table>
```

Form Field Specification

The primary purpose of the `editprofile.jsp` template is to allow customers to edit their profile information using various HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `editprofile.jsp` template, and a description for each of these form fields are listed in Table 3-8.

Table 3-8 editprofile.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`editprofile.jsp`), used by the Webflow.
`HttpRequestConstants.CUSTOMER_FIRST_NAME`	Textbox	The customer’s first name.
`HttpRequestConstants.CUSTOMER_MIDDLE_NAME`	Textbox	The customer’s middle initial.
`HttpRequestConstants.CUSTOMER_LAST_NAME`	Textbox	The customer’s last name.
`HttpRequestConstants.CUSTOMER_ADDRESS1`	Textbox	The first line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_ADDRESS2`	Textbox	The second line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_CITY`	Textbox	The city in the customer’s address.
`HttpRequestConstants.CUSTOMER_STATE`	Listbox	The state in the customer’s address.
`HttpRequestConstants.CUSTOMER_ZIPCODE`	Textbox	The zip code in the customer’s address.

Table 3-8 editprofile.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.CUSTOMER_COUNTRY`	Listbox	The country in the customer’s address.
`HttpRequestConstants.CUSTOMER_HOME_PHONE`	Textbox	The customer’s home phone number.
`HttpRequestConstants.CUSTOMER_BUSINESS_PHONE`	Textbox	The customer’s business phone number.
`HttpRequestConstants.CUSTOMER_EMAIL`	Textbox	The customer’s email address.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_EMAIL %>`) for use in the JSP.

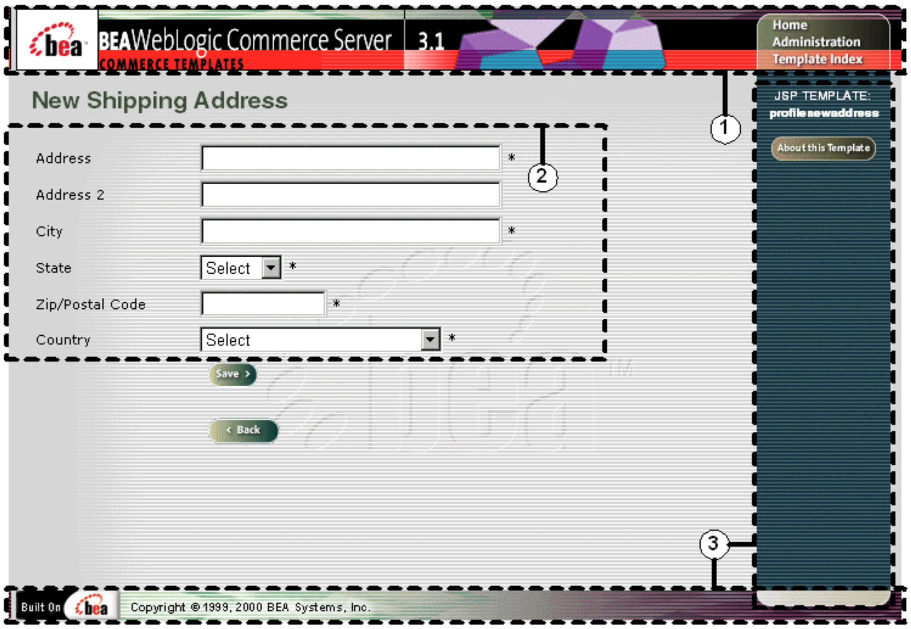
profilenewaddress.jsp Template

The `profilenewaddress.jsp` template (shown in Figure 3-3) allows a registered customer to add a new shipping address to their stored profile.

Sample Browser View

Figure 3-3 shows an annotated version of the `profilenewaddress.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-3 Annotated profilenewaddress.jsp Template



New Shipping Address

Address	*
Address 2	
City	*
State	Select *
Zip/Postal Code	*
Country	Select *

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

```
<%@ include file="/commerce/includes/innerheader.jsp" %>
```

2. Region 2 provides customers with a series of form fields that allow customers to add a shipping address. This region utilizes the form fields defined in the included `newaddresstemplate.jsp` template file, which itself includes the `states.jsp` and `countries.jsp` template files. The import call in `profilenewaddress.jsp` is:

   ```
   <%@ include file="/commerce/includes/newaddresstemplate.jsp” %>
   ```

3. The `profilenewaddress.jsp` template’s content in region 3 contains the included `innerfooter.jsp` template. The include call in `profilenewaddress.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `profilenewaddress.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`profilenewaddress.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`profilenewaddress.jsp` (UNIX)

Tag Library Imports

The `profilenewaddress.jsp` template uses the Webflow and Pipeline JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”webflow.tld” prefix=”webflow” %>
<%@ taglib uri=”pipeline.tld” prefix=”pipeline” %>
```

Note: For more information about the Webflow and Pipeline JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `profilenewaddress.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before `profilenewaddress.jsp` is the page on which a customer can view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, `profilenewaddress.jsp` is reloaded with an appropriate error message.

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `profilenewaddress.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

- newaddresstemplate.jsp, described in "About the Included newaddresstemplate.jsp Template" on page 2-23.

Events

The profilenewaddress.jsp template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-9 provides information about these events and the business logic they invoke.

Table 3-9 profilenewaddress.jsp Events

Event	Webflow Response(s)
button(back)	No business logic required. Loads viewprofile.jsp.
button(save)	UpdateAddressInfoIP ProfileNewAddress

Table 3-10 briefly describes each of the Pipelines from Table 3-9, as they are defined in the pipeline.properties file. For more information about individual Pipeline components, see "Pipeline Components" on page 3-65.

Table 3-10 New Profile Shipping Address Pipelines

Pipeline	Description
ProfileNewAddress	Contains UpdateShippingInfoPC and is transactional.

Dynamic Data Display

No dynamic data is presented on the profilenewaddress.jsp template.

Form Field Specification

The primary purpose of the `profilenewaddress.jsp` template is to allow customers to enter a new shipping address using various HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `profilenewaddress.jsp` template (most of which are actually imported from the `newaddresstemplate.jsp` file), and a description for each of these form fields are shown in Table 3-11.

Table 3-11 profilenewaddress.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`profilenewaddress.jsp`), used by the Webflow.
`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1`	Textbox	The first line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2`	Textbox	The second line in the customer’s street address.
`HttpRequestConstants.CUSTOMER_SHIPPING_CITY`	Textbox	The city in the customer’s address.
`HttpRequestConstants.CUSTOMER_SHIPPING_STATE`	Listbox	The state in the customer’s address.
`HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE`	Textbox	The zip code in the customer’s address.
`HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY`	Listbox	The country in the customer’s address.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY %>`) for use in the JSP.

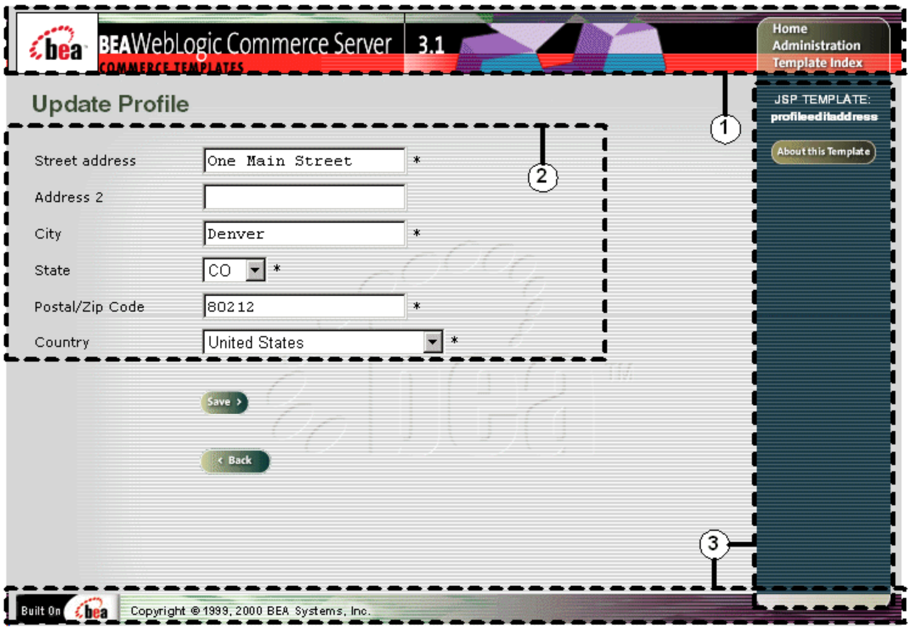
profileeditaddress.jsp Template

The `profileeditaddress.jsp` template (shown in Figure 3-4) allows a registered customer to update the shipping address information stored as part of their profile.

Sample Browser View

Figure 3-4 shows an annotated version of the `profileeditaddress.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-4 Annotated profileeditaddress.jsp Template



Update Profile

Street address	One Main Street	*
Address 2		
City	Denver	*
State	CO	*
Postal/Zip Code	80212	*
Country	United States	*

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

```
<%@ include file="/commerce/includes/innerheader.jsp" %>
```

2. Region 2 provides customers with a series of form fields that allow customers to update a shipping address. This region utilizes the form fields defined in the included `editaddresstemplate.jsp` template file, which itself includes the `states.jsp` and `countries.jsp` template files. The import call in `profileeditaddress.jsp` is:

   ```
   <%@ include file="/commerce/includes/editaddresstemplate.jsp”
   %>
   ```

3. The `profileeditaddress.jsp` template’s content in region 3 contains the included `innerfooter.jsp` template. The include call in `profileeditaddress.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in WebLogic Commerce Server Directory Structure

You can find the `profileeditaddress.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`profileeditaddress.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`profileeditaddress.jsp` (UNIX)

Tag Library Imports

The `profileeditaddress.jsp` template uses the Webflow JSP tags and WebLogic Personalization Server’s User Management JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”webflow.tld” prefix=”webflow” %>
<%@ taglib uri=”um.tld” prefix=”um” %>
```

Note: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation. For more information about the Webflow JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `profileeditaddress.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before the `profileeditaddress.jsp` template is the page that allows a customer to view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, the `profileeditaddress.jsp` template is reloaded with an appropriate error message.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `profileeditaddress.jsp` template:

- `innerheader.jsp`, which creates the top banner.

- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.
- `editaddresstemplate.jsp`, described below.

About the Included editaddresstemplate.jsp Template

The `editaddresstemplate.jsp` template (included in all JSP templates that allow customers to edit a shipping address) provides a standardized format for both the form field presentation and error handling. The form fields are organized in a table, and upon form submission, the input processors associated with the `editaddresstemplate.jsp` template will validate the form to ensure that all required fields contain values. If errors are detected, the `editaddresstemplate.jsp` template will be redisplayed, with an error message at the top and the offending field labels shown in red (as opposed to the original black) font. Further, the information your customer entered correctly will still be displayed in the form.

Since the `editaddresstemplate.jsp` template allows customers to edit an existing shipping address, the form fields on the page are also prefilled with information previously entered by the customer.

The behavior described above is accomplished on the `editaddresstemplate.jsp` template using the `getValidatedValue` JSP tag and the accessor methods/attributes for `defaultShippingAddress`, as shown in Listing 3-9.

Listing 3-9 Use of the getValidatedValue JSP Tag and Accessor Methods/Attributes on editaddresstemplate.jsp

```
<table>
<tr>

<!-- Use the webflow:getValidatedValue to retrieve the default value for the
shipping address from the HttpRequest. This value was placed there by the
CustomerProfileIP input processor. Use the defaultShippingAddress to display the
first line using its getStreet1() accessor method. -->

<tr>
  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1%>"
     fieldDefaultValue="<%=defaultShippingAddress.getStreet1()%>"
    fieldValue="customerShippingAddress1" fieldStatus="status"
    validColor="black" invalidColor="red" unspecifiedColor="black"
    fieldColor="fontColor" />

    <div class="tabletext">
      <font color=<%= fontColor %>>Street address</font>
    </div>

 </td>

                <td>
                  <input type="text"
                   name="<%=HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1%>"
                   value="<%=customerShippingAddress1%>" maxlength="30">*
                </td>

              </tr>
              </table>
```

Notes: For more information about the `getValidatedValue` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

For a list of the available accessor methods/attributes for `defaultShippingAddress`, see Table 3-14.

Because the `editaddresstemplate.jsp` template collects address information, this template also includes `states.jsp` and `countries.jsp` where appropriate.

Events

The `profileeditaddress.jsp` template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-12 provides information about these events and the business logic they invoke.

Table 3-12 profileeditaddress.jsp Events

Event	Webflow Response(s)
`button(back)`	No business logic required. Loads `viewprofile.jsp`.

Table 3-12 profileeditaddress.jsp Events

Event	Webflow Response(s)
`button(save)`	`UpdateShippingInfoIP` `ProfileEditAddress`

Table 3-13 briefly describes each of the Pipelines from Table 3-12, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see "Pipeline Components" on page 3-65.

Table 3-13 Edit Profile Shipping Address Pipelines

Pipeline	Description
`ProfileEditAddress`	Contains `UpdateShippingInfoPC` and is transactional.

Dynamic Data Display

One purpose of the `profileeditaddress.jsp` template is to prepare the address information a customer had previously entered, so the `editaddresstemplate.jsp` template can display this information in the address form fields. This is accomplished on the `profileeditaddress.jsp` template using a combination of Webflow JSP tags, the WebLogic Personalization Server's User Management JSP tags, and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) for which the customer information should be retrieved, as shown in Listing 3-10.

Listing 3-10 Setting the Customer Context

```
<um:getProfile profileKey="<%=request.getRemoteUser()%>"
 profileType="WLCS_Customer" />
```

Note: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Next, the `getProperty` JSP tag is used to obtain a list of the customer's shipping addresses, which are then initialized with data from the customer object, as shown in Listing 3-11.

Listing 3-11 Obtaining the Customer's Shipping Address

```
<um:getProperty propertyName="shippingAddressMap"
 id="shippingAddressMapObject" />

<% Map shippingAddressMap = (Map) shippingAddressMapObject;
   String addressKey = request.getParameter(HttpRequestConstants.ADDRESS_KEY);
  Address defaultShippingAddress = (Address) shippingAddressMap.get(addressKey);
%>
```

The data stored within the `defaultShippingAddress` object can now be accessed by calling accessor methods/attributes within Java scriptlets. Table 3-14 provides more detailed information about the methods/attributes for the default shipping address.

Table 3-14 defaultShippingAddress Accessor Methods/Attributes

Method/Attribute	Description
`getStreet1()`	The first line in the customer's shipping street address.
`getStreet2()`	The second line in the customer's shipping street address.
`getCity()`	The city in the customer's shipping address.
`getCounty()`	The county in the customer's shipping address.
`getState()`	The state in the customer's shipping address.
`getPostalCode()`	The zip/postal code in the customer's shipping address.
`getCountry()`	The country in the customer's shipping address.

Form Field Specification

The primary purpose of the `profileeditaddress.jsp` template is to allow customers to edit their profile information using various HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `profileeditaddress.jsp` template (most of which are actually imported from the `editaddresstemplate.jsp` file), and a description for each of these form fields are listed in Table 3-15.

Table 3-15 profileeditaddress.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`profileeditaddress.jsp`), used by the Webflow.
`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1`	Textbox	The first line in the customer’s shipping address.
`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2`	Textbox	The second line in the customer’s shipping address.
`HttpRequestConstants.CUSTOMER_SHIPPING_CITY`	Textbox	The city in the customer’s shipping address.
`HttpRequestConstants.CUSTOMER_SHIPPING_STATE`	Listbox	The state in the customer’s shipping address.
`HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE`	Textbox	The zip/postal code in the customer’s shipping address.
`HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY`	Listbox	The country in the customer’s shipping address.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY %>`) for use in the JSP.

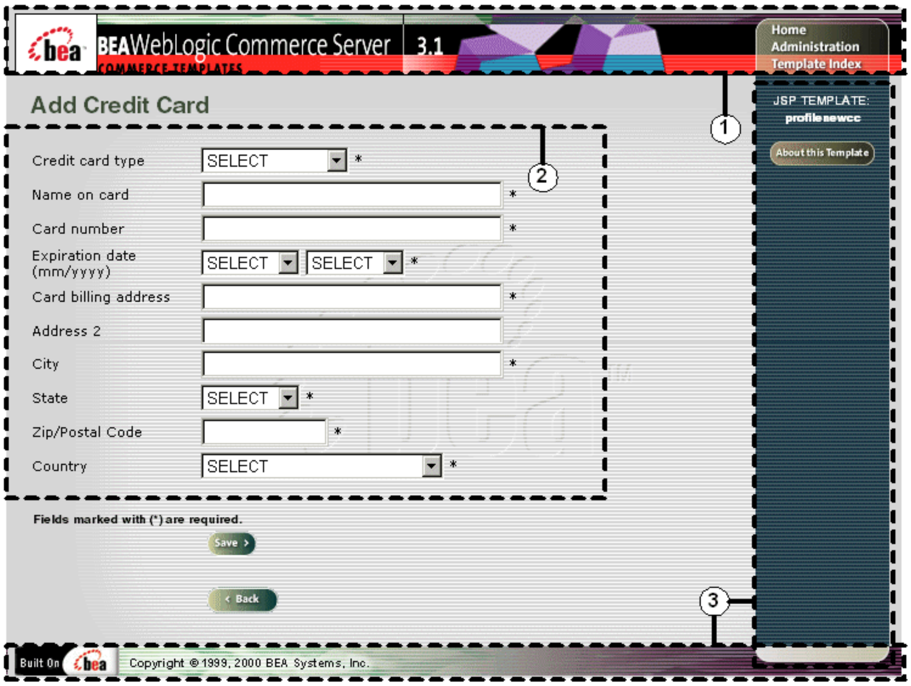
profilenewcc.jsp Template

The `profilenewcc.jsp` template (shown in Figure 3-5) allows an existing customer to add new credit card information, which will be stored as part of their profile.

Sample Browser View

Figure 3-5 shows an annotated version of the `profilenewcc.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-5 Annotated profilenewcc.jsp Template



Add Credit Card

Credit card type	SELECT *
Name on card	*
Card number	*
Expiration date (mm/yyyy)	SELECT SELECT *
Card billing address	*
Address 2	
City	*
State	SELECT *
Zip/Postal Code	*
Country	SELECT *

Fields marked with (*) are required.

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. Region 2 provides customers with a series of form fields that allow customers to enter payment information related to a new credit card. This region utilizes the form fields defined in the included `newcctemplate.jsp` template file, which itself includes the `states.jsp` and `countries.jsp` template files. The include call in `profilenewcc.jsp` is:

   ```
   <%@ include file="/commerce/includes/newcctemplate.jsp” %>
   ```

3. The `profilenewcc.jsp` template’s content in region 3 contains the included `innerfooter.jsp` template. The include call in `profilenewcc.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `profilenewcc.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`profilenewcc.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`profilenewcc.jsp` (UNIX)

Tag Library Imports

The `profilenewcc.jsp` template uses the Webflow JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”webflow.tld” prefix=”webflow” %>
```

Note: For more information about the Webflow JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `profilenewcc.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before the `profilenewcc.jsp` template is the page on which a customer can view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, the `profilenewcc.jsp` template is reloaded.

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `profilenewcc.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.
- `newcctemplate.jsp`, described in "About the Included newcctemplate.jsp Template" on page 2-25.

Events

The `profilenewcc.jsp` template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-16 provides information about these events and the business logic they invoke.

Table 3-16 profilenewcc.jsp Events

Event	Webflow Response(s)
`button(back)`	No business logic required. Loads `viewprofile.jsp.`
`button(save)`	`UpdatePaymentInfoIP` `NewCreditCard`

Table 3-17 briefly describes each of the Pipelines from Table 3-16, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see "Pipeline Components" on page 3-65.

Table 3-17 New Credit Card Pipelines

Pipeline	Description
`NewCreditCard`	Contains `EncryptCreditCardPC` and `UpdatePaymentInfoPC`, and is transactional.

Dynamic Data Display

No dynamic data is presented on the `profilenewcc.jsp` template.

Form Field Specification

The primary purpose of the `profilenewcc.jsp` template is to allow customers to enter new credit card information using various HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `profilenewcc.jsp` template (most of which are actually imported from the `newcctemplate.jsp` file), and a description for each of these form fields are listed in Table 3-18.

Table 3-18 profilenewcc.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`profilenewcc.jsp`), used by the Webflow.
`HttpRequestConstants. CUSTOMER_CREDITCARD_TYPE`	Listbox	The type of the customer’s credit card.
`HttpRequestConstants. CUSTOMER_CREDITCARD_HOLDER`	Textbox	The name on the credit card.
`HttpRequestConstants. CUSTOMER_CREDITCARD_NUMBER`	Textbox	The number of the customer’s credit card.
`HttpRequestConstants. CUSTOMER_CREDITCARD_MONTH`	Listbox	The month of the customer’s credit card expiration date.
`HttpRequestConstants. CUSTOMER_CREDITCARD_YEAR`	Listbox	The year of the customer’s credit card expiration date.
`HttpRequestConstants. CUSTOMER_CREDITCARD_ADDRESS1`	Textbox	The first line in the customer’s billing address.

Table 3-18 profilenewcc.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2`	Textbox	The second line in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_CITY`	Textbox	The city in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_STATE`	Listbox	The state in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE`	Textbox	The zip/postal code in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY`	Listbox	The country in the customer's billing address.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY %>`) for use in the JSP.

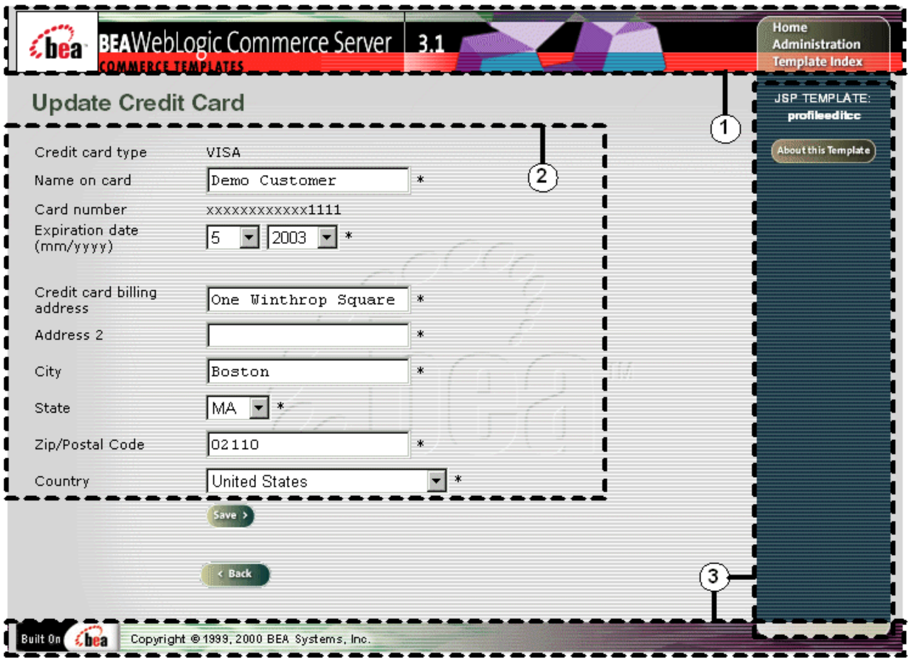
profileeditcc.jsp Template

The `profileeditcc.jsp` template (shown in Figure 3-6) allows a customer to edit existing credit card information, which will be stored as part of their profile.

Sample Browser View

Figure 3-6 shows an annotated version of the `profileeditcc.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-6 Annotated profileeditcc.jsp Template



Update Credit Card

Credit card type	VISA
Name on card	Demo Customer *
Card number	xxxxxxxxxxxx1111
Expiration date (mm/yyyy)	5 2003 *
Credit card billing address	One Winthrop Square *
Address 2	*
City	Boston *
State	MA *
Zip/Postal Code	02110 *
Country	United States *

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/innerheader.jsp" %>
   ```

2. Region 2 provides customers with a series of form fields that allow customers to update their payment information related to a credit card. This region utilizes the form fields defined in the included `editcctemplate.jsp` template file, which itself includes the `states.jsp` and `countries.jsp` template files. The import call in `profileeditcc.jsp` is:

   ```
   <%@ include file="/commerce/includes/editcctemplate.jsp” %>
   ```

3. The `profileeditcc.jsp` template's content in region 3 contains the included `innerfooter.jsp` template. The include call in `profileeditcc.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `profileeditcc.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`profileeditcc.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`profileeditcc.jsp` (UNIX)

Tag Library Imports

The `profileeditcc.jsp` template uses the Webflow JSP tags and WebLogic Personalization Server's User Management JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”webflow.tld” prefix=”webflow” %>
<%@ taglib uri=”um.tld” prefix=”um” %>
```

Note: For more information about the Webflow JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*. For more information on the WebLogic Personalization Server’s User Management JSP tags, see “JSP Tag Reference” in the *BEA WebLogic Personalization Server* documentation.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `profileeditcc.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before the `profileeditcc.jsp` template is the page on which a customer can view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, `profileeditcc.jsp` is reloaded.

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `profileeditcc.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.
- `editcctemplate.jsp`, described below.

About the Included editcctemplate.jsp Template

The `editcctemplate.jsp` template (included in all JSP templates that allow customers to edit a credit card) provides a standardized format for both the form field presentation and error handling. The form fields are organized in a table, and upon form submission, the input processors associated with the `editcctemplate.jsp` template will validate the form to ensure that all required fields contain values. If errors are detected, the `editcctemplate.jsp` template will be redisplayed, with an error message at the top and the offending field labels shown in red (as opposed to the original black) font. Further, the information your customer entered correctly will still be displayed in the form.

Since the `editcctemplate.jsp` template allows customers to edit an existing shipping address, the form fields on the page are also prefilled with information previously entered by the customer.

The behavior described above is accomplished on the `editcctemplate.jsp` template using the `getValidatedValue` JSP tag and the accessor methods/attributes for `defaultCreditCard`, as shown in Listing 3-12.

Listing 3-12 Use of the getValidatedValue JSP Tag and Accessor Methods/Attributes on editcctemplate.jsp

```
<table>
<tr>

<!-- use the webflow:getValidatedValue to retrieve a value from the HttpRequest.
This value was placed there by the CustomerProfileIP input processor -->

  <td>
    <webflow:getValidatedValue
     fieldName="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER%>"
     fieldDefaultValue="<%=defaultCreditCard.getName()%>"
     fieldValue="customerCreditCardHolder" fieldStatus="status"
     validColor="black" invalidColor="red" unspecifiedColor="black"
     fieldColor="fontColor" />

     <div class="tabletext">
       <font color=<%= fontColor %>>Name on card</font>
     </div>
  </td>

  <td>
    <input type="text"
     name="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER%>"
     value="<%=customerCreditCardHolder%>">*
  </td>

</tr>
</table>
```

Notes: For more information about the `getValidatedValue` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

For a list of the available accessor methods/attributes for `defaultCreditCard`, see Table 3-21.

Because the `editcctemplate.jsp` template collects address information, this template also includes `states.jsp` and `countries.jsp` where appropriate.

Events

The `profileeditcc.jsp` template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-19 provides information about these events and the business logic they invoke.

Table 3-19 profileeditcc.jsp Events

Event	Webflow Response(s)
`button(back)`	No business logic required. Loads `viewprofile.jsp.`
`button(save)`	`UpdatePaymentInfoIP` `UpdateCreditCard`

Table 3-20 briefly describes each of the Pipelines from Table 3-19, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see “Pipeline Components” on page 3-65.

Table 3-20 Edit Credit Card Pipelines

Pipeline	Description
`UpdateCreditCard`	Contains `UpdatePaymentPC` and is transactional.

Dynamic Data Display

One purpose of the `profileeditcc.jsp` template is to prepare the credit card information a customer had previously entered, so the `editcctemplate.jsp` template can display this information in the payment information form fields. This is accomplished on the `profileeditcc.jsp` template using a combination the WebLogic Personalization Server’s User Management JSP tags and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) for which the customer information should be retrieved, as shown in Listing 3-13.

Listing 3-13 Setting the Customer Context

```
<um:getProfile profileKey="<%=request.getRemoteUser()%>"
 profileType="WLCS_Customer" />
```

Note: For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Next, the `getProperty` JSP tag is used to obtain the customer's list of credit cards (and related billing information), which is then initialized with data from the customer object, as shown in Listing 3-14.

Listing 3-14 Obtaining the Customer's Credit Cards and Billing Information

```
<um:getProperty propertyName="creditCardsMap"
 id="creditCardsMapObject" />

<%

Map creditCardsMap = (Map) creditCardsMapObject;
String creditCardKey =
  request.getParameter(HttpRequestConstants.CREDITCARD_KEY);
CreditCard defaultCreditCard = null;
defaultCreditCard = (CreditCard)
creditCardsMap.get(creditCardKey);
Address billingAddress = (Address)
defaultCreditCard.getBillingAddress();

%>
```

The data stored within the `defaultCreditCard` and `billingAddress` objects can now be accessed by calling accessor methods/attributes within Java scriptlets. Table 3-21 provides more detailed information about the methods/attributes for the default credit card, while Table 3-22 provides more information about the accessor methods/attributes on `billingAddress`.

Table 3-21 defaultCreditCard Accessor Methods/Attributes

Method/Attribute	Description
`getType()`	The credit card type (VISA, MasterCard, AMEX, etc.).
`getName()`	The credit card holder's name.
`getDisplayNumber()`	The credit card number for display (12 Xs and last 4 digits).
`getNumber()`	The credit card number.
`getExpirationDate()`	The credit card's expiration date.

Table 3-22 billingAddress Accessor Methods/Attributes

Method/Attribute	Description
`getStreet1()`	The first line in the customer's billing street address.
`getStreet2()`	The second line in the customer's billing street address.
`getCity()`	The city in the customer's billing address.
`getCounty()`	The county in the customer's billing address.
`getState()`	The state in the customer's billing address.
`getPostalCode()`	The zip/postal code in the customer's billing address.
`getCountry()`	The country in the customer's billing address.

Form Field Specification

Another purpose of the `profileeditcc.jsp` template is to allow customers to make changes to their credit card information using various HTML form fields. Unknown to your customers, it is also used to pass needed information to the Webflow.

The form fields used in the `profileeditcc.jsp` template (most of which are actually imported from the `editcctemplate.jsp` file), and a description for each of these fields are listed in Table 3-23.

Table 3-23 profileeditcc.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`profileeditcc.jsp`), used by the Webflow.
`HttpRequestConstants.CUSTOMER_CREDITCARD_KEY_ORIGINAL`	Hidden	The map key of the customer’s credit card.
`HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE`	Listbox	The type of the customer’s credit card.
`HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER`	Textbox	The name on the credit card.
`HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER`	Textbox	The number of the customer’s credit card.
`HttpRequestConstants.CUSTOMER_CREDITCARD_DISPLAY_NUMBER`	Hidden	The display version of the customer’s credit card (12 Xs and last 4 digits).
`HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH`	Listbox	The month of the customer’s credit card expiration date.
`HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR`	Listbox	The year of the customer’s credit card expiration date.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1`	Textbox	The first line in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2`	Textbox	The second line in the customer’s billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_CITY`	Textbox	The city in the customer’s billing address.

Table 3-23 profileeditcc.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.CUSTOMER_CREDITCARD_STATE`	Listbox	The state in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE`	Textbox	The zip/postal code in the customer's billing address.
`HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY`	Listbox	The country in the customer's billing address.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY %>`) for use in the JSP.

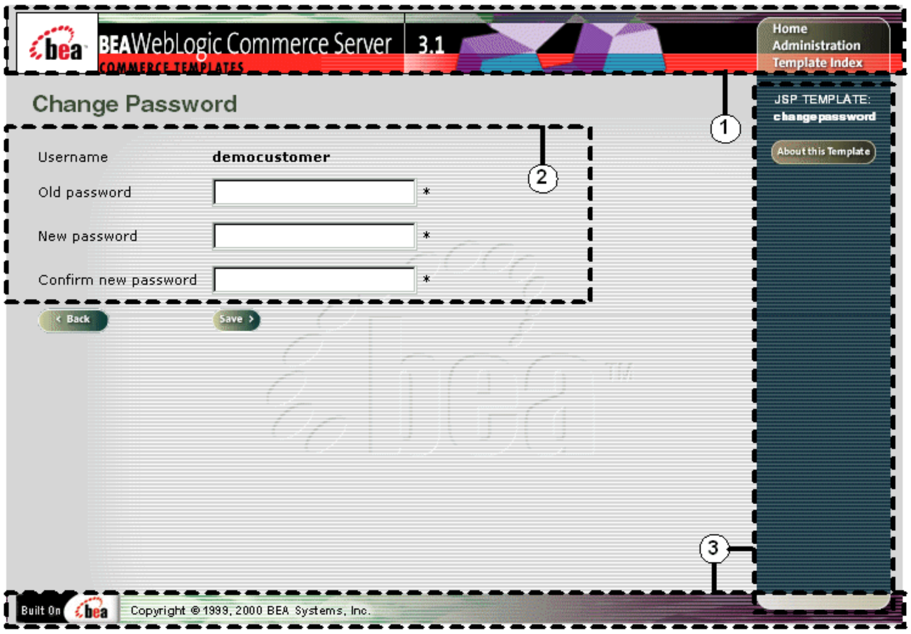
changepassword.jsp Template

The `changepassword.jsp` template (shown in Figure 3-7) allows a customer to change their password, which will be stored as part of their profile.

Sample Browser View

Figure 3-7 shows an annotated version of the `changepassword.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 3-7 Annotated changepassword.jsp Template



Change Password

Username	democustomer
Old password	*
New password	*
Confirm new password	*

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `innerheader.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

```
<%@ include file="/commerce/includes/innerheader.jsp" %>
```

2. Region 2 provides customers with a series of form fields that allow customers to change their password, by first entering their old password, then entering and confirming their new password.

3. The `changepassword.jsp` template's content in region 3 contains the included `innerfooter.jsp` template. The include call in `changepassword.jsp` is:

   ```
   <%@ include file="/commerce/includes/innerfooter.jsp" %>
   ```

   `innerfooter.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `innerfooter.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `changepassword.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\user\`
`changepassword.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/user/`
`changepassword.jsp` (UNIX)

Tag Library Imports

The `changepassword.jsp` template uses Webflow JSP tags and WebLogic Personalization Server's User Management JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri="webflow.tld" prefix="webflow" %>
<%@ taglib uri="um.tld" prefix="um" %>
```

Note: For more information about the Webflow JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*. For more information on the WebLogic Personalization Server's User Management JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `changepassword.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```

Location in Default Webflow

The page before the `changepassword.jsp` template is the page on which a customer can view their current profile (`viewprofile.jsp`). If there are no errors in the form submission, the next page in the default Webflow is `viewprofile.jsp`. If corrections do need to be made, `changepassword.jsp` is reloaded.

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `changepassword.jsp` template:

- `innerheader.jsp`, which creates the top banner.
- `innerfooter.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

The `changepassword.jsp` template presents customers with two button events that trigger a particular response in the default Webflow, thereby allowing customers to continue. While this response can be to load another JSP, it is usually the case that an input processor or Pipeline is invoked first. Table 3-24 provides information about these events and the business logic they invoke.

Table 3-24 changepassword.jsp Events

Event	Webflow Response(s)
`button(back)`	No business logic required. Loads `viewprofile.jsp`.
`button(save)`	`UpdateAccountInfoIP` `UpdateAccountProfile`

Table 3-25 briefly describes each of the Pipelines from Table 3-24, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see "Pipeline Components" on page 3-65.

Table 3-25 Change Password Pipelines

Pipeline	Description
`UpdateAccountProfile`	Contains `UpdatePasswordPC` and is transactional.

Dynamic Data Display

One purpose of the `changepassword.jsp` template is to display the customer's username. This is accomplished on the `changepassword.jsp` template using a simple Java scriptlet, as shown in Listing 3-15.

Listing 3-15 Displaying the Customer's Username

```
...

  <td>
    <div class="tabletext">
      <b><%=request.getRemoteUser()%></b>
    </div>
  </td>

...
```

Note: Customers cannot change their username, only their password. If the New Password and Confirm New Password form fields are not filled in correctly, the page is displayed with all fields empty (that is, no fields are dynamically prefilled upon reload).

Form Field Specification

The primary purpose of the `changepassword.jsp` template is to allow customers to make changes to their password using HTML form fields. It is also used to pass needed information to the Webflow.

The form fields used in the `changepassword.jsp` template, and a description for each of these form fields are listed in Table 3-26.

Table 3-26 changepassword.jsp Form Fields

Parameter Name	Type	Description
`“event”`	Hidden	Indicates which event has been triggered. It is used by the Webflow to determine what happens next.
`“origin”`	Hidden	The name of the current page (`changepassword.jsp`), used by the Webflow.
`HttpRequestConstants. PASSWORD`	Password	The customer's existing password used to login.

Table 3-26 changepassword.jsp Form Fields

Parameter Name	Type	Description
`HttpRequestConstants.NEW_PASSWORD`	Password	The new password chosen by the customer for login.
`HttpRequestConstants.CONFIRM_PASSWORD`	Password	Confirmation of the new password chosen by the customer for login.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CONFIRM_PASSWORD %>`) for use in the JSP.

Input Processors

This section provides a brief description of each input processor associated with the Customer Profile Services JSP template(s).

DeleteCreditCardIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.` `DeleteCreditCardIP`
Description	Deletes a `CreditCard` from the `CreditCardMap` and creates a new `CustomerValue` object; then sets the `CreditCardMap` on `CustomerValue` and places it into the Pipeline session.
Required `HTTPServletRequest` Parameters	`HttpRequestConstants.CREDITCARD_KEY`
Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER`
Removed Pipeline Session Attributes	None
Validation	Verifies that `HttpRequestConstants.CREDITCARD_KEY` is not `NULL`.
Exceptions	`InvalidInputException`, thrown if `HttpRequestConstants.CREDITCARD_KEY` is `NULL`. `InvalidSessionStateException`, thrown if the session is unavailable or has expired.

DeleteShippingAddressIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.DeleteShippingAddressIP`
Description	Deletes a `ShippingAddress` from the `ShippingAddressMap` and creates a new `CustomerValue` object; then sets the `ShippingAddressMap` on `CustomerValue` and places it into the Pipeline session.
Required `HTTPServletRequest` Parameters	`HttpRequestConstants.ADDRESS_KEY`
Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER`
Removed Pipeline Session Attributes	None
Validation	Verifies that `HttpRequestConstants.ADDRESS_KEY` is not `NULL`.
Exceptions	`InvalidInputException`, thrown if `HttpRequestConstants.ADDRESS_KEY` is `NULL`. `InvalidSessionStateException`, thrown if the session is unavailable or has expired.

UpdateAccountInfoIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.` `UpdateAccountInfoIP`
Description	Processes the customer's input from the `changepassword.jsp`. Creates a `CustomerValue` object in the Pipeline session containing the new information.

Required `HTTPServletRequest` Parameters	`HttpRequestConstants.PASSWORD`
	`HttpRequestConstants.NEW_PASSWORD`
	`HttpRequestConstants.CONFIRM_PASSWORD`

Required Pipeline Session Attributes	None
Updated Pipeline Session Attributes	`PipelineSessionConstants.PASSWORD`
Removed Pipeline Session Attributes	None
Validation	Validates the current password and verifies that the required fields contain values.
Exceptions	`InvalidInputException`, thrown when the current password is incorrect, when the required fields do not contain values, or if the new password and confirm password values do not match. `ProcessingException`, thrown in the case of a configuration error.

UpdateBasicInfoIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.UpdateBasicInfoIP`
Description	Processes the customer's input from the `editprofile.jsp`. Creates a `CustomerValue` object in the Pipeline session containing the new information.

Required `HTTPServletRequest` Parameters	`HttpRequestConstants.CUSTOMER_FIRST_NAME`
	`HttpRequestConstants.CUSTOMER_MIDDLE_NAME`
	`HttpRequestConstants.CUSTOMER_LAST_NAME`
	`HttpRequestConstants.CUSTOMER_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_CITY`
	`HttpRequestConstants.CUSTOMER_STATE`
	`HttpRequestConstants.CUSTOMER_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_COUNTRY`
	`HttpRequestConstants.CUSTOMER_HOME_PHONE`
	`HttpRequestConstants.CUSTOMER_BUSINESS_PHONE`
	`HttpRequestConstants.CUSTOMER_EMAIL`

Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER`
Removed Pipeline Session Attributes	None
Validation	Verifies that the required fields contain values.
Exceptions	`InvalidInputException`, thrown if the required fields do not contain values. `ProcessingException`, thrown if the required Pipeline session attributes are not available.

UpdatePaymentInfoIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.` `UpdatePaymentInfoIP`
Description	Processes the customer's input from `profilenewcc.jsp` and `profileeditcc.jsp`. Creates a `CustomerValue` object in the Pipeline session containing the new information.

Required `HTTPServletRequest` Parameters	`HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_DISPLAY_NUMBER`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_CITY`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_STATE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY`

Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER` `PipelineSessionConstants.CREDITCARD_KEY`
Removed Pipeline Session Attributes	None
Validation	Verifies that the required fields contain values, and verifies that the length of the credit card number is not less than 16 digits (15 digits for AMEX).
Exceptions	`InvalidInputException`, thrown if the required fields do not contain values or the credit card number is less than the minimum required for the type. `InvalidSessionStateException`, thrown when the session is unavailable or has expired.

UpdateShippingInfoIP

Class Name	`com.beasys.commerce.ebusiness.customer.webflow.UpdateShippingInfoIP`
Description	Processes the customer's input from the `profileeditaddress.jsp`. Creates a `CustomerValue` object in the Pipeline session containing the new information.

Required `HTTPServletRequest` Parameters	`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1`
	`HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2`
	`HttpRequestConstants.CUSTOMER_SHIPPING_CITY`
	`HttpRequestConstants.CUSTOMER_SHIPPING_STATE`
	`HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE`
	`HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY`

Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER`
Removed Pipeline Session Attributes	None
Validation	Verifies that the required fields contain values.
Exceptions	`InvalidInputException`, thrown when the required fields do not contain values. `InvalidSessionStateException`, thrown if the session is unavailable or is expired.

Pipeline Components

This section provides a brief description of each Pipeline component associated with the Customer Profile Services JSP template(s).

Note: Some Pipeline components extend other, base Pipeline components. For more information on the base classes, see the *Javadoc*.

UpdateBasicInfoPC

Class Name	`com.beasys.commerce.ebusiness.customer.pipeline.UpdateBasicInfoPC`
Description	Updates the `Customer` object for changes made by `UpdateBasicInfoIP`. This Pipeline component must stay in sync with the the `UpdateBasicInfoIP` input processor.
Required Pipeline Session Attributes	None
Updated Pipeline Session Attributes	None
Removed Pipeline Session Attributes	None
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when the Pipeline component is not able to set the customer’s properties.

UpdatePaymentInfoPC

Class Name	`com.beasys.commerce.ebusiness.customer.pipeline.UpdatePaymentInfoPC`
Description	Updates the `Customer` object for changes made by `UpdatePaymentInfoIP`. This Pipeline component must stay in sync with the the `UpdatePaymentInfoIP` input processor.
Required Pipeline Session Attributes	None
Updated Pipeline Session Attributes	None
Removed Pipeline Session Attributes	None
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when the Pipeline component is not able to set the customer's properties.

UpdateShippingInfoPC

Class Name	`com.beasys.commerce.ebusiness.customer.pipeline.UpdateShippingInfoPC`
Description	Updates the `Customer` object for changes made by `UpdateShippingInfoIP`. This Pipeline component must stay in sync with the the `UpdateShippingInfoIP` input processor.
Required Pipeline Session Attributes	None

Updated Pipeline Session Attributes	None
Removed Pipeline Session Attributes	None
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when the Pipeline component is not able to set the customer's properties.

UpdatePasswordPC

Class Name	`com.beasys.commerce.ebusiness.customer.pipeline.UpdatePasswordPC`
Description	Retrieves the `USER_NAME` and `PASSWORD` from the Pipeline session and updates the password for the user.
Required Pipeline Session Attributes	`PipelineSessionConstants.CUSTOMER` `PipelineSessionConstants.PASSWORD`
Updated Pipeline Session Attributes	None
Removed Pipeline Session Attributes	`PipelineSessionConstants.PASSWORD`
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when the Pipeline component is not able to set the customer's properties.

CHAPTER

4 Customer Self-Service

Customers who make purchases from an e-commerce site often want access to their order and payment history. In many cases, customers expect to have this information available. To meet this need, the Registration and User Processing package provides you with a series of JSPs designed specifically for this purpose. The customer self-service pages allow registered customers who have previously placed orders with your e-business to locate information about their past orders and payments, and to check on the status of these orders. The customer self-service pages can help you maintain a high level of service for all your customers by giving them the information they require. This topic describes each of the customer self-service pages in detail.

This topic includes the following sections:

- JavaServer Pages (JSPs)
  - main.jsp Template
  - orderhistory.jsp Template
  - orderstatus.jsp Template
  - paymenthistory.jsp Template
- Input Processors
  - SelectOrderForViewingIP
- Pipeline Components
  - RefreshOrderHistoryPC
  - RefreshPaymentHistoryPC

JavaServer Pages (JSPs)

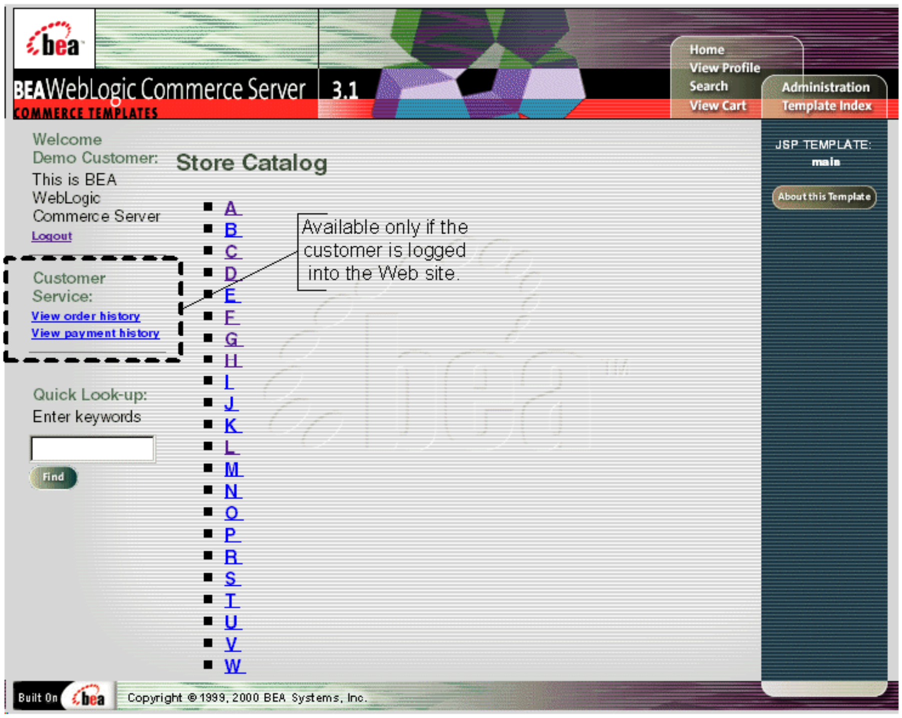
Like the other services available in the Registration and User Processing package, customer self-service is implemented through a number of JavaServer Pages (JSPs). You can use these JSPs as an out-of-the-box solution, or customize them to meet your unique business requirements. This section describes each of these pages in detail.

Note: For a description of the complete set of JSPs used in the WebLogic Commerce Server Web application and a listing of their locations in the directory structure, see the Summary of JSP Templates documentation.

main.jsp Template

The gateway into the customer self-service pages is via the `main.jsp` template (shown in Figure 4-1), or the home page for the product catalog. A customer must be logged into your e-commerce site for the customer self-service options to be available. For more information about the `main.jsp` template, see "The Product Catalog JSP Templates and JSP Tags" in the *BEA WebLogic Commerce Server Product Catalog Management* documentation.

Figure 4-1 The main.jsp Template's Customer Self-Service Section



Welcome Demo Customer:
This is BEA WebLogic Commerce Server
Logout

Customer Service:
View order history
View payment history

Quick Look-up:
Enter keywords

Store Catalog

- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
- K
- L
- M
- N
- O
- P
- R
- S
- T
- U
- V
- W

Available only if the customer is logged into the Web site.

Copyright © 1999, 2000 BEA Systems, Inc.

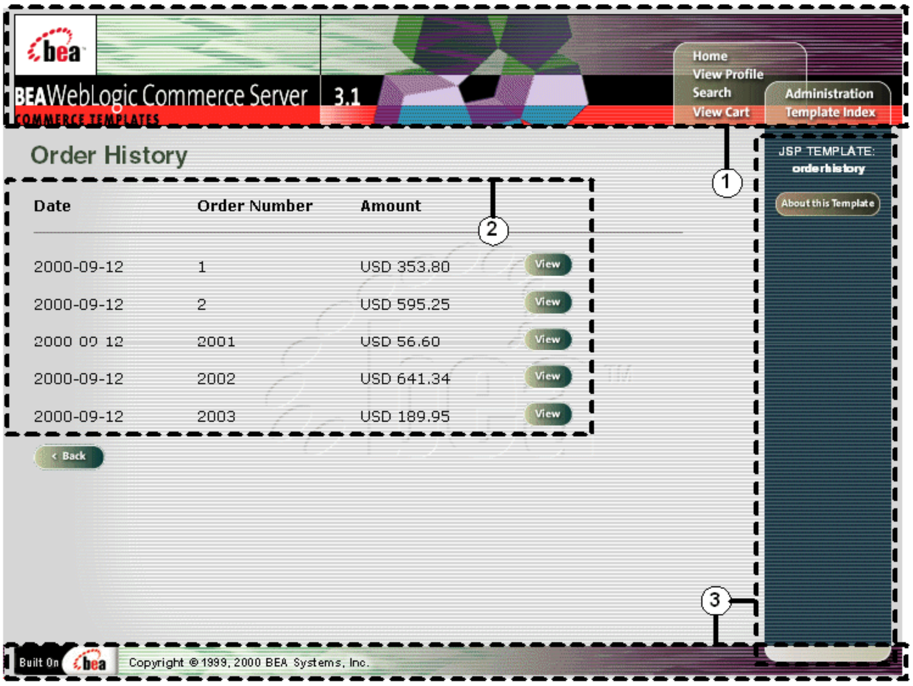
orderhistory.jsp Template

The `orderhistory.jsp` template (shown in Figure 4-2) displays a list of order summaries (including order date, order number, and order amount) for each of the customer's orders. It also provides the customer with a View button for each order in the list, which allows the customer to view details about the order, including its status.

Sample Browser View

Figure 4-2 shows an annotated version of the `orderhistory.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 4-2 Annotated orderhistory.jsp Template



Order History

Date	Order Number	Amount
2000-09-12	1	USD 353.80
2000-09-12	2	USD 595.25
2000-09-12	2001	USD 56.60
2000-09-12	2002	USD 641.34
2000-09-12	2003	USD 189.95

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `header2.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/header2.jsp" %>
   ```

2. This region is the main content area for the page, which contains dynamically-generated data about the customer's order history. The dynamic content on `orderhistory.jsp` is obtained using Pipeline JSP tags and displayed by iterating through the orders using WebLogic Server JSP tags. For the `orderhistory.jsp` template, the only form posts are View (per order), allowing customers to locate more detailed information about a particular order in their order history.

3. The `orderhistory.jsp` template's content in region 3 contains the included `footer2.jsp` template. The include call in `orderhistory.jsp` is:

   ```
   <%@ include file="/commerce/includes/footer2.jsp" %>
   ```

   `footer2.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `footer2.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `orderhistory.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\order\orderhistory.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/order/orderhistory.jsp` (UNIX)

Tag Library Imports

The `orderhistory.jsp` template uses WebLogic and Pipeline JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”weblogic.tld” prefix=”wl” %>
<%@ taglib uri=”pipeline.tld” prefix=”pipeline” %>
```

Note: For more information about the Pipeline JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*. For more information on the WebLogic Server JSP tags, see “JSP Tag Reference” in the *BEA WebLogic Personalization Server* documentation.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `orderhistory.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="java.util.*" %>
<%@ page import="java.text.*" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.axiom.units.*" %>
<%@ page import="com.beasys.commerce.ebusiness.shipping.*" %>
<%@ page import="com.beasys.commerce.ebusiness.order.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
```

Location in Default Webflow

Customers arrive at the `orderhistory.jsp` template from the product catalog home page (`main.jsp`). From here, customers can return back to the product catalog home page, or display the details of a specific order by selecting it (`orderstatus.jsp`).

Note: For more information about the default Webflow, see “Overview of the Registration and User Processing Package” on page 1-1.

Included JSP Templates

The following JSP templates are included in the `orderhistory.jsp` template:

- `header2.jsp`, which creates the top banner.
- `footer2.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

Every time a customer clicks a button to view more detail about an order, it is considered an event. Each event triggers a particular response in the default Webflow that allows them to continue. While this response can be to load another JSP, it is usually the case that an input processor and/or Pipeline is invoked first. Table 4-1 provides information about these events and the business logic they invoke.

Table 4-1 orderhistory.jsp Events

Event	Webflow Response(s)
`--`	`RefreshOrderHistory`
`button(viewOrderStatus)`	`SelectOrderForViewingIP`

Table 4-2 briefly describes each of the Pipelines from Table 4-1, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see “Pipeline Components” on page 4-28.

Table 4-2 Order History Pipelines

Pipeline	Description
`RefreshOrderHistory`	Contains `RefreshOrderHistoryPC` and is not transactional.

Note: Although the `RefreshOrderHistory` Pipeline is associated with the `orderhistory.jsp` template, it is not triggered by an event on the page. Rather, the `RefreshOrderHistory` Pipeline is executed before the `orderhistory.jsp` is viewed, to locate the orders associated with the customer requesting the information.

Dynamic Data Display

One purpose of the `orderhistory.jsp` template is to display the data specific to a customer's orders for their review and possible selection. This is accomplished on `orderhistory.jsp` using a combination of WebLogic Server JSP tags, Pipeline JSP tags, and attributes/methods.

First, the `getPipelineProperty` JSP tag retrieves the `ORDER_HISTORY` attribute from the Pipeline session. Table 4-3 provides more detailed information on this attribute.

Table 4-3 orderhistory.jsp Pipeline Session Attributes

Attribute	Type	Description
`PipelineSessionConstants .ORDER_HISTORY`	List of `com.beasys.commerce. ebusiness.order.OrderValue`	List of the orders available for the customer.

Listing 4-1 illustrates how this attribute is retrieved from the Pipeline session using the `getPipelineProperty` JSP tag.

Listing 4-1 Retrieving the Order History Attribute

```
<pipeline:getPipelineProperty
propertyName="<%=PipelineSessionConstants.ORDER_HISTORY%>"
returnName="orderHistory" returnType="java.util.List"/>
```

Note: For more information on the `getPipelineProperty` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

The data stored within the Pipeline session attribute is then accessed by using accessor methods/attributes within Java scriptlets. Table 4-4 provides more detailed information about these methods/attributes for `OrderValue`.

Table 4-4 OrderValue Accessor Methods/Attributes

Method/Attribute	Description
`createdDate`	The date the customer's order was created.
`identifier`	Key in the database for the order.

`getTotal(int totalType)`	The total amount specified by the `totalType` parameter. Valid parameters include:
	
	`OrderConstants.LINE_UNIT_PRICE_TIMES_QUANTITY`
	`OrderConstants.LINE_SHIPPING`
	`OrderConstants.LINE_TAX`
	
	Note: The `getTotal()` method also allows you to combine different total types. For more information, see the *Javadoc*.

Listing 4-2 illustrates how these accessor methods/attributes are used within Java scriptlets along with the WebLogic Server JSP tags to display the information.

Listing 4-2 Using Accessor Methods/Attributes Within orderhistory.jsp Java Scriptlets

```
<wl:repeat set="<%=orderHistory%>" id="orderValue" type="OrderValue"
count="100">

<table>
<tr>
  <td>
    <div class="tabletext"><%=orderValue.createdDate%></div>
  </td>
  <td>
    <div class="tabletext"><%=orderValue.identifier%></div>
  </td>
  <td>
    <div class="tabletext">
    <% Money total =
       orderValue.getTotal(OrderConstants.LINE_UNIT_PRICE_TIMES_QUANTITY
       + OrderConstants.LINE_SHIPPING + OrderConstants.LINE_TAX); %>
    <%=total.getCurrency()%>
    <%=WebflowJSPHelper.priceFormat(total.getValue())%>
    </div>
  </td>
</tr>
</table>

</wl:repeat>
```

Note: For more information on the WebLogic Server JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Form Field Specification

No form fields are used in the `orderhistory.jsp` template.

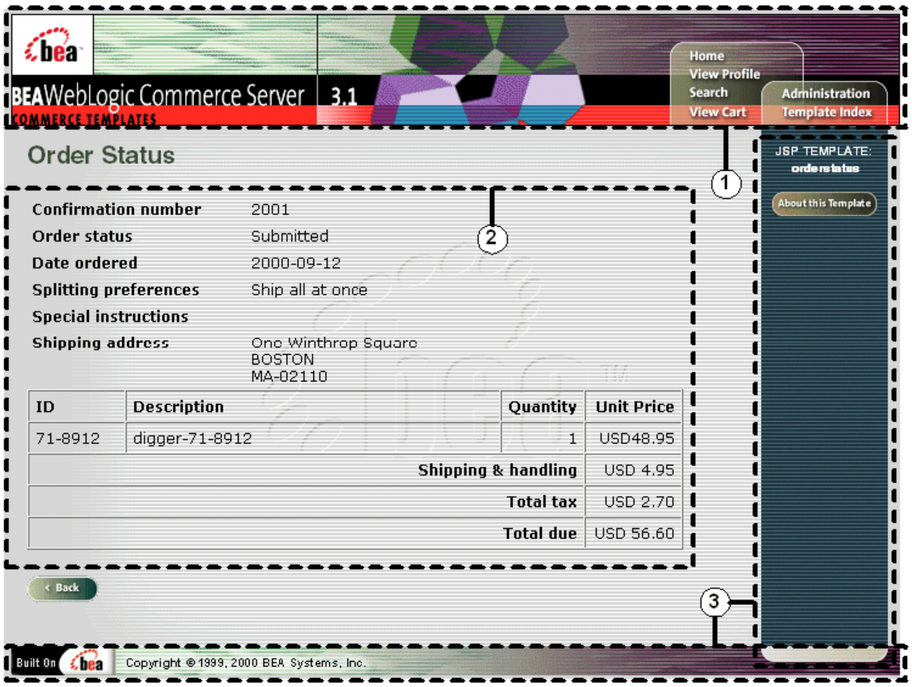
orderstatus.jsp Template

The `orderstatus.jsp` template (shown in Figure 4-3) displays a variety of information for the order summary the customer selected from the list presented on the `orderhistory.jsp` template. This order information includes the order confirmation number, the order status, the date the order was placed, splitting instructions, special instructions, the shipping address, information related to the specific shopping cart items (name, description, quantity, unit price), and total amounts (shipping and handling, tax, and total order cost).

Sample Browser View

Figure 4-3 shows an annotated version of the `orderstatus.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 4-3 Annotated orderstatus.jsp Template



Order Status

Confirmation number	2001
Order status	Submitted
Date ordered	2000-09-12
Splitting preferences	Ship all at once
Special instructions	
Shipping address	One Winthrop Square BOSTON MA-02110

ID	Description	Quantity	Unit Price
71-8912	digger-71-8912	1	USD48.95
		Shipping & handling	USD 4.95
		Total tax	USD 2.70
		Total due	USD 56.60

Copyright © 1999, 2000 BEA Systems, Inc.

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `header2.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/header2.jsp" %>
   ```

2. This region is the main content area for the page, which contains dynamically-generated data about a particular order the customer selected from the `orderhistory.jsp` template. The dynamic content on `orderstatus.jsp` is obtained using Pipeline JSP tags and displayed by iterating through the shopping cart items using WebLogic Server JSP tags.

3. The `orderstatus.jsp` template's content in region 3 contains the included `footer2.jsp` template. The include call in `orderstatus.jsp` is:

   ```
   <%@ include file="/commerce/includes/footer2.jsp" %>
   ```

   `footer2.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `footer2.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in WebLogic Commerce Server Directory Structure

You can find the `orderstatus.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\order\`
`orderstatus.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/order/`
`orderstatus.jsp` (UNIX)

Tag Library Imports

The `orderstatus.jsp` template uses WebLogic and Pipeline JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri="weblogic.tld" prefix="wl" %>
<%@ taglib uri="pipeline.tld" prefix="pipeline" %>
```

Note: For more information on the WebLogic Server JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation. For more information about the Pipeline JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `orderstatus.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="java.util.*" %>
<%@ page import="java.text.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page import="com.beasys.commerce.axiom.units.*" %>
<%@ page import="com.beasys.commerce.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.order.*" %>
<%@ page import="com.beasys.commerce.ebusiness.payment.*" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
```

Location in Default Web Flow

Customers arrive at the `orderstatus.jsp` template from the page that displays summaries of their past orders (`orderhistory.jsp`). The default Webflow does not define a subsequent JSP template.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `orderstatus.jsp` template:

- `header2.jsp`, which creates the top banner.
- `footer2.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

There are no events on the `orderstatus.jsp` template.

Dynamic Data Display

The purpose of the `orderstatus.jsp` template is to display the data specific to a customer's order for their review. This is accomplished on `orderstatus.jsp` using a combination of WebLogic Server JSP tags, Pipeline JSP tags, and accessor methods/attributes.

First, the `getPipelineProperty` JSP tag retrieves the `SELECTED_ORDER` attribute from the Pipeline session. Table 4-5 provides more detailed information on this attribute.

Table 4-5 orderstatus.jsp Pipeline Session Attributes

Attribute	Type	Description
`PipelineSessionConstants .SELECTED_ORDER`	`com.beasys.commerce. ebusiness.order.OrderValue`	Contains a variety of information about the order selected by the customer.

Listing 4-3 illustrates how this attribute is retrieved from the Pipeline session using the `getPipelineProperty` JSP tag.

Listing 4-3 Retrieving the Selected Order Attribute

```
<pipeline:getPipelineProperty
  propertyName="<%=PipelineSessionConstants.SELECTED_ORDER%>"
  returnName="orderValue"
  returnType="com.beasys.commerce.ebusiness.order.OrderValue"/>
```

Note: For more information on the `getPipelineProperty` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

The data stored within the Pipeline session attribute is then accessed by using accessor methods/attributes within Java scriptlets. Table 4-6 provides more detailed information about these methods/attributes for `OrderValue`.

Table 4-6 OrderValue Accessor Methods/Attributes

Method/Attribute	Description
`createdDate`	The date the customer's order was created.
`identifier`	Key in the database for the order; the order confirmation number.
`orderStatus`	The status of the order.
`splittingPreference`	The splitting preference for the order.
`specialInstructions`	Any special instructions for the order.
`shippingAddress`	The shipping address for the order.
`orderLines`	A collection of the lines in the shopping cart that make up the customer's order.

`getTotal(int totalType)`	The total amount specified by the `totalType` parameter. Valid parameters include:
	
	`OrderConstants.LINE_UNIT_PRICE_TIMES_QUANTITY`
	`OrderConstants.LINE_SHIPPING`
	`OrderConstants.LINE_TAX`
	
	Note: The `getTotal()` method also allows you to combine different total types. For more information, see the *Javadoc*.

Table 4-7 describes the accessor methods/attributes available within the `shippingAddress` attribute of `OrderValue`.

Table 4-7 shippingAddress Accessor Methods

Method/Attribute	Description
`getStreet1()`	The first line of the customer's street address.
`getStreet2()`	The second line of the customer's street address.
`getCity()`	The city in the customer's address.
`getCounty()`	The county in the customer's address.
`getState()`	The state in the customer's address.
`getPostalCode()`	The zip/postal code in the customer's address.
`getCountry()`	The country in the customer's address.

Table 4-8 describes the accessor methods/attributes available for each `OrderLine` of the `OrderLines` attribute.

Table 4-8 OrderLine Accessor Methods

Method/Attribute	Description
`getProductIdentifier()`	The name (identifier) for the shopping cart item.
`getDescription()`	A description of the shopping cart item.
`getQuantity()`	The quantity of the shopping cart item.
`getUnitPrice()`	The unit price for the shopping cart item.

The `getUnitPrice()` method also has accessor methods/attributes that you can use. These are shown in Table 4-9.

Table 4-9 getUnitPrice() Accessor Methods/Attributes

Method/Attribute	Description
`getCurrency()`	Obtains the currency associated with the amount.

Table 4-9 getUnitPrice() Accessor Methods/Attributes

Method/Attribute	Description
`getValue()`	Obtains the value of the amount.

Listing 4-4 illustrates how these accessor methods/attributes are used within Java scriptlets along with the WebLogic Server JSP tags to display the information.

Listing 4-4 Using Accessor Methods/Attributes Within orderstatus.jsp Java Scriptlets

```
<table border="0" width="90%" cellpadding="5" cellspacing="0">
<tr>
  <td><div class="tabletext"><b>Confirmation number</b></div></td>
  <td><div class="tabletext"><%=orderValue.identifier%></div></td>
</tr>
.
.
.

<tr>
  <td><div class="tabletext"><b>Shipping address</b></div></td>
  <td><div class="tabletext">
      <%=orderValue.shippingAddress.getStreet1()%><br>
      <%=orderValue.shippingAddress.getStreet2()%><br>
      <%=orderValue.shippingAddress.getCity()%><br>
      <%String stateZip  = orderValue.shippingAddress.getState()+
      "-" + orderValue.shippingAddress.getPostalCode();%><br>
      <%=stateZip%></div>
  </td>
</tr>
</table>

.
.
.
<wl:repeat set="<%=orderValue.orderLines.iterator()%>"
id="orderLine" type="OrderLine" count="100">

<table>
<tr>
  <td>
    <div class="tabletext">
    <%=orderLine.getProductIdentifier()%>
    </div>
  </td>
  <td>
    <div class="tabletext"><%=orderLine.getDescription()%></div>
  </td>
  <td align="right">
    <div class="tabletext">
    <%=quantityFormat.format(orderLine.getQuantity())%>
    </div>
  </td>
  <td align="right">
    <div class="tabletext">
    <%=orderLine.getUnitPrice().getCurrency()%>
    <%=WebflowJSPHelper.priceFormat(orderLine.getUnitPrice().
        getValue())%>
    </div>
  </td>
</tr>
</table>

</wl:repeat>
```

Note: For more information on the WebLogic Server JSP tags, see the "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation.

Form Field Specification

No form fields are used in the `orderstatus.jsp` template.

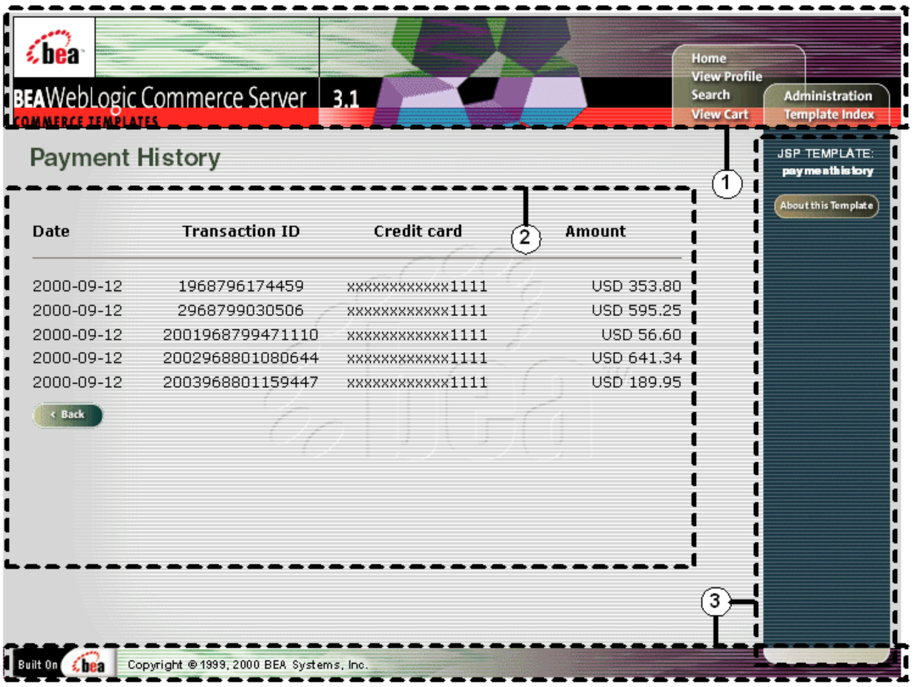
paymenthistory.jsp Template

The `paymenthistory`.jsp template (shown in Figure 4-4) allows the customer to view information regarding the payments that have been made. This information includes the date, the payment transaction ID, the credit card used, and the amount that was billed to the credit card.

Sample Browser View

Figure 4-4 shows an annotated version of the `paymenthistory.jsp` template. The dashed lines and numbers in the diagram are not part of the template; they are referenced in the explanation that follows the screen shot.

Figure 4-4 Annotated paymenthistory.jsp Template



Payment History

Date	Transaction ID	Credit card	Amount
2000-09-12	1968796174459	xxxxxxxxxxxx1111	USD 353.80
2000-09-12	2968799030506	xxxxxxxxxxxx1111	USD 595.25
2000-09-12	2001968799471110	xxxxxxxxxxxx1111	USD 56.60
2000-09-12	2002968801080644	xxxxxxxxxxxx1111	USD 641.34
2000-09-12	2003968801159447	xxxxxxxxxxxx1111	USD 189.95

The numbers in the following list refer to the numbered regions in the figure:

1. The page header (top banner) is created from an import of the `header2.jsp` template. This is standard across many of the JSP templates provided by WebLogic Commerce Server. The import call is:

   ```
   <%@ include file="/commerce/includes/header2.jsp" %>
   ```

2. This region is the main content area for the page, which contains dynamically-generated data about a customer's payments. This dynamically generated data is obtained and displayed using a combination of Pipeline JSP tags, WebLogic Server JSP tags, and accessor methods/attributes.

3. The `paymenthistory.jsp` template's content in region 3 contains the included `footer2.jsp` template. The include call in `paymenthistory.jsp` is:

   ```
   <%@ include file="/commerce/includes/footer2.jsp" %>
   ```

   `footer2.jsp` consists of the horizontal footer at the bottom of the page, plus the right-side vertical column that describes (for the benefit of you and your development team) the name of the current template and links to its *About* information. In the `footer2.jsp` file, the right-side vertical column is an include file:

   ```
   <%@ include file="/commerce/includes/rightside.jsp" %>
   ```

Location in the WebLogic Commerce Server Directory Structure

You can find the `paymenthistory.jsp` template file at the following location, where `WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\commerce\order\`
`paymenthistory.jsp` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/commerce/order/`
`paymenthistory.jsp` (UNIX)

Tag Library Imports

The `paymenthistory.jsp` template uses WebLogic and Pipeline JSP tags. Therefore, the template includes the following JSP tag libraries:

```
<%@ taglib uri=”weblogic.tld” prefix=”wl” %>
<%@ taglib uri=”pipeline.tld” prefix=”pipeline” %>
```

Note: For more information on the WebLogic Server JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Personalization Server* documentation. For more information about the Pipeline JSP tags, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

These files reside in the following directory for the WebLogic Commerce Server Web application:

`%WL_COMMERCE_HOME%\server\webapps\wlcs\WEB-INF` (Windows)
`$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF` (UNIX)

Java Package Imports

The `paymenthistory.jsp` template uses Java classes in the following packages and therefore includes these import statements:

```
<%@ page import="java.util.*" %>
<%@ page import="java.text.*" %>
<%@ page import="com.beasys.commerce.foundation.pipeline.*" %>
<%@ page import="com.beasys.commerce.ebusiness.payment.*" %>
<%@ page import="com.beasys.commerce.webflow.*" %>
```

Location in Default Webflow

Customers arrive at `paymenthistory.jsp` from the product catalog home page (`main.jsp`). The default Webflow does not define a subsequent JSP template.

Note: For more information about the default Webflow, see "Overview of the Registration and User Processing Package" on page 1-1.

Included JSP Templates

The following JSP templates are included in the `paymenthistory.jsp` template:

- `header2.jsp`, which creates the top banner.
- `footer2.jsp`, which creates a horizontal footer at the bottom of the page, and also includes the `rightside.jsp` template. `rightside.jsp` describes (for the benefit of you and your development team) the name of the current template and links to its *About* information.

Events

There are no events on the `paymenthistory.jsp` template that trigger input processors or Pipelines in the Webflow. However, Table 4-10 briefly describes each of the Pipelines associated with the `paymenthistory.jsp` template, as they are defined in the `pipeline.properties` file. For more information about individual Pipeline components, see "Pipeline Components" on page 4-28.

Table 4-10 Payment History Pipelines

Pipeline	Description
`RefreshPaymentHistory`	Contains `RefreshPaymentHistoryPC` and is not transactional.

Note: Although the `RefreshPaymentHistory` Pipeline is associated with the `paymenthistory.jsp` template, it is not triggered by an event on the page. Rather, the `RefreshPaymentHistory` Pipeline is executed before the `paymenthistory.jsp` is viewed, to locate the payments associated with the customer requesting the information.

Dynamic Data Display

The purpose of the `paymenthistory.jsp` template is to display the data specific to a customer's payments for their review. This is accomplished on `paymenthistory.jsp` using a combination of WebLogic Server JSP tags, Pipeline JSP tags, and accessor methods/attributes.

First, the `getPipelineProperty` JSP tag retrieves the `PAYMENT_HISTORY` attribute from the Pipeline session. Table 4-11 provides more detailed information on this attribute.

Table 4-11 paymenthistory.jsp Pipeline Session Attributes

Attribute	Type	Description
`PipelineSessionConstants.PAYMENT_HISTORY`	List of `com.beasys.commerce.ebusiness.payment.PaymentTransactionValue`	List of the payments available for the customer.

Listing 4-5 illustrates how this attribute is retrieved from the Pipeline session using the `getPipelineProperty` JSP tag.

Listing 4-5 Retrieving the Payment History Attribute

```
<pipeline:getPipelineProperty
 propertyName="<%=PipelineSessionConstants.PAYMENT_HISTORY%>"
 returnName="paymentHistory" returnType="java.util.List"
 attributeScope="<%=PipelineConstants.REQUEST_SCOPE%>"/>
```

Note: For more information on the `getPipelineProperty` JSP tag, see *BEA WebLogic Commerce Server Webflow and Pipeline Management*.

The data stored within the Pipeline session attribute is then accessed by using accessor methods/attributes within Java scriptlets. Table 4-12 provides more detailed information about these methods/attributes for `PaymentTransactionValue`.

Table 4-12 PaymentTransactionValue Accessor Methods/Attributes

Method/Attribute	Description
`transactionDate`	The date of the payment transaction.
`transactionId`	Key in the database for the transaction; the payment confirmation number.
`creditCard`	The status of the order.
`transactionAmount`	The splitting preference for the order.

The `creditCard` and `transactionAmount` attributes also have accessor methods/attributes, as shown in Table 4-13 and Table 4-14.

Table 4-13 creditCard Accessor Methods/Attributes

Method/Attribute	Description
`getDisplayNumber()`	Obtains the displayable version of the credit card number (12 Xs and last 4 digits).

Table 4-14 transactionAmount Accessor Methods/Attributes

Method/Attribute	Description
`getCurrency()`	Obtains the currency associated with the transaction amount.
`getValue()`	Obtains the value of the transaction amount.

Listing 4-6 illustrates how these accessor methods/attributes are used within Java scriptlets along with the WebLogic Server JSP tags to display the information.

Listing 4-6 Using Accessor Methods/Attributes Within paymenthistory.jsp Java Scriptlets

```
<wl:repeat set="<%=paymentHistory%>" id="paymentTransactionValue"
type="PaymentTransactionValue" count="100">

<table>
<tr>
  <td align="left">
    <div class="tabletext">
    <%=paymentTransactionValue.transactionDate%>
    </div>
  </td>
  <td align="center">
    <div class="tabletext">
    <%=paymentTransactionValue.transactionId%>
    </div>
  </td>
  <td align="center">
    <div class="tabletext">
    <%=paymentTransactionValue.creditCard.getDisplayNumber()%>
    </div>
  </td>
  <td align="right">
    <div class="tabletext">
     <%=paymentTransactionValue.transactionAmount.getCurrency()%>
     <%=WebflowJSPHelper.priceFormat(paymentTransactionValue.
        transactionAmount.getValue())%></div>
  </td>
</tr>
</table>
```

```
</wl:repeat>
```

Note: For more information on the WebLogic Server JSP tags, see "JSP Tag Reference" in the *BEA WebLogic Server Personalization* documentation.

Form Field Specification

No form fields are used in the `paymenthistory.jsp` template.

Input Processors

This section provides a brief description of each input processor associated with the Customer Self-Service JSP template(s).

SelectOrderForViewingIP

Class Name	`com.beasys.commerce.ebusiness.order.webflow.SelectOrderForViewingIP`
Description	Reads the order identifier and uses it to locate an `OrderValue` object from the `ORDER_HISTORY` attribute, then places the object in the Pipeline session.
Required `HTTPServletRequest` Parameters	`HttpRequestConstants.ORDER_IDENTIFIER`
Required Pipeline Session Attributes	`PipelineSessionConstants.ORDER_HISTORY`
Updated Pipeline Session Attributes	`PipelineSessionConstants.SELECTED_ORDER`
Removed Pipeline Session Attributes	None
Validation	None
Exceptions	`ProcessingException`, thrown when the required Pipeline session attributes are not available.

Pipeline Components

This section provides a brief description of each Pipeline component associated with the Customer Self-Service JSP template(s).

Note: Some Pipeline components extend other, base Pipeline components. For more information on the base classes, see the *Javadoc*.

RefreshOrderHistoryPC

Class Name	`com.beasys.commerce.ebusiness.order.pipeline.RefreshOrderHistoryPC`
Description	Uses the `USER_NAME` Pipeline session attribute to obtain the customer's order history.
Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.ORDER_HISTORY`
Removed Pipeline Session Attributes	None
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when required Pipeline session attributes are not available.

RefreshPaymentHistoryPC

Class Name	`com.beasys.commerce.ebusiness.order.pipeline.RefreshPaymentHistoryPC`
Description	Uses the `USER_NAME` Pipeline session attribute to obtain the customer's payment history.
Required Pipeline Session Attributes	`PipelineSessionConstants.USER_NAME`
Updated Pipeline Session Attributes	`PipelineSessionConstants.PAYMENT_HISTORY`
Removed Pipeline Session Attributes	None
Type	Java class
JNDI Name	None
Exceptions	`PipelineFatalException`, thrown when required Pipeline session attributes are not available.

Index

A

accessor method(s)
    ShoppingCart 2-18

B

badlogin.jsp
    default Webflow 2-10
    directory locations 2-9
    events 2-10
    form field specification 2-11
    included JSP templates 2-10
business logic
    orderhistory.jsp 4-7
    successfullogin.jsp 2-16

C

changepassword.jsp
    about 3-53
    default Webflow 3-55
    directory locations 3-54
    dynamic data display 3-56
    events 3-56
    form field specifications 3-57
    included JSP templates 3-55
    Java import statements 3-55
    JSP tag libraries 3-54
    parameters 3-57
    pipeline components 3-56
Customer Profile Services 3-1
customer registration 2-1
customer self-service 4-1
customer support contact information xi

D

database schema 1-7
documentation, where to find it x

E

editaddresstemplate.jsp
    about 3-32
    getValidatedValue JSP tag 3-32
editcctemplate.jsp
    about 3-46
    getValidatedValue JSP tag 3-46
editprofile.jsp
    about 3-13
    default Webflow 3-16
    directory locations 3-15
    dynamic data display 3-18
    events 3-17
    form field specification 3-21
    included JSP templates 3-16
    Java import statements 3-16
    Java scriptlets 3-20
    JSP tag libraries 3-15
    methods/attributes 3-18
    parameters 3-21
    pipeline components 3-17
Events

badlogin.jsp 2-10
changepassword.jsp 3-56
editprofile.jsp 3-17
login.jsp 2-6
newuser.jsp 2-26
orderhistory.jsp 4-7
paymenthistory.jsp 4-23
profileeditaddress.jsp 3-33
profileeditcc.jsp 3-48
profilenewaddress.jsp 3-26
profilenewcc.jsp 3-40
successfullogin.jsp 2-16
viewprofile.jsp 3-7

I

input processors 2-30
CustomerProfileIP 2-30
DeleteCreditCardIP 3-59
DeleteShippingAddressIP 3-60
SelectOrderforViewingIP 4-27
UpdateAccountInfoIP 3-61
UpdateBasicInfoIP 3-62
UpdatePaymentInfoIP 3-63
UpdateShippingInfoIP 3-64

J

JavaServer Page (JSP)
overview 2-2
JavaServer Page templates
changepassword.jsp 3-53
editaddresstemplate.jsp 3-32
editprofile.jsp 3-13
login.jsp 2-2
main.jsp 4-2
newcctemplate.jsp 2-25
newuser.jsp 2-19
orderhistory.jsp 4-4
orderstatus.jsp 4-11
paymenthistory.jsp 4-20
profileeditaddress.jsp 3-29
profileeditcc.jsp 3-43
profilenewaddress.jsp 3-23
profilenewcc.jsp 3-37
successfullogin.jsp 2-12
viewprofile.jsp 3-2

L

login services 2-1
login.jsp
about 2-2
default Webflow 2-4
directory locations 2-4
events 2-6
form field specification 2-6
included JSP templates 2-5
Java import statements 2-4

M

main.jsp
about 4-2

N

newaddresstemplate.jsp
about 2-23
getValidatedValue JSP tag 2-24
newcctemplate.jsp
about 2-25
getValidatedValid JSP tag 2-25
newuser.jsp
about 2-19
default Webflow 2-23
directory locations 2-22
events 2-26
form field specification 2-27
included JSP templates 2-23
Java import statements 2-22
JSP tag libraries 2-22

newaddresstemplate.jsp 2-23
newcctemplate.jsp 2-25
parameters 2-27
pipeline components 2-26

O

orderhistory.jsp
about 4-4
default Webflow 4-6
directory location 4-5
dynamic data display 4-8
events 4-7
included JSP templates 4-7
Java import statements 4-6
JSP tag libraries 4-6
methods/attributes 4-9
pipeline components 4-7
orderstatus.jsp
about 4-11
default Webflow 4-14
directory locations 4-13
dynamic data display 4-15
included JSP templates 4-15
Java import statements 4-14
JSP tag libraries 4-14
methods/attributes 4-16
pipeline components 4-15

P

paymenthistory.jsp
about 4-20
attributes 4-23
default Webflow 4-22
directory locations 4-21
dynamic data display 4-23
events 4-23
included JSP templates 4-22
Java import statements 4-22
JSP tag libraries 4-21
methods/attributes 4-24
pipeline components 4-23
pipeline components 2-33
RefreshOrderHistoryPC 4-28
RegisterUserPC 2-33
UpdateBasicInfoPC 3-65
UpdatePasswordPC 3-67
UpdatePaymentInfoPC 3-66
UpdateShippingInfoPC 3-66
printing product documentation x
profileeditaddress.jsp
about 3-29
default Webflow 3-31
directory locations 3-30
dynamic data display 3-34
editaddresstemplate.jsp 3-32
events 3-33
form field specification 3-35
included JSP templates 3-31
Java import statements 3-31
JSP tag libraries 3-30
methods/attributes 3-35
parameters 3-36
pipeline components 3-34
profileeditcc.jsp
about 3-43
default Webflow 3-45
directory locations 3-44
dynamic data display 3-48
editcctemplate.jsp 3-46
events 3-48
form field specifications 3-50
included JSP templates 3-45
Java import statements 3-45
JSP tag libraries 3-44
parameters 3-50
pipeline components 3-48
profilenewaddress.jsp
about 3-23
default Webflow 3-25
directory locations 3-24

events 3-26
form field specifications 3-27
included JSP templates 3-25
Java import statements 3-25
JSP tag libraries 3-24
parameters 3-27
profilenewcc.jsp
about 3-37
default Webflow 3-39
directory locations 3-38
events 3-40
form field specifications 3-41
included JSP templates 3-40
Java import statements 3-39
JSP tag libraries 3-39
parameters 3-41
pipeline components 3-40

R

Registration and User Processing package
architecture 1-4
customer profile 3-1
customer registration 2-1
customer self-service 4-1
database schema 1-7
default Webflow 1-4
overview 1-2
related information xi
retrieving Pipeline session attributes
shopping cart 2-17, 4-8

S

schema
about 1-7
successfullogin.jsp
about 2-12
default Webflow 2-15
directory locations 2-15
dynamic data display 2-17
events 2-16
included JSP templates 2-16
Java import statements 2-15
JSP tag libraries 2-15
session attributes 2-17
support
technical xi

V

viewprofile.jsp
about 3-2
default Webflow 3-6
directory locations 3-4
dynamic data display 3-8
events 3-7
included JSP templates 3-6
Java import statements 3-5
JSP tag libraries 3-5
methods/attributes 3-9
pipeline components 3-7