



BEA WebLogic Portal™

Guide to Registering Customers
and Managing Customer Services

Version 4.0
Document Date: October 2001

Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, WebLogic, Tuxedo, and Jolt are registered trademarks of BEA Systems, Inc. How Business Becomes E-Business, BEA WebLogic E-Business Platform, BEA Builder, BEA Manager, BEA eLink, BEA WebLogic Portal, BEA Campaign Manager for WebLogic, BEA WebLogic Commerce Server, BEA WebLogic Personalization Server, BEA E-Business Control Center, BEA WebLogic Process Integrator, BEA WebLogic Collaborate, BEA WebLogic Enterprise, and BEA WebLogic Server are trademarks of BEA Systems, Inc.

All other product names may be trademarks of the respective companies with which they are associated.

Guide to Registering Customers and Managing Customer Services

Document Edition	Date	Software Version
4.0	October 2001	BEA WebLogic Portal 4.0

Contents

About This Document

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

1. Overview of Registering Customers and Managing Customer Services

What Are the Registering Customers and Managing Customer Services?	1-2
High-level Architecture	1-3
Template Quick Reference	1-6
About the Database Schema	1-8
Development Roles	1-8
Next Steps	1-9

2. Customer Registration and Login Services

JavaServer Pages (JSPs)	2-2
Login.jsp Template	2-3
Sample Browser View	2-3
Location in the 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 Directory Structure.....	2-9
Tag Library Imports	2-9
Java Package Imports	2-9
Location in Default Webflow.....	2-9
Included JSP Templates	2-10
Events	2-10
Dynamic Data Display	2-10
Form Field Specification	2-10
newuser.jsp Template	2-11
Sample Browser View.....	2-11
Location in the Directory Structure.....	2-14
Tag Library Imports	2-14
Java Package Imports	2-14
Location in Default Webflow.....	2-15
Included JSP Templates	2-15
Events	2-19
Dynamic Data Display	2-20
Form Field Specification	2-20
newusercreation.jsp Template	2-25
Sample Browser View.....	2-25
Location in the Directory Structure.....	2-26
Tag Library Imports	2-27
Java Package Imports	2-27
Location in Default Webflow.....	2-27
Included JSP Templates	2-28
Events	2-29
Dynamic Data Display	2-29
Form Field Specification	2-30
newuserforward.jsp Template	2-31
Location in the Directory Structure.....	2-31
Tag Library Imports	2-31

Java Package Imports.....	2-31
Location in Default Webflow	2-31
Included JSP Templates	2-32
Events.....	2-32
Dynamic Data Display	2-32
Form Field Specification.....	2-32
usercreationforward.jsp Template.....	2-33
Location in the Directory Structure	2-33
Tag Library Imports	2-33
Java Package Imports.....	2-33
Location in Default Webflow	2-33
Included JSP Templates	2-34
Events.....	2-34
Dynamic Data Display	2-34
Form Field Specification.....	2-34
Input Processors.....	2-35
CustomerProfileIP	2-35
LoginCustomerIP	2-37
Pipeline Components.....	2-39
RegisterUserPC	2-39
EncryptCreditCardPC.....	2-40

3. Customer Profile Services

JavaServer Pages (JSPs).....	3-2
viewprofile.jsp Template.....	3-2
Sample Browser View	3-3
Location in the Directory Structure	3-4
Tag Library Imports	3-4
Java Package Imports.....	3-4
Location in Default Webflow	3-5
Included JSP Templates	3-6
Events.....	3-7
Dynamic Data Display	3-8
Form Field Specification.....	3-12
editprofile.jsp Template	3-12

Sample Browser View.....	3-13
Location in the Directory Structure.....	3-14
Tag Library Imports	3-14
Java Package Imports	3-14
Location in Default Webflow.....	3-15
Included JSP Templates	3-15
Events	3-16
Dynamic Data Display	3-16
Form Field Specification.....	3-19
profilenewaddress.jsp Template	3-21
Sample Browser View.....	3-21
Location in the Directory Structure.....	3-22
Tag Library Imports	3-22
Java Package Imports	3-23
Location in Default Webflow.....	3-23
Included JSP Templates	3-23
Events	3-24
Dynamic Data Display	3-25
Form Field Specification.....	3-25
profileeditaddress.jsp Template.....	3-27
Sample Browser View.....	3-27
Location in Commerce services Directory Structure	3-27
Tag Library Imports	3-28
Java Package Imports	3-28
Location in Default Webflow.....	3-29
Included JSP Templates	3-29
Events	3-31
Dynamic Data Display	3-31
Form Field Specification.....	3-33
profilenewcc.jsp Template	3-35
Sample Browser View.....	3-35
Location in the Directory Structure.....	3-36
Tag Library Imports	3-37
Java Package Imports	3-37
Location in Default Webflow.....	3-37

Included JSP Templates	3-38
Events.....	3-39
Dynamic Data Display	3-39
Form Field Specification.....	3-39
profileeditcc.jsp Template.....	3-41
Sample Browser View	3-41
Location in the Directory Structure	3-42
Tag Library Imports	3-43
Java Package Imports.....	3-43
Location in Default Webflow	3-43
Included JSP Templates	3-44
Events.....	3-46
Dynamic Data Display	3-47
Form Field Specification.....	3-49
changepassword.jsp Template.....	3-51
Sample Browser View	3-51
Location in the Directory Structure	3-52
Tag Library Imports	3-52
Java Package Imports.....	3-53
Location in Default Webflow	3-53
Included JSP Templates	3-54
Events.....	3-55
Dynamic Data Display	3-55
Form Field Specification.....	3-56
editdemographics.jsp Template.....	3-57
Sample Browser View	3-57
Location in the Directory Structure	3-59
Tag Library Imports	3-59
Java Package Imports.....	3-60
Location in Default Webflow	3-60
Included JSP Templates	3-61
Events.....	3-62
Dynamic Data Display	3-62
Form Field Specification.....	3-62
Input Processors.....	3-65

DeleteCreditCardIP	3-65
DeleteShippingAddressIP.....	3-66
UpdateAccountInfoIP.....	3-67
UpdateBasicInfoIP	3-68
UpdateDemographicInfoIP.....	3-69
UpdatePaymentInfoIP	3-70
UpdateShippingInfoIP.....	3-71
Pipeline Components	3-72
UpdateBasicInfoPC	3-72
UpdateDemographicInfoPC	3-73
UpdatePasswordPC	3-74
UpdatePaymentInfoPC	3-75
UpdateShippingInfoPC.....	3-76

4. Customer Self-Service

JavaServer Pages (JSPs)	4-2
orderhistory.jsp Template.....	4-4
Sample Browser View.....	4-4
Location in the Directory Structure.....	4-5
Tag Library Imports	4-5
Java Package Imports	4-5
Location in Default Webflow.....	4-6
Included JSP Templates	4-6
Events	4-7
Dynamic Data Display	4-7
Form Field Specification	4-9
orderstatus.jsp Template.....	4-10
Sample Browser View.....	4-10
Location in Commerce services Directory Structure	4-11
Tag Library Imports	4-11
Java Package Imports	4-11
Location in Default Web Flow.....	4-12
Included JSP Templates	4-12
Events	4-13
Dynamic Data Display	4-13

Form Field Specification.....	4-17
paymenthistory.jsp Template	4-18
Sample Browser View	4-18
Location in the Directory Structure	4-19
Tag Library Imports	4-19
Java Package Imports.....	4-19
Location in Default Webflow	4-20
Included JSP Templates	4-20
Events.....	4-20
Dynamic Data Display	4-21
Form Field Specification.....	4-24
Input Processors.....	4-25
SelectOrderForViewingIP	4-25
Pipeline Components.....	4-26
RefreshOrderHistoryPC	4-26
RefreshPaymentHistoryPC.....	4-27

Index



About This Document

This document explains how to use the services available within BEA WebLogic Portal™ Registering Customers and Managing Customer services.

This document includes the following topics:

- Chapter 1, “Overview of Registering Customers and Managing Customer Services,” which describes the high-level architecture and provides introductory information about the services.
- Chapter 2, “Customer Registration and Login Services,” which describes the JSP templates, input processors, and Pipeline Components associated with the customer registration and login Web pages.
- Chapter 3, “Customer Profile Services,” which describes the JSP templates, input processors, and Pipeline Components associated with the customer profile Web pages.
- Chapter 4, “Customer Self-Service,” which describes the JSP templates, input processors, and Pipeline Components 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 Portal documentation Home page on the e-docs Web site, <http://e-docs.bea.com>, and 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 Portal 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 Commerce services documents contain information that is relevant to using the *Guide to Registering Customers and Managing Customer Services* and understanding how to customize or extend the provided services. (The links are to locally installed documentation; if you did not install documentation locally, you can locate this documentation on the e-docs Web site.)

- *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*
- *Guide to Managing Purchases and Processing Orders*
- *Guide to Building a Product Catalog*
- For more information about J2EE as it relates to WebLogic Server security, see the information posted on the Sun Microsystems, Inc. Java™ 2 Platform, Enterprise Edition Web site at <http://java.sun.com/j2ee/>.

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Portal 4.0 release.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support through BEA WebSupport at <http://www.bea.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.
<i>italics</i>	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. <i>Examples:</i> #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. <i>Example:</i> void commit ()

Convention	Item
<i>monospace</i> <i>italic</i> text	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> ■ 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.



1 Overview of Registering Customers and Managing Customer Services

The processes related to customer profiles and self-service are necessary components of any e-business expecting return customers. To help you get to market faster than your competitors, WebLogic Portal provides you with the Registering Customers and Managing Customer services. 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). The registration and customer JSP templates also allow your site designers to customize these processes, without the need for advanced programming skills. This topic provides you with some background information about the Registering Customers and Managing Customer services, and introduces you to the types of services that are available.

This topic includes the following sections:

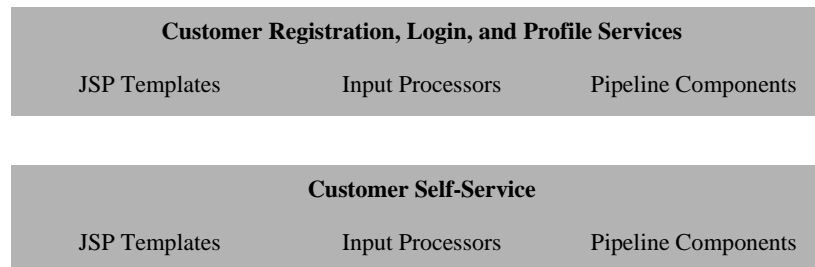
- What Are the Registering Customers and Managing Customer Services?
- High-level Architecture
- Template Quick Reference
- About the Database Schema
- Development Roles
- Next Steps

What Are the Registering Customers and Managing Customer Services?

The Registering Customers and Managing Customer services are a collection of JSP templates used to facilitate the registration of customers with your e-business site and the activities customers 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 JSP templates for the Registering Customers and Managing Customer services.

Figure 1-1 Registering Customers and Managing Customer services



Because all of the business logic is managed by Pipeline Processor and accessed within a Pipeline Processor session, the state of your customers' experiences can be maintained. For detailed information, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

In addition to the templates available in the Registering Customers and Managing Customer services, the Commerce services also contain templates for browsing the product catalog and for order and purchase processing. For information on services

related to the product catalog, see the *Guide to Building a Product Catalog*. For information on services related to order and purchase processing, see the *Guide to Managing Purchases and Processing Orders*.

High-level Architecture

The Registering Customers and Managing Customer services is essentially an application that utilizes the Webflow 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 work together in the default Webflow. It is also important that you understand how this package works in conjunction with the order and purchase processing JSP templates described in the *Guide to Managing Purchases and Processing Orders*.

- For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.
- For more information about order and purchase processing, see the *Guide to Managing Purchases and Processing Orders*.

Figure 1-2 shows the ways in which a customer might move through the JSP templates in the login and registration portion of the Registering Customers and Managing Customer services. It also shows how the customer-processing services, the product catalog services, and the order and purchase processing services are used.

Note: The shopping cart management piece of the Webflow is not discussed in this document. For more information about the shopping cart and the checkout process, see “Shopping Cart Management Services” in the *Guide to Managing Purchases and Processing Orders*.

1 Overview of Registering Customers and Managing Customer Services

Figure 1-2 Default Webflow for Login/Registration

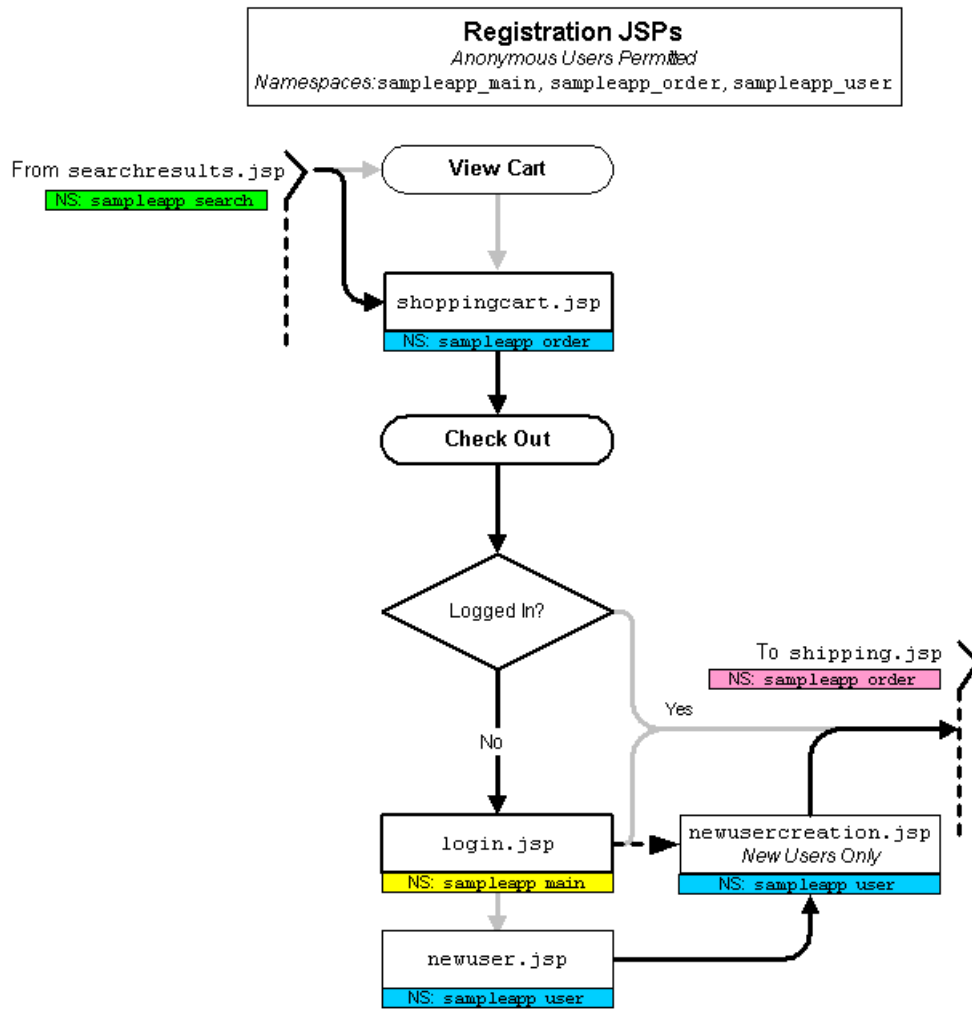
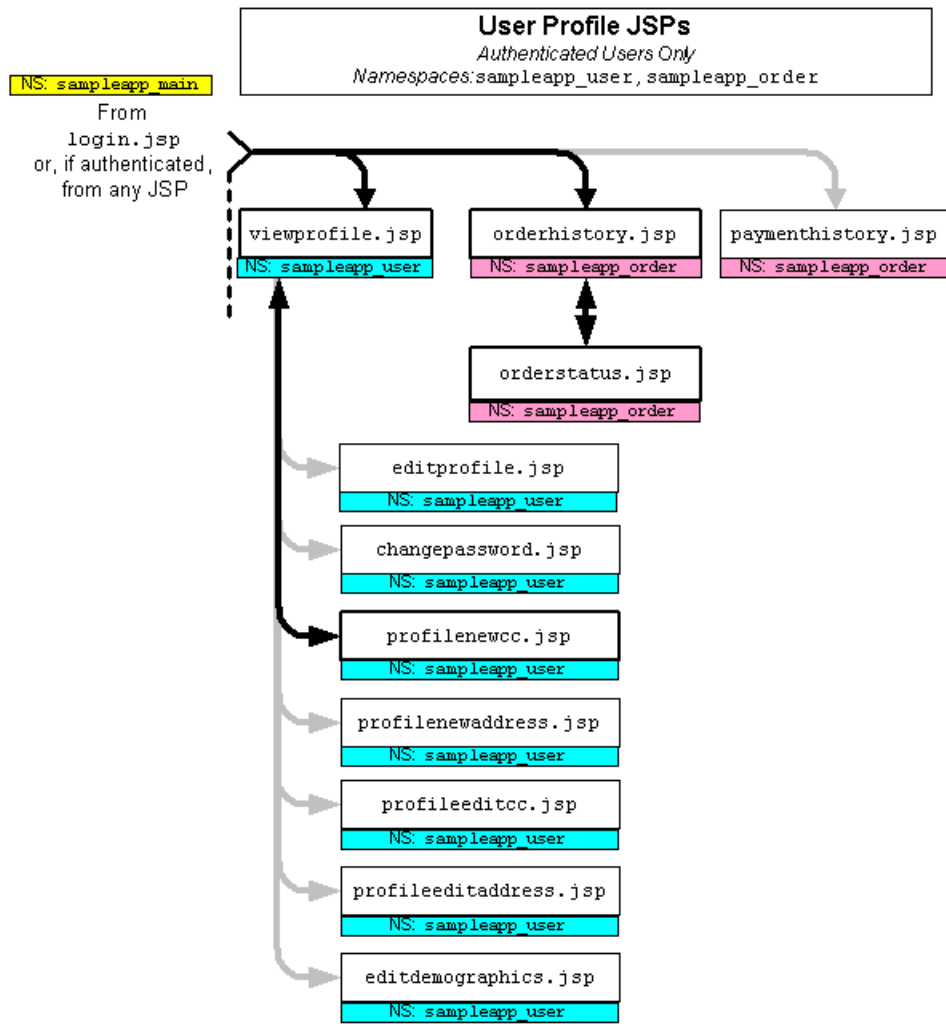


Figure 1-3 shows the ways in which your customer might move through the JSP templates in the customer-processing and customer self-service portions of the Registering Customers and Managing Customer services. It also shows how the product catalog services, the order and purchase processing services, and the login/registration JSPs are used.

Figure 1-3 Default Webflow for customer 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.

1 Overview of Registering Customers and Managing Customer Services

Whether you are customizing or extending this architecture, everything you need to know about the Registering Customers and Managing Customer services is provided in this document.

Template Quick Reference

The JSP templates in this guide are shown in the following table.

Table 1-1 Services JSP Quick Reference

Template Name	Location	Description
login.jsp (see page 2-3)	PORTAL_HOME\applications\wlcs App\wlcs\commerce	Provides form-based submission of username and password to gain access to account, and a link to create a new user.
badlogin.jsp (see page 2-8)	PORTAL_HOME\applications\wlcs App\wlcs\commerce	Similar to login.jsp, displays an error message (includes login.jsp).
newuser.jsp (see page 2-11)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\register	Empty profile fields for all personal information. Also includes a demographic survey to record user profile information.
newusercreation.jsp (see page 2-25)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	Displayed after the user registers successfully - provides links to view cart, check out, and return home.
newuserforward.jsp (see page 2-31)	PORTAL_HOME\applications\wlcs App\wlcs\commerce	Redirects to the user/newusercreation.jsp.
usercreationforward.jsp (see page 2-33)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	Used to redirect the newusercreation.jsp page after creating the user. This is done because the request does not contain the user information unless you do a redirect via the response object to the proper URL. This usercreationforward.jsp creates a new request that has the authenticated user's information, which allows the campaigns to start.

Table 1-1 Services JSP Quick Reference (Continued)

Template Name	Location	Description
viewprofile.jsp (see page 3-2)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	Presents the known personal information for the logged-in user, with buttons for modifying and adding information.
editprofile.jsp (see page 3-12)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	Change personal info (name, billing address, phones, email).
profilenewaddress.jsp (see page 3-21)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	User may supply an additional new shipping address; includes newaddresstemplate.inc.
profileeditaddress.jsp (see page 3-27)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	User may modify an existing shipping address in profile; includes editaddresstemplate.inc.
profilenewcc.jsp (see page 3-35)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	User may supply new credit account info - includes newcctemplate.inc.
profileeditcc.jsp (see page 3-41)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	User may change the credit account info as selected in payment.jsp - includes editcctemplate.inc.
changepassword.jsp (see page 3-51)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	User may change their password.
editdemographics.jsp (see page 3-57)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\user	Allows registered user to change demographic information that was entered previously.
orderhistory.jsp (see page 4-4)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\order	User may view a summary of past or pending orders; may select one to view its contents, shipping dates, and other details.
orderstatus.jsp (see page 4-10)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\order	Individual orders are displayed with their items, costs, status, shipping address and other details.
paymenthistory.jsp (see page 4-18)	PORTAL_HOME\applications\wlcs App\wlcs\commerce\order	User may view a summary of past orders organized by payment.

About the Database Schema

The database schema used for Registering Customers and Managing Customer services is the one used for the BEA personalization services. For more information about this database schema, see the “WebLogic Personalization Server Database Schema” in the *Guide to Building Personalized Applications*. Additionally, customer profiles in the BEA Commerce services are implemented as Unified User Profiles (UUP). For more information about UUP, see “Creating and Managing Users” in the *Guide to Building Personalized Applications*.

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 Campaign and Commerce services 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 Registering Customers and Managing Customer services 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”

1 *Overview of Registering Customers and Managing Customer Services*

2 Customer Registration and Login Services

For customers who plan on frequenting your e-business, 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 customers 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
 - newuser.jsp Template
 - newusercreation.jsp Template
 - newuserforward.jsp Template
 - usercreationforward.jsp Template
- Input Processors
 - CustomerProfileIP
 - LoginCustomerIP
 - Pipeline Components
 - RegisterUserPC
 - EncryptCreditCardPC

JavaServer Pages (JSPs)

The Registering Customers and Managing Customer services contain a number of JavaServer Pages (JSPs) that handle customer registration (initial customer profile creation) and customer login. 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.

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 black lines and callout text are not part of the template; they are explanations of the template components.

Figure 2-1 Annotated login.jsp Template

The screenshot shows a web page with the following annotated components:

- Commerce Templates header:** A black navigation bar with the BEA logo, "About Current Template: login.jsp", "Template Index", "Administration", and a red "Commerce Templates" button.
- Page header:** A purple banner with "Your Logo Here" and a "Storage Boxes!" advertisement.
- Left column:** A vertical sidebar with a "See Our Large Selection of Saws Here!" advertisement and "Catalog data provided courtesy of TPA Register, where supply meets demand."
- Log In section:** A central form titled "Log In" for "Registered User" with fields for "Username" (containing "democustomer") and "Password" (containing "democustomer"), and a "Log In" button.
- New Customer section:** A section titled "New Customer" with a "Create" button and the text "Click Create to begin a new user profile."
- Footer:** A purple footer with "Copyright © 1999-2001, BEA Systems Inc." and a "Built On BEA" logo.

Callout boxes provide the following explanations:

- "The Commerce Templates header (admin.inc) contains useful information for the benefit of you and your development team."
- "The page header is created by importing the header.inc template."
- "The left column is created by importing the leftside.inc template."
- "This region provides two form fields for customers who already have a username and password combination."
- "This region provides a link to the newuser.jsp template that allows new customers to register with your e-commerce site."
- "The footer is created by importing the footer.inc template."

2 Customer Registration and Login Services

Location in the Directory Structure

You can find the `login.jsp` template file at the following location, where `$WL_PORTAL_HOME` is the directory in which you installed Commerce services:

`$WL_PORTAL_HOME\applications\wlcsApp\wlcs\commerce\login.jsp`
(Windows)

`$WL_PORTAL_HOME/applications/wlcsApp/wlcs/
commerce/login.jsp` (UNIX)

Tag Library Imports

The `login.jsp` template does not use any custom 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.*" %>
```

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. It is part of the `sampleapp_main` namespace.

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, the `badlogin.jsp` template is loaded.

If the customer is not yet registered and clicks on the Create button, the next page loaded allows the customer to create a profile and a username/password combination (`newuser.jsp`). After the customer has registered, the customer is automatically logged in and forwarded to the `newusercreation.jsp` template, which allows

customers to continue shopping, view their shopping carts, or check out. If the auto-login is unsuccessful, the `login.jsp` template is loaded for the customer to enter their username and password. If the customer's login attempt is unsuccessful, the `badlogin.jsp` is loaded.

Notes: The option to proceed to checkout is only provided on the `newusercreation.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” in the *Guide to Building a Product Catalog*.

For a detailed description of the `shoppingcart.jsp` and `shipping.jsp` templates, see “Shopping Cart Management Services” or “Shipping Services” in the *Guide to Managing Purchases and Processing Orders*.

For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by the Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by the Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```

2 Customer Registration and Login Services

- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by the Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

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)
<code>button.createUser</code>	No business logic required. Loads <code>newuser.jsp</code> .

Note: The Login button 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 6.1 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.
"namespace"	Hidden	The namespace for the JSP; sampleapp_main in this JSP.
"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 `<%= HttpServletRequestConstants.USER_NAME %>`) for use in the JSP.

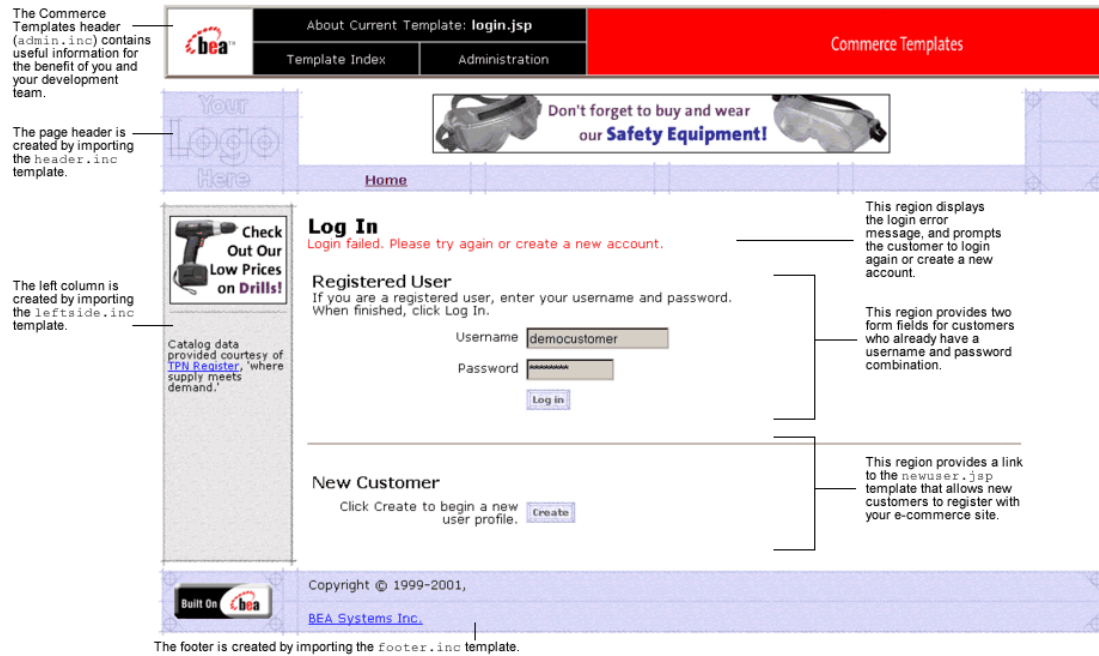
badlogin.jsp Template

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 black lines and callout text are not part of the template; they are explanations of the template components.

Figure 2-2 Annotated `badlogin.jsp` Template



Location in the Directory Structure

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\badlogin.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/  
commerce/badlogin.jsp (UNIX)
```

Tag Library Imports

The `badlogin.jsp` template does not use any custom 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 allows the customer to create a profile and obtain a username/password combination (`newuser.jsp`). After the customer has registered, the customer is automatically logged in and forwarded to the `newusercreation.jsp` template, which allows customers to continue shopping, view their shopping carts, or check out. If the auto-login is unsuccessful, the `login.jsp` template is loaded for the customer to enter their username and password. If the customer's login attempt is unsuccessful, the `badlogin.jsp` is loaded.

This template is part of the `sampleapp_main` namespace in the Webflow.

2 Customer Registration and Login Services

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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 includes the `login.jsp` template, the `badlogin.jsp` template uses the same events. For more information about these events, see “Login.jsp Template” on page 2-3.

Dynamic Data Display

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

Form Field Specification

Because the `badlogin.jsp` template includes the `login.jsp` template, the `badlogin.jsp` template uses the same form fields. For more information about these form fields, see “Login.jsp Template” on page 2-3.

newuser.jsp Template

The `newuser.jsp` template (shown in Figure 2-3 through Figure 2-6) 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 2-3 through Figure 2-6 show annotated versions of the `newuser.jsp` template. The black lines and callout text are explanations of the template components.

Figure 2-3 Annotated newuser.jsp Template - Personal Information

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

This registration form populates the Unified Customer Profile (UCP). Through the Campaign Manager you can use this information to create rules that target content based upon specific user profiles. [Register now](#) to see for yourself.

Catalog data provided courtesy of [Tech Register](#), where supply meets demand.

This region provides form fields for customers to enter their personal customer profile information, including their name, address, phone number(s), and email address.

Register Now and save \$10 on your order of at least \$50 placed today!!

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 / Province (Required for U.S. and Canadian address)

Zip/Postal Code *

Country *

Home phone

Business phone

Email address *

Yes, I want to be offered specials and notified of discounts. Please send me promotional e-mail.

2 Customer Registration and Login Services

Figure 2-4 Annotated newuser.jsp Template - Demographic Information

This Demographic Information region provides radio buttons/form fields for customers to provide demographic information that can interface with the BEA E-Business Control Center, allowing a business analyst to define, regulate, and personalize campaigns for each customer that visits your site.

Demographic Options	
Gender *	<input type="radio"/> Female <input type="radio"/> Male
Date of Birth *	<input type="text"/> (mm/dd/yyyy)
Occupation *	<input type="radio"/> Clerical <input type="radio"/> Executive Management <input type="radio"/> Professional <input type="radio"/> Engineering <input type="radio"/> Management <input type="radio"/> Sales
Employment Status *	<input type="radio"/> Not employed, not looking for work <input type="radio"/> Self-employed <input type="radio"/> Employed <input type="radio"/> Not employed, looking for work
Marital Status *	<input type="radio"/> Widowed <input type="radio"/> Married <input type="radio"/> Single <input type="radio"/> Divorced
Education Level *	<input type="radio"/> High School <input type="radio"/> College Graduate <input type="radio"/> Graduate Degree <input type="radio"/> Professional Degree <input type="radio"/> Some College
Income Range *	<input type="radio"/> Under \$35,000 <input type="radio"/> \$35,000 to \$49,999 <input type="radio"/> \$50,000 to \$74,999 <input type="radio"/> \$75,000 to \$99,999 <input type="radio"/> \$100,000 to \$124,999 <input type="radio"/> \$125,000 and above
Handiness *	<input type="radio"/> Do It Yourself <input type="radio"/> All thumbs <input type="radio"/> Saturday Helper <input type="radio"/> Professional

(Or)

Figure 2-5 Annotated newuser.jsp Template - Shipping Address and Payment Information (Optional)

The Shipping Address region provides form fields for customers to enter a shipping address. The Same as Above checkbox allows customers to use the address they provided in the personal information section as their shipping address.

mail.

Shipping Address

Same as above

Address *

Address 2

City *

State / Province (Required for U.S. and Canadian addresses)

Zip/Postal Code *

Country *

Demographic Options

Figure 2-6 Annotated newuser.jsp Template - Account Information

The Payment Information region provides form fields for credit card information. The information requested in this region is optional. The Account Information region provides form fields for username and password.

Payment Information (Optional)

Credit card type *

Name on card *

Card number *

Expiration date (mm/yyyy) *

Card billing address *

Address 2

City *

State / Province (Required for U.S. and Canadian addresses)

Zip/Postal Code *

Country *

Username and Password

Username *


Password *

Repeat password to confirm *

Fields marked with (*) are required.

The Account Information region provides form fields for username and password.

The footer is created by importing the footer.inc template.

Built On  Copyright © 1999-2001, [BFA Systems Inc.](#)

2 Customer Registration and Login Services

Note: The maximum number of characters allowed for usernames and passwords is set to 50. 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 Input Processor.

Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\register\  
newuser.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/  
commerce/register/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 library:

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

Note: For more information about the Webflow JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

This file resides in the following directory for the Commerce services Web application:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.HttpRequestConstants" %>  
<%@ page import="com.beasys.commerce.webflow.tags.WebFlowTagConstants" %>
```


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 auto-logged in and forwarded to a welcome page where they can select from the various links to continue shopping or check out (`newusercreation.jsp`). If errors are found, the `newuser.jsp` is reloaded with an appropriate message next to the invalid form fields.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `states.inc`, which contains a list of states as part of an address. The state is only required for U.S. citizens. The import call is:

```
<%@ include file="/commerce/includes/states.inc" %>
```
- `countries.inc`, which contains a list of countries as part of an address. The import call is:

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

2 Customer Registration and Login Services

- `newaddresstemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates.
- `newdemographictemplate.inc`, which contains formatting for the demographic data.
- `newcctemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/footer.inc" %>`

About the Included `newaddresstemplate.inc` Template

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

The behavior described above is accomplished on the `newaddresstemplate.inc` template using the `getValidatedValue` JSP tag, as shown in Listing 2-1.

Listing 2-1 Use of the `getValidatedValue` JSP Tag on `newaddresstemplate.inc`

```
<!-- begin table with customer's shipping address information -->
<table width="90%" border="0">
  <tr>
    <td width="26%"><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 %>><b>Address </b>
    </font>
    </div>
  </td>
</tr>
</table>
```

```

</td>
<td width="74%"> <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 the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

About the Included `newcctemplate.inc` Template

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

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

Listing 2-2 Use of the `getValidatedValue` JSP Tag on `newcctemplate.inc`

```

<table>
.
.
.
  <td width="27%"><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 %>"><b>Name on card</b>

```

2 Customer Registration and Login Services

```
        </font>
    </div>
</td>
<td width="73%"> <input type="text"
    name="<%=HttpRequestConstants.CUSTOMER_CREDITCARD HOLDER%>"
    value="<%=customerCreditCardHolder%>" size="30" maxlength="50">*
</td>
.
.
.
</table>
```

Notes: For more information about the `getValidatedValue` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

About the Included `newdemographictemplate.inc` Template

The `newdemographictemplate.inc` template provides a standardized format for both the form presentation and error handling in all JSP templates that prompt customers for demographic information. The radio buttons are organized in a table, and upon form submission, the Input Processors associated with the `newdemographictemplate.inc` template will validate the form to ensure that all required fields contain values. If errors are detected, the `newdemographictemplate.inc` template will be redisplayed, with an error message at the top of the including page and the invalid field labels shown in a red (as opposed to the original black) font. Previously entered correct information will still be displayed in the form.

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

Listing 2-3 Use of the `getValidatedValue` JSP Tag on `newdemographictemplate.inc`

```
<webflow:getValidatedValue fieldName="<%=HttpRequestConstants.CUSTOMER_GENDER%>"
fieldDefaultValue="<%= (String)currentPropertyValue%>" fieldValue="genderValue"
fieldStatus="status" validColor="black" invalidColor="red"
unspecifiedColor="black" fieldColor="fontColor" />
    <td width="26%"><div class="tabletext"><b><font color=<%= fontColor %>>
Gender*</font></b></div>
    </td>
```

```

<td width="74%">
<!-- get the property values for Gender
propertyBean.setPropertyName(GENDER);
property = propertyBean.getPropertyObject();
if(property == null || property.getRestrictedValues() == null)
arr = new Object[0];
else arr = property.getRestrictedValues().toArray();%>
<ps:getRestrictedPropertyValues propertySetName="Demographics"
propertySetType="USER" propertyName="<%= GENDER %>" id="arr" result="foobar" />
<table width="100%" border="0" cellpadding="0"
cellspacing="0"><es:forEachInArray id="valueObject" array="<%= arr %>"
type="String">

<tr>
<td width="4%"><input type="radio" name="
<%= HttpRequestConstants.CUSTOMER_GENDER %"> value="<%= valueObject %">
<% if ( valueObject.equals(genderValue) ) { %>CHECKED<% } %>></td>
<td><%= valueObject %></td>
</tr>
</es:forEachInArray>

</table>

```

Note: For more information about the `getValidatedValue` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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 Component is invoked first. Table 2-3 provides information about these events and the business logic they invoke.

Table 2-3 newuser.jsp Events

Event	Webflow Response(s)
button.cancel	GetCategoryIP GetTopCategories Pipeline

2 Customer Registration and Login Services

Table 2-3 newuser.jsp Events

Event	Webflow Response(s)
button.save	CustomerProfileIP CustomerProfile Pipeline

Table 2-4 briefly describes each of the Pipelines from Table 2-3. For more information about individual Pipeline Components, see “Pipeline Components” on page 2-39.

Table 2-4 newuser.jsp Associated Pipelines

Pipeline	Description
CustomerProfile	Contains EncryptedCreditCardPC 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, most of which are imported from other templates, and a description for each of these form fields are listed in Table 2-5.

Note: If a form field is imported from another template, it is indicated in the description. Form fields without import information are in the `newuser.jsp` template.

Table 2-5 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 (<code>newuser.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpRequestConstants.CUSTOMER_FIRST_NAME</code>	Textbox	The customer's first name.
<code>HttpRequestConstants.CUSTOMER_MIDDLE_NAME</code>	Textbox	The customer's middle initial.
<code>HttpRequestConstants.CUSTOMER_LAST_NAME</code>	Textbox	The customer's last name.
<code>HttpRequestConstants.CUSTOMER_ADDRESS1</code>	Textbox	The first line in the customer's street address.
<code>HttpRequestConstants.CUSTOMER_ADDRESS2</code>	Textbox	The second line in the customer's street address.
<code>HttpRequestConstants.CUSTOMER_CITY</code>	Textbox	The city in the customer's address.
<code>HttpRequestConstants.CUSTOMER_STATE</code>	Listbox	The state in the customer's address. Imported from <code>states.inc</code> .
<code>HttpRequestConstants.CUSTOMER_ZIPCODE</code>	Textbox	The zip code in the customer's address.
<code>HttpRequestConstants.CUSTOMER_COUNTRY</code>	Listbox	The country in the customer's address. Imported from <code>countries.inc</code> .
<code>HttpRequestConstants.CUSTOMER_HOME_PHONE</code>	Textbox	The customer's home phone number.
<code>HttpRequestConstants.CUSTOMER_BUSINESS_PHONE</code>	Textbox	The customer's business phone number.

2 Customer Registration and Login Services

Table 2-5 newuser.jsp Form Fields (Continued)

Parameter Name	Type	Description
HttpRequestConstants. CUSTOMER_EMAIL	Textbox	The customer's e-mail address.
HttpRequestConstants. CUSTOMER_EMAIL_OPT_IN	Checkbox	Indicates that the customer wants to receive promotional items via e-mail.
HttpRequestConstants. SAME_AS_ABOVE	Checkbox	Indicates that the customer's shipping address is the same as the contact address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_ADDRESS1	Textbox	The first line in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_ADDRESS2	Textbox	The second line in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_CITY	Textbox	The city in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_STATE	Listbox	The state in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_ZIPCODE	Textbox	The zip/postal code in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_SHIPPING_COUNTRY	Listbox	The country in the customer's shipping address. Imported from <code>newaddresstemplate.inc</code> .
HttpRequestConstants. CUSTOMER_GENDER	Radio buttons	Identifies the customer as male or female. Imported from <code>newdemographictemplate.inc</code> .
HttpRequestConstants. CUSTOMER_DATE_OF_BIRTH	Textboxes	The customer's date of birth. Imported from <code>newdemographictemplate.inc</code> .
HttpRequestConstants. CUSTOMER_OCCUPATION	Radio buttons	The customer's job description. Imported from <code>newdemographictemplate.inc</code> .

Table 2-5 newuser.jsp Form Fields (Continued)

Parameter Name	Type	Description
HttpRequestConstants. CUSTOMER_EMPLOYMENT_STATUS	Radio buttons	Identifies if the customer has a job at the time of registration. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_MARITAL_STATUS	Radio buttons	Identifies the customer's marital status. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_EDUCATION_LEVEL	Radio buttons	Identifies how much formal education the customer has completed. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_INCOME_RANGE	Radio buttons	Identifies the customer's yearly income. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_QUALITY	Radio buttons	Ranks customer from beginner to expert in using your product. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_TYPE	Listbox	The type of the customer's credit card. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD HOLDER	Textbox	The name on the credit card. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_NUMBER	Textbox	The number of the customer's credit card. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_MONTH	Listbox	The month of the customer's credit card expiration date. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_YEAR	Listbox	The year of the customer's credit card expiration date. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_ADDRESS1	Textbox	The first line in the customer's billing address. Imported from newcctemplate.inc.
HttpRequestConstants. CUSTOMER_CREDITCARD_ ADDRESS2	Textbox	The second line in the customer's billing address. Imported from newcctemplate.inc.

2 Customer Registration and Login Services

Table 2-5 newuser.jsp Form Fields (Continued)

Parameter Name	Type	Description
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_CITY</code>	Textbox	The city in the customer's billing address. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_STATE</code>	Listbox	The state in the customer's billing address. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE</code>	Textbox	The zip/postal code in the customer's billing address. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY</code>	Listbox	The country in the customer's billing address. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.USER_NAME</code>	Textbox	An identity chosen by the customer for login.
<code>HttpRequestConstants.PASSWORD</code>	Password	A password chosen by the customer for login.
<code>HttpRequestConstants.CONFIRM_PASSWORD</code>	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.

newusercreation.jsp Template

The `newusercreation.jsp` template (shown in Figure 2-7) informs a customer who has just created a new user profile that they have been logged in and that registration was successful. It also provides the customer with the opportunity to return to their shopping experience through several navigation options.

Sample Browser View

Figure 2-7 shows an annotated version of the `newusercreation.jsp` template. The black lines and callout text are not part of the template; they are explanations of the template components.

Figure 2-7 Annotated `newusercreation.jsp` Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

The main body of the page indicates to the customer that registration and auto-login was successful. It also provides the customer with links to their shopping cart (`shoppingcart.jsp`), to continue shopping, to the checkout process (`shipping.jsp`), or to the main `.jsp` via the home link in the `header.inc`.

Now that the user has a profile and is logged in, the `leftside.inc` shows links to the customer's order history (`orderhistory.jsp`), payment history (`paymenthistory.jsp`), to their profile (`viewprofile.jsp`), or to logout.

The footer is created by importing the `footer.inc` template.

The screenshot shows a browser view of the `newusercreation.jsp` template. At the top, there is a red header with the BEA logo and navigation links for 'Template Index' and 'Administration'. Below the header is a blue navigation bar with 'Home' and a promotional message: 'You are now in a campaign that uses both your classification as a new user and your gender to target a specific message to you (notice the ad below). Store data indicates that women buy 75% of all Saws and Men buy 63% of all Drills. Click the ad below.' The main content area features a 'Congratulations' message: 'Welcome Demo220, you have successfully registered and logged in.' Below this, there are three buttons: 'View shopping cart', 'Check out', and 'Continue shopping'. A left sidebar contains a 'Welcome Demo Customer2' section with links for 'View Profile', 'Logout', 'View History', 'Orders', and 'Payments'. Below the sidebar is an advertisement for 'Buy a Saw Today & Redeem Your \$10!'. The footer includes the BEA logo, copyright information for 1999-2001, and the text 'BEA Systems Inc.'.

2 Customer Registration and Login Services

Notes: For a detailed description of the `main.jsp` template, see “Product Catalog JSP Templates” in the *Guide to Building a Product Catalog* book. For a detailed description of the `shoppingcart.jsp` and `shipping.jsp` templates, see “Shopping Cart Management Services” or “Shipping Services” in the *Guide to Managing Purchases and Processing Orders* book.

The option to proceed to checkout is only provided on the `newusercreation.jsp` template if there are items in the customer’s shopping cart. Otherwise, the `newusercreation.jsp` template will leave out this option as shown in Figure 2-8.

Figure 2-8 `newusercreation.jsp` - Without Checkout Option



Location in the Directory Structure

You can find the `newusercreation.jsp` template file at the following location, where `PORTAL_HOME` is the directory in which you installed WebLogic Portal:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\  
newusercreation.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/commerce/user/  
newusercreation.jsp (UNIX)
```

Tag Library Imports

The `newusercreation.jsp` template uses Pipeline Processor JSP tags. Therefore, the template includes the following JSP tag library:

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

Note: For more information about the Pipeline Processor JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

This file resides in the following directory for the Commerce services Web application:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF (UNIX)
```

Java Package Imports

The `newusercreation.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.PipelineSessionConstants" %>
```

Location in Default Webflow

Customers arrive at the `newusercreation.jsp` template when they have successfully created a new user profile and the auto-login using JAAS (Java Authentication and Authorization Service) has completed. If the customer creates a new profile, but the auto-login does not complete successfully, the customer is routed to the `login.jsp` template and will not see the `newusercreation.jsp` template. After manual login, the customer is routed to the `main.jsp` template.

2 Customer Registration and Login Services

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 `newusercreation.jsp` template, the customer can return to their shopping cart (`shoppingcart.jsp`), continue shopping, continue to the checkout process (`shipping.jsp`), view their order history (`orderhistory.jsp`), view their profile (`viewprofile.jsp`), view their payment history (`paymenthistory.jsp`), logout, or return to the main catalog page (`main.jsp`).

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

For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

This template is part of the `sampleapp_user` namespace in the Webflow.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

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 2-6 provides information about these events and the business logic they invoke.

Table 2-6 newusercreation.jsp Events

Event	Webflow Response(s)
link.shoppingcart	InitShoppingCartIP
button.checkout	InitShippingMethodListIP
link.home	GetTopCategoriesIP GetTopCategories Pipeline

Note: For more information about the `GetTopCategoriesIP` and `GetTopCategories Pipeline`, see the *Guide to Building a Product Catalog*.

Dynamic Data Display

One purpose of the `newusercreation.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 `newusercreation.jsp` using a combination of Pipeline Processor JSP tags and accessor methods/attributes.

First, the `getProperty` JSP tag retrieves the `SHOPPING_CART` attribute from the Pipeline Processor session. Table 2-7 provides more detailed information on this attribute.

Table 2-7 newusercreation.jsp Pipeline Processor Session Properties

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

2 Customer Registration and Login Services

Listing 2-4 illustrates how this attribute is retrieved from the Pipeline Processor session using the `getProperty` JSP tag.

Listing 2-4 Retrieving the Shopping Cart Attribute

```
<webflow:getProperty id="shoppingCart"
property="<%=PipelineSessionConstants.SHOPPING_CART%"
type="com.beasys.commerce.ebusiness.shoppingcart.ShoppingCart"
scope="session" namespace="sampleapp_main" />
```

Note: For more information on the `getProperty` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

The data stored within the `Pipeline` session attribute is then accessed by using accessor methods/attributes within Java scriptlets. Table 2-8 provides more detailed information about these methods/attributes for `shoppingCart`.

Table 2-8 `shoppingCart` Accessor Methods/Attributes

Method/Attribute	Description
<code>isEmpty()</code>	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 2-5.

Listing 2-5 Accessor Methods/Attributes in `newusercreation.jsp` Java Scriptlets

```
<% if (shoppingCart != null && shoppingCart.isEmpty() == false) { %>
<a href="<webflow:createWebflowURL event="button.checkout"
namespace="sampleapp_order" />">" border="0" vspace="2"
hspace="3"></a> &nbsp; &nbsp; &nbsp;
<% } %>
```

Form Field Specification

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

newuserforward.jsp Template

The `newuserforward.jsp` template is used to direct unregistered users to the `newuser.jsp` because dynamic URIs are not supported in placeholders. This page is accessed when an unregistered user clicks an ad placeholder that contains a static URI. The `newuserforward.jsp` template then forwards the user to `newuser.jsp`. Additionally, the `newuserforward.jsp` bridges the transition from a non-secure to a secure connection (http to https).

Because this page is never seen by the end user and uses no includes, instead of a figure, the code for this page is shown in Listing 2-6.

Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\
commerce\newuserforward.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/
commerce/newuserforward.jsp (UNIX)
```

Tag Library Imports

The `newuserforward.jsp` template does not use any tag libraries.

Java Package Imports

The `newuserforward.jsp` template does not use any imports.

Location in Default Webflow

The page prior to `newuserforward.jsp` can be any page that an anonymous user can access. However, this template is only needed if an unregistered user clicks the ad placeholder that prompts them to register. The static URI in the placeholder accesses the `newuserforward.jsp` which then forwards the user to the `newuser.jsp` template.

2 Customer Registration and Login Services

This template is part of the `sampleapp_main` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

No JSP templates are included in the `newuserforward.jsp` template.

Events

The `newuserforward.jsp` template has one event. This event triggers 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 2-9 provides information about this event and the business logic it invokes.

Table 2-9 `newuserforward.jsp` Events

Event	Webflow Response(s)
<code>button.createUser</code>	<code>newuser.jsp</code>

Listing 2-6 `newuserforward.jsp` Code

```
<% String s =
com.bea.p13n.appflow.webflow.WebflowJSPHelper.createWebflowURL(pageContext,
"sampleapp_main", "login.jsp", "button.createUser", true); %>

<% response.sendRedirect(s) ; %>
```

Dynamic Data Display

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

Form Field Specification

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

usercreationforward.jsp Template

The `usercreationforward.jsp` template is used to forward new users to the `newusercreation.jsp` template after the registration and auto-login process using JAAS is completed by the Webflow. Once the user is created, the request must be flushed and the `usercreationforward.jsp` template allows that to happen.

Because this page is never seen by the end user and uses no includes, instead of a figure, the code for this page is shown in Listing 2-7.

Location in the Directory Structure

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\user\usercreationforward.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/  
commerce/user/usercreationforward.jsp (UNIX)
```

Tag Library Imports

The `usercreationforward.jsp` template does not use any tag libraries.

Java Package Imports

The `usercreationforward.jsp` template uses Java classes in the following package and therefore includes the import statement:

```
<%@ page import="com.bea.p13n.appflow.webflow.WebflowJSPHelper*" %>
```

Location in Default Webflow

The page prior to `usercreationforward.jsp` is the `newuser.jsp` template. When new users save their profiles, they are auto-logged in using JAAS and if the login is successful, because the old request must be flushed, the `usercreationforward.jsp` is needed to redirect the user to the `newusercreation.jsp` template.

2 Customer Registration and Login Services

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

No JSP templates are included in the `usercreationforward.jsp` template.

Events

The `usercreationforward.jsp` template has one event. This event triggers 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 2-10 provides information about this event and the business logic it invokes.

Table 2-10 `usercreationforward.jsp` Events

Event	Webflow Response(s)
<code>forward.usercreation</code>	<code>newusercreation.jsp</code>

Listing 2-7 `usercreationforward.jsp` Code

```
<% String s = WebflowJSPHelper.createWebflowURL(pageContext, "sampleapp_user",  
"usercreationforward.jsp", "forward.usercreation", true); %>  
  
<% response.sendRedirect(s) ; %>
```

Dynamic Data Display

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

Form Field Specification

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

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 the *Guide to Building a Product Catalog*.

CustomerProfileIP

Class Name	<code>examples.wlcs.sampleapp.customer.webflow.CustomerProfileIP</code>
Description	Processes the input of <code>newuser.jsp</code> and allows the customer to store their profile. Creates and places a <code>CustomerValue</code> object into the Pipeline Processor session.
Required HttpServletRequest Parameters (Personal Information)	<code>HttpRequestConstants.CUSTOMER_FIRST_NAME</code> <code>HttpRequestConstants.CUSTOMER_MIDDLE_NAME</code> <code>HttpRequestConstants.CUSTOMER_LAST_NAME</code> <code>HttpRequestConstants.CUSTOMER_ADDRESS1</code> <code>HttpRequestConstants.CUSTOMER_ADDRESS2</code> <code>HttpRequestConstants.CUSTOMER_CITY</code> <code>HttpRequestConstants.CUSTOMER_STATE</code> <code>HttpRequestConstants.CUSTOMER_ZIPCODE</code> <code>HttpRequestConstants.CUSTOMER_COUNTRY</code> <code>HttpRequestConstants.CUSTOMER_HOME_PHONE</code> <code>HttpRequestConstants.CUSTOMER_BUSINESS_PHONE</code> <code>HttpRequestConstants.CUSTOMER_EMAIL</code> <code>HttpRequestConstants.CUSTOMER_EMAIL_OPT_IN</code> (code location: <code>newuser.jsp</code> template.)

2 Customer Registration and Login Services

Required HttpServletRequest Parameters (Demographic Information)	HttpRequestConstants.CUSTOMER_INCOME_RANGE HttpRequestConstants.CUSTOMER_EDUCATION_LEVEL HttpRequestConstants.CUSTOMER_DATE_OF_BIRTH HttpRequestConstants.CUSTOMER_GENDER HttpRequestConstants.CUSTOMER_OCCUPATION HttpRequestConstants.CUSTOMER_MARITAL_STATUS HttpRequestConstants.CUSTOMER_EMPLOYMENT_STATUS HttpRequestConstants.CUSTOMER_QUALITY (code location: newdemographictemplate.inc template.)
Required HttpServletRequest Parameters (Shipping Information)	HttpRequestConstants.SAME_AS_ABOVE (code location: newuser.jsp template.) 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 (code location: newaddresstemplate.inc template.)
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 (code location: newcctemplate.inc template.)
Required HttpServletRequest Parameters (Account Information)	HttpRequestConstants.USER_NAME HttpRequestConstants.PASSWORD HttpRequestConstants.CONFIRM_PASSWORD (code location: newuser.jsp template.)

Required Pipeline Session properties	None
Updated Pipeline Session properties	PipelineSessionConstants.CUSTOMER PipelineSessionConstants.PASSWORD PipelineSessionConstants.CREDITCARD_KEY (only if customer provides a credit card update).
Removed Pipeline Session properties	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).

LoginCustomerIP

Class Name	examples.wlcs.sampleapp.customer.webflow.LoginCustomerIP
Description	Processes the input of login.jsp and allows the customer to access the secure pages of the site. Creates and places a CustomerValue object into the Pipeline Processor session.
Required HttpServletRequest Parameters	None
Required Pipeline Session properties	PipelineSessionConstants.CUSTOMER PipelineSessionConstants.PASSWORD PipelineSessionConstants.CREDITCARD_KEY (only if the customer provides a credit card update).

2 Customer Registration and Login Services

Updated Pipeline Session properties	None
Removed Pipeline Session properties	<code>PipelineSessionConstants.PASSWORD</code>
Validation	Verifies that the username and password are correct.
Exceptions	<code>InvalidInputException</code> , thrown if either the username or password is invalid. <code>ProcessingException</code> , thrown if the username is invalid or cannot get authentication.

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	examples.wlcs.sampleapp.customer.pipeline. RegisterUserPC
Description	Retrieves the CustomerValue object and password from the Pipeline Processor session, and creates a CUSTOMER attribute.
Contained in	CustomerProfile Pipeline
Required Pipeline Session Properties	PipelineSessionConstants.CUSTOMER PipelineSessionConstants.PASSWORD
Updated Pipeline Session Properties	None
Removed Pipeline Session Properties	PipelineSessionConstants.PASSWORD
Type	Java class
JNDI Name	None
Exceptions	PipelineException, thrown when the Pipeline Component cannot create the user.

EncryptCreditCardPC

Class Name	<code>examples.wlcs.sampleapp.customer.pipeline. EncryptCreditCardPC</code>
Description	Uses the <code>CREDITCARD_KEY</code> object to retrieve a customer credit card, encrypts the credit card number, and then adds the modified credit card back to the <code>PipelineSession CustomerValue</code> attribute.
Contained in	<code>CustomerProfile Pipeline</code>
Required Pipeline Session Properties	<code>PipelineSessionConstants.CREDITCARD_KEY</code>
Updated Pipeline Session Properties	<code>PipelineSessionConstants.CUSTOMER</code>
Removed Pipeline Session Properties	<code>PipelineSessionConstants.CREDITCARD_KEY</code>
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when the Pipeline Component cannot find the user in the Pipeline Processor session or the <code>creditcard_key</code> is invalid or the encryption did not complete successfully.

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 Registering Customers and Managing Customer services provide 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
 - editdemographics.jsp Template
- Input Processors
 - DeleteCreditCardIP
 - DeleteShippingAddressIP

- UpdateAccountInfoIP
- UpdateBasicInfoIP
- UpdateDemographicInfoIP
- UpdatePaymentInfoIP
- UpdateShippingInfoIP
- Pipeline Components
 - UpdateBasicInfoPC
 - UpdateDemographicInfoPC
 - UpdatePasswordPC
 - UpdatePaymentInfoPC
 - UpdateShippingInfoPC

JavaServer Pages (JSPs)

The Registering Customers and Managing Customer services contain 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 Commerce services Web application and a listing of their locations in the directory structure, see the “Summary of the JSP Templates” documentation.

viewprofile.jsp Template

The `viewprofile.jsp` template (shown in Figure 3-1) allows a registered customer to view his or her existing profile information. It displays the existing information in five categories: personal information, shipping addresses, credit cards, username and password, and demographic information. 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 black lines and callout text are explanations of the template components.

Figure 3-1 Annotated viewprofile.jsp Template

The Commerce Templates header (admin.inc) contains useful information for the benefit of you and your development team.

The page header is created by importing the header.inc template.

The left column is created by importing the leftside.inc template.

Catalog data provided courtesy of [TPN Reader](#), where supply meets demand.

The Payment Information region displays the customer's existing credit card information.

The Account Information region displays the customer's current username.

The footer is created by importing the footer.inc template.

BEA About Current Template: **viewprofile.jsp** Commerce Templates
 Template Index Administration

Your Logo Here

Store your stuff in our **Storage Boxes!**

Home

Welcome Demo Customer
[View Profile](#)
[Logout](#)

View History
[Orders](#)
[Payments](#)

See Our Large Selection of Saws Here!

Demo Customer's Profile

Contact Info [Update]

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

OK to E-mail? Yes, I want to be offered specials and notified of discounts. Please send me promotional e-mail.

The Personal Information region displays the customer's existing personal information using the editprofile.jsp template.

Shipping Addresses [Add address]

One Winthrop Square [Update] [Remove]
 Boston MA 02110
 United States

One Main Street [Update] [Remove]
 Denver CO 80212
 United States

The Shipping Address region displays any shipping addresses the customer stored as part of their customer profile.

Credit Cards [Add card]

Select an account on file to update or delete.

VISA-1111 [Update this card] [Remove]

User Name & Password [Update]

User Name democustomer

Demographic Data [Update]

Gender Female

Date of Birth October 8, 1948

Occupation Executive Management

Employment Status Employed

Marital Status Married

Education Level College Graduate

Income Range \$75,000 to \$99,999

Handiness Do It Yourselfer

The Demographic Information region displays the customer's demographic information using the editdemographics.jsp template.

Built On **BEA** Copyright © 1999-2001, BEA Systems Inc.

3 Customer Profile Services

Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\user\viewprofile.jsp (Windows)
```

```
%PORTAL_HOME/applications/wlcsApp/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="webflow.tld" prefix="webflow" %>  
<%@ 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 *Guide to Building Personalized Applications* documentation.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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="java.text.DateFormat" %>
```

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,
- `editdemographics.jsp`, which allows customers to edit their demographic information,
- `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 page is described in other sections of this document.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by the Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```


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)
<code>button.updateBasicInfo</code>	No business logic required. Loads <code>editprofile.jsp</code> .
<code>button.addNewShippingAddress</code>	No business logic required. Loads <code>profilenewaddress.jsp</code> .
<code>button.updateShippingInfo</code>	No business logic required. Loads <code>profileeditaddress.jsp</code> .
<code>button.deleteShippingAddress</code>	DeleteShippingAddressIP DeleteShippingAddressFromProfile Pipeline
<code>button.updateDemograhpicInfo</code>	No business logic required. Loads <code>editdemographics.jsp</code> .
<code>button.addNewCreditCard</code>	No business logic required. Loads <code>profilenewcc.jsp</code> .
<code>button.updatePaymentInfo</code>	No business logic required. Loads <code>profileeditcc.jsp</code> .
<code>button.deletePaymentInfo</code>	DeleteCreditCardIP DeleteCreditCard Pipeline
<code>button.changePassword</code>	No business logic required. Loads <code>changepassword.jsp</code> .

Table 3-2 briefly describes each of the Pipelines from Table 3-1. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

3 Customer Profile Services

Table 3-2 viewprofile.jsp Associated 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) in the session 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 User Management JSP tags, see "Personalization Server JSP Tag Library Reference" in the *Guide to Building Personalized Applications* 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 propertySet="CustomerProperties"  
propertyName="contactAddress" id="contactAddressObject" />
```

```

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

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

<%
// Convert contactAddressObject and shippingAddressMapObject to the
// correct types.
Address contactAddress = (Address) contactAddressObject;
Map shippingAddressMap = (Map) shippingAddressMapObject;
Map creditCardsMap = (Map) creditCardsMapObject;

// initialize shippingAddressMap
if(shippingAddressMap == null) {
    shippingAddressMap = new HashMap();    }
%>

```

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.

3 *Customer Profile Services*

Table 3-3 `contactAddress/shippingAddress` Accessor Methods/Attributes

Method/Attribute	Description
<code>getCountry()</code>	The country in the customer's contact or shipping address.

Table 3-4 `creditCard` Accessor Methods/Attributes

Method/Attribute	Description
<code>creditCard()</code>	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

```

<tr>
<td align="right" valign="top" width="5%"><div
class="tabletext"><b>Address</b></div></td>
<td width="5%">" width="5"
height="5"></td>
  <td align="left"><div class="tabletext">
    <%=contactAddress.getStreet1()%><br>
    <% if(contactAddress.getStreet2().length() != 0) { %>
    <%=contactAddress.getStreet2()%><br> <% } %>
    <%=contactAddress.getCity()%><br>
    <%=contactAddress.getState()%> &nbsp;    
    <%=contactAddress.getPostalCode()%><br>
    <%=contactAddress.getCountry()%></div>
  </td>
</tr>
.
.
.
<!-- Loop through all of the credit cards -->
<wl:repeat
set="<%=((Map)creditCardsMapObject).keySet().iterator()%>"
id="creditCard" type="String" count="100000">
<tr>
<!-- Output the credit card name -->
  <td width="55%"><div
  class="tabletext"><b><%=creditCard%></b></div><td>
<!-- The update button -->
<%
extraParams = HttpRequestConstants.CREDITCARD_KEY + "=" +
creditCard;
%>
  <td align="center">
    <a href="<webflow:createWebflowURL
    event="button.updatePaymentInfo" namespace="sampleapp_user"
    extraParams="<%= extraParams %>" />"><img src=
    "<webflow:createResourceURL
    resource="/commerce/images/btn_updatecard.gif" />"
    border="0"></a>

```

3 Customer Profile Services

```
</td>
.
.
.
</wl:repeat>
```

Notes: For more information on User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

The `getPropertyAsString` JSP tag is used to directly obtain the customer’s first and last name, the customer’s home and business phone numbers, the customer’s e-mail address, demographic information, and username and password. Listing 3-4 illustrates how to use the `getPropertyAsString` JSP tag to display the 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
propertySet="CustomerProperties" propertyName="firstName" />
<um:getPropertyAsString propertySet="CustomerProperties"
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 e-mail address.

Sample Browser View

Figure 3-2 shows an annotated version of the `editprofile.jsp` template. The black lines and callout text are not part of the template; they are explanations of the template components.

Figure 3-2 Annotated editprofile.jsp Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

The footer is created by importing the `footer.inc` template.

This region provides a series of form fields that allow customers to change their personal information.

BEA About Current Template: `editprofile.jsp` Commerce Templates

Template Index Administration

Your Logo Here

Click here to see our full line of powerful **Routers!**

Home

Welcome Demo Customer
[View Profile](#)
[Logout](#)

View History
[Orders](#)
[Payments](#)

Check Out Our Low Prices on Drills!

Catalog data provided courtesy of [TPH Register](#), where supply meets demand.

Edit Contact Information

First name: Demo *
 Middle initial: *
 Last name: Customer *
 Street address: One Winthrop Square *
 Address 2: *
 City: Boston *
 State / Province: MA (Required for U.S. and Canadian addresses)
 Zip/Postal Code: 00237 *
 Country: United States *
 Home phone: 617-555-5555
 Business phone: 708-555-5555
 Email address: democustomer@bea.co *

Yes, I want to be offered specials and notified of discounts. Please send me promotional e-mail.

Save
 < Back

Built On **BEA** Copyright © 1999-2001, BEA Systems Inc.

3 Customer Profile Services

Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\  
editprofile.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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="webflow" %>  
<%@ taglib uri="um.tld" prefix="um" %>  
<%@ taglib uri="es.tld" prefix="es" %>
```

Note: For more information on the WebLogic Server JSP tags or the User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.axiom.contact.*" %>  
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
```



```
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ page import="com.beasys.commerce.webflow.PipelineSessionConstants" %>
<%@ page import="java.util.*" %>
```

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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `states.inc`, which contains a list of states as part of an address. The state is only required for U.S. citizens. The import call is:

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

3 Customer Profile Services

- `countries.inc`, which contains a list of countries as part of an address. The import call is:
`<%@ include file="/commerce/includes/countries.inc" %>`
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/footer.inc" %>`

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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdateBasicInfoIP</code> <code>EditBasicInfo</code> Pipeline

Table 3-6 briefly describes each of the Pipelines from Table 3-5. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-6 `editprofile.jsp` Associated Pipelines

Pipeline	Description
<code>EditBasicInfo</code>	Contains <code>UpdateBasicInfoPC</code> 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 User Management JSP tags, and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) in the session 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 User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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 propertySet="CustomerProperties"
  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
<code>getStreet1()</code>	The first line in the customer’s contact street address.
<code>getStreet2()</code>	The second line in the customer’s contact street address.
<code>getCity()</code>	The city in the customer’s contact address.

3 Customer Profile Services

Table 3-7 contactAddress Accessor Methods/Attributes (Continued)

Method/Attribute	Description
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.

Notes: The `getPropertyAsString` JSP tag is used to obtain the customer's first and last name, the customer's home and business phone numbers, the customer's e-mail address, demographic information, and username and password.

The `getProperty` JSP tag is used to obtain a value from the `EMAIL_OPT_IN` attribute. This attribute designates if the customer wants to receive promotional items via e-mail.

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 propertySet="CustomerProperties"
propertyName="firstName" id="firstName" />
```

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

```
<tr>
  <um:getPropertyAsString propertySet="CustomerProperties"
  propertyName="lastName" id="lastName" />
  <td width="26%"><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 %>><b>Last name
</b></font></div>
  </td>
  <td width="74%">
    <input type="text" name="<%=HttpRequestConstants.CUSTOMER_LAST_NAME%>"
value="<%=customerLastName%>" maxlength="30"> * </td>
</tr>
<tr>
  <td width="26%"><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 %>><b>Street
address</b></font></div>
  </td>
  <td width="74%">
    <input type="text" name="<%=HttpRequestConstants.CUSTOMER_ADDRESS1%>"
value="<%=customerAddress1%>" maxlength="30">*</td>
</tr>

```

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.

3 Customer Profile Services

Table 3-8 editprofile.jsp Form Fields (Continued)

Parameter Name	Type	Description
“origin”	Hidden	The name of the current page (editprofile.jsp), used by the Webflow.
“namespace”	Hidden	The namespace for the JSP; sampleapp_user in this JSP.
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.
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 e-mail address.

Table 3-8 editprofile.jsp Form Fields (Continued)

Parameter Name	Type	Description
<code>HttpRequestConstants.CUSTOMER_EMAIL_OPT_IN</code>	Checkbox	Indicates that the customer wants to receive promotional materials via e-mail.

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 black lines and callout text are not part of the template; they are explanations of the template components.

3 Customer Profile Services

Figure 3-3 Annotated profilenewaddress.jsp Template



Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\profilenewaddress.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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" %>
```

Note: For more information about the Webflow and Pipeline JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.tags.WebFlowTagConstants" %>  
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>  
<%@ 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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

3 Customer Profile Services

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `newaddresstemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates. The template is described in “About the Included `newaddresstemplate.inc` Template” on page 2-16.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdateShippingInfoIP</code> <code>ProfileNewAddress Pipeline</code>

Table 3-10 briefly describes each of the Pipelines from Table 3-9. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-10 *profilenewaddress.jsp* Associated 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 imported from the `newaddresstemplate.jsp` file, and a description for each of these form fields are shown in Table 3-11.

3 Customer Profile Services

Note: If a form field is imported from another template, it is indicated in the description. Form fields without import information are in the `profilenewaddress.jsp` template.

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 (<code>profilenewaddress.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1</code>	Textbox	The first line in the customer's street address. Imported from <code>newaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2</code>	Textbox	The second line in the customer's street address. Imported from <code>newaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_CITY</code>	Textbox	The city in the customer's address. Imported from <code>newaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_STATE</code>	Listbox	The state in the customer's address. Imported from <code>newaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE</code>	Textbox	The zip code in the customer's address. Imported from <code>newaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY</code>	Listbox	The country in the customer's address. Imported from <code>newaddresstemplate.inc</code> .

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 black lines and callout text are not part of the template; they are explanations of the template components.

Figure 3-4 Annotated profileeditaddress.jsp Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

This region provides a series of form fields to update an existing shipping address.

The footer is created by importing the `footer.inc` template.

Update Profile

Street address:

Address 2:

City:

State / Province: (Required for U.S. and Canadian addresses)

Postal/Zip Code:

Country:

Location in Commerce services Directory Structure

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

3 Customer Profile Services

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\user\profileeditaddress.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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 User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*. For more information about the Webflow JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.tags.WebFlowTagConstants" %>  
<%@ page import="com.beasys.commerce.axiom.contact.*" %>  
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>  
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
```

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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `editadresstemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

3 Customer Profile Services

About the Included editaddressstemplate.inc Template

The `editaddressstemplate.inc` 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 `editaddressstemplate.inc` template will validate the form to ensure that all required fields contain values. If errors are detected, the `editaddressstemplate.inc` template will be redisplayed, with an error message at the top and the offending field labels shown in a red (as opposed to the original black) font. Previously entered correct information will still be displayed in the form.

Since the `editaddressstemplate.inc` 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 `editaddressstemplate.inc` 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 `editaddressstemplate.inc`

```
<tr>
  <td width="26%"> <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 width="74%">
  <input type="text" name="<%=HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1%">"
value="<%=customerShippingAddress1%">" maxlength="30">*</td>
</tr>
```

Notes: For more information about the `getValidatedValue` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdateShippingInfoIP</code> <code>ProfileEditAddress Pipeline</code>

Table 3-13 briefly describes each of the Pipelines from Table 3-12. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-13 `profileeditaddress.jsp` Associated Pipelines

Pipeline	Description
<code>ProfileEditAddress</code>	Contains <code>UpdateShippingInfoPC</code> 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 `editaddressstemplate.inc` 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 User Management JSP tags, and accessor methods/attributes.

3 Customer Profile Services

First, the `getProfile` JSP tag is used to set the customer profile (context) in the session 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 User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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 propertySet="CustomerProperties"
  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. In this scenario, the scriptlets are in the `editaddressstemplate.inc`. 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
<code>getStreet1()</code>	The first line in the customer's shipping street address.
<code>getStreet2()</code>	The second line in the customer's shipping street address.
<code>getCity()</code>	The city in the customer's shipping address.
<code>getCounty()</code>	The county in the customer's shipping address.
<code>getState()</code>	The state in the customer's shipping address.
<code>getPostalCode()</code>	The zip/postal code in the customer's shipping address.
<code>getCountry()</code>	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 imported from the `editaddressstemplate.inc` file, and a description for each of these form fields are listed in Table 3-15.

3 Customer Profile Services

Note: If a form field is imported from another template, it is indicated in the description. Form fields without import information are in the `profileeditaddress.jsp` template.

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 (<code>profileeditaddress.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1</code>	Textbox	The first line in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2</code>	Textbox	The second line in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_CITY</code>	Textbox	The city in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_STATE</code>	Listbox	The state in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE</code>	Textbox	The zip/postal code in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY</code>	Listbox	The country in the customer's shipping address. Imported from <code>editaddresstemplate.inc</code> .

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 black lines and callout text are not part of the template, but explanations of the components.

3 Customer Profile Services

Figure 3-5 Annotated profilenewcc.jsp Template



Location in the Directory Structure

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

`%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\profilenewcc.jsp` (Windows)

`PORTAL_HOME/applications/wlcsApp/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 library:

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

Note: For more information about the Webflow JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

This file resides in the following directory for the Commerce services Web application:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.tags.WebFlowTagConstants" %>  
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>  
<%@ 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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `newcctemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates. The template is described in “About the Included `newcctemplate.inc` Template” on page 2-17.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```


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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdatePaymentInfoIP</code> <code>NewCreditCard Pipeline</code>

Table 3-17 briefly describes each of the Pipelines from Table 3-16. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-17 `profilenewcc.jsp` Associated Pipelines

Pipeline	Description
<code>NewCreditCard</code>	Contains <code>EncryptCreditCardPC</code> and <code>UpdatePaymentInfoPC</code> , 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.

3 Customer Profile Services

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

Note: If a form field is imported from another template, it is indicated in the description. Form fields without import information are in the `profilenewcc.jsp` template.

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 (<code>profilenewcc.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE</code>	Listbox	The type of the customer's credit card. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_HOLDER</code>	Textbox	The name on the credit card. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER</code>	Textbox	The number of the customer's credit card. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH</code>	Listbox	The month of the customer's credit card expiration date. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR</code>	Listbox	The year of the customer's credit card expiration date. Imported from <code>newcctemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1</code>	Textbox	The first line in the customer's billing address. Imported from <code>newcctemplate.inc</code> .

Table 3-18 profilenewcc.jsp Form Fields (Continued)

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

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 black lines and callout text are not part of the template; they are explanations of the template components.

3 Customer Profile Services

Figure 3-6 Annotated profileeditcc.jsp Template

The Commerce Templates header (admin.inc) contains useful information for the benefit of you and your development team.

The page header is created by importing the header.inc template.

The left column is created by importing the leftside.inc template.

This region provides a series of form fields to update credit card payment information.

The footer is created by importing the footer.inc template.

Commerce Templates

About Current Template: **profileeditcc.jsp**

Template Index Administration

Your Logo Here

Click here to see our full line of powerful **Routers!**

Home

Welcome Demo Customer
[View Profile](#)
[Logout](#)

View History
[Orders](#)
[Payments](#)

See Our Large Selection of Saws Here!

Catalog data provided courtesy of **TPN Register**, where supply meets demand.

Update Credit Card

Credit card type: VISA

Name on card: *

Card number: *

Expiration date (mm/yyyy): *

Credit card billing address: *

Address 2:

City: *

State: (Required for U.S. and Canada addresses)

Zip/Postal Code: *

Country: *

Fields marked with (*) are required.

Built On BEA Systems Inc.

Copyright © 1999-2001, BEA Systems Inc.

Location in the Directory Structure

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

```
%PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\profileeditcc.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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 the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*. For more information on the User Management JSP tags, see "Personalization Server JSP Tag Library Reference" in the *Guide to Building Personalized Applications*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.tags.WebFlowTagConstants" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ 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.

3 Customer Profile Services

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `editcctemplate.inc`, which also uses the `states.inc` and the `countries.inc` templates.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

About the Included `editcctemplate.inc` Template

The `editcctemplate.inc` 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.inc` template will validate the form to ensure that all required fields contain values. If errors are detected, the `editcctemplate.inc` template will be redisplayed, with an error

message at the top and the offending field labels shown in a red (as opposed to the original black) font. Previously entered correct information will still be displayed in the form.

Note: The `profileeditcc.jsp` template and the `editcctemplate.inc` do not allow the user to edit a credit card number. To change a credit card number, the credit card must be deleted and then added as a new card.

Since the `editcctemplate.inc` 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.inc` 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.inc`

```
<tr>
  <td width="27%"><webflow:getValidatedValue
  fieldName="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE%">"
  fieldDefaultValue="<%=defaultCreditCard.getType()%>"
  fieldValue="customerCreditCardType" fieldStatus="status" validColor="black"
  invalidColor="red" unspecifiedColor="black" fieldColor="fontColor" />
    <div class="tabletext"><font color=<%= fontColor %>>Credit card type</font>
  </div>
</td>
  <td width="73%">
    <div class="tabletext"><%=customerCreditCardType%">
  </div>
    <input type="hidden"
  name="<%=HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE%">"
  value="<%=customerCreditCardType%">">
  </td>
</tr>
```

3 Customer Profile Services

Notes: For more information about the `getValidatedValue` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdatePaymentInfoIP</code> <code>UpdateCreditCard Pipeline</code>

Table 3-20 briefly describes each of the Pipelines from Table 3-19. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-20 profileeditcc.jsp Associated Pipelines

Pipeline	Description
<code>UpdateCreditCard</code>	Contains <code>UpdatePaymentPC</code> 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.inc` template can display this information in the payment information form fields. This is accomplished on the `profileeditcc.jsp` template using a combination the User Management JSP tags and accessor methods/attributes.

First, the `getProfile` JSP tag is used to set the customer profile (context) in the session 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 User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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 propertySet="CustomerProperties"
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();
%>
```

3 Customer Profile Services

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 `defaultCreditCard`, while Table 3-22 provides more information about the accessor methods/attributes on `billingAddress`.

Table 3-21 defaultCreditCard Accessor Methods/Attributes

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

Table 3-22 billingAddress Accessor Methods/Attributes

Method/Attribute	Description
<code>getStreet1()</code>	The first line in the customer's billing street address.
<code>getStreet2()</code>	The second line in the customer's billing street address.
<code>getCity()</code>	The city in the customer's billing address.
<code>getState()</code>	The state in the customer's billing address.
<code>getPostalCode()</code>	The zip/postal code in the customer's billing address.
<code>getCountry()</code>	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. It is also used to pass needed information to the Webflow.

The form fields used in the `profileeditcc.jsp` template, most of which are imported from the `editcctemplate.inc` file. A description for each of these fields are listed in Table 3-23, “`profileeditcc.jsp` Form Fields,” on page 3-50.

3 Customer Profile Services

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.
“namespace”	Hidden	The namespace for the JSP; sampleapp_user in this JSP.
HttpRequestConstants.CUSTOMER_CREDITCARD_KEY	Hidden	The map key of the customer’s credit card.
HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE	Hidden	The type of the customer’s credit card. Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD HOLDER	Textbox	The name on the credit card. Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER	Hidden	The number of the customer’s credit card. This field does not display on screen. Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD_DISPLAY_NUMBER	Hidden	The display version of the customer’s credit card (12 Xs and last 4 digits). Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH	Listbox	The month of the customer’s credit card expiration date. Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR	Listbox	The year of the customer’s credit card expiration date. Imported from editcctemplate.inc.
HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1	Textbox	The first line in the customer’s billing address. Imported from editcctemplate.inc.

Table 3-23 profileeditcc.jsp Form Fields (Continued)

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

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 black lines and callout text are not part of the template; they are explanations of the template components.

3 Customer Profile Services

Figure 3-7 Annotated changepassword.jsp Template



Location in the Directory Structure

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

`PORTAL_HOME\applications\wlcsApp\wlcs\commerce\user\changepassword.jsp` (Windows)

`PORTAL_HOME/applications/wlcsApp/wlcs/commerce/user/changepassword.jsp` (UNIX)

Tag Library Imports

The `changepassword.jsp` template uses Webflow JSP tags and 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 the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*. For more information on the User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
```

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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```


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)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdateAccountInfoIP</code> <code>UpdateAccountProfile Pipeline</code>

Table 3-25 briefly describes each of the Pipelines from Table 3-24. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-25 `changepassword.jsp` Associated Pipelines

Pipeline	Description
<code>UpdateAccountProfile</code>	Contains <code>UpdatePasswordPC</code> and is not 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.

3 Customer Profile Services

Listing 3-15 Displaying the Customer's Username

```
...  
  
<td width="73%" valign="top">>  
  <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 (<code>changepassword.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpServletRequestConstants.PASSWORD</code>	Password	The customer's existing password used to login.

Table 3-26 `changepassword.jsp` Form Fields (Continued)

Parameter Name	Type	Description
<code>HttpServletRequest.NEW_PASSWORD</code>	Password	The new password chosen by the customer for login.
<code>HttpServletRequest.CONFIRM_PASSWORD</code>	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 `<%= HttpServletRequest.CONFIRM_PASSWORD %>`) for use in the JSP.

editdemographics.jsp Template

The `editdemographics.jsp` template (shown in Figure 3-8) allows a customer to change their demographic information, which will be stored as part of their profile.

Sample Browser View

Figure 3-8 and Figure 3-9 show annotated versions of the `editdemographics.jsp` template. Although there are two figures, together these screen shots form the single `editdemographics.jsp` template. The black lines and callout text are not part of the template; they are explanations of the template components.

3 Customer Profile Services

Figure 3-8 Annotated editdemographics.jsp Template - First half

The Commerce Templates header (admin.inc) contains useful information for the benefit of you and your development team.

The page header is created by importing the header.inc template.

The left column is created by importing the leftside.inc template.

The main body of the page provides a series of radio buttons/form fields to change or update a customer's demographic information.

Commerce Templates

About Current Template: `editdemographics.jsp`

Template Index Administration

Your Logo Here

Click here to see our full line of powerful **Routers!**

Home

Welcome Demo Customer
[View Profile](#)
[Logout](#)

View History
[Orders](#)
[Payments](#)

Check Out Our Low Prices on Drills!

Catalog data provided courtesy of [TPN Register](#), where supply meets demand.

Edit Demographic Data

Gender * Female Male

Date of Birth * (mm/dd/yyyy)

Occupation * Clerical Executive Management Professional Engineering Management Sales

Employment Status * Not employed, not looking for work Self-employed Employed Not employed, looking for work

Marital Status * Widowed Married Single Divorced

Education Level * High School College Graduate Graduate Degree Professional Degree Some College

Income Range * Under \$35,000 \$35,000 to \$49,999 \$50,000 to \$74,999 \$75,000 to \$99,999

Figure 3-9 Annotated editdemographics.jsp Template - Second half

The screenshot displays the second half of the `editdemographics.jsp` template. It features three main form sections, each with a title and a list of radio button options:

- Education Level ***:
 - Graduate Degree
 - Professional Degree
 - Some College
- Income Range ***:
 - Under \$35,000
 - \$35,000 to \$49,999
 - \$50,000 to \$74,999
 - \$75,000 to \$99,999
 - \$100,000 to \$124,999
 - \$125,000 and above
- Handiness ***:
 - Do It Yourselfer
 - All thumbs
 - Saturday Helper
 - Professional

Below these sections, a note states: "Fields marked with (*) are required." At the bottom of the form area, there are two buttons: "Save" and "< Back".

The footer section contains the following text: "The footer is created by importing the footer.inc template." (with a line pointing to the footer area), "Built On" with the BEA logo, "Copyright © 1999-2001, BEA Systems Inc.", and a text box containing: "The main body of the page provides a series of radio buttons/form fields to change or update a customer's demographic information."

Location in the Directory Structure

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\
commerce\user\editdemographics.jsp (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/
commerce/user/editdemographics.jsp (UNIX)
```

Tag Library Imports

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

3 Customer Profile Services

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

Note: For more information about the Webflow JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*. For more information on the User Management JSP tags, see “Personalization Server JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/
WEB-INF (UNIX)
```

Java Package Imports

The `editdemographics.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.axiom.contact.*" %>
<%@ page import="com.beasys.commerce.ebusiness.customer.*" %>
<%@ page import="java.util.*" %>
```

Location in Default Webflow

The page before the `editdemographics.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, `editdemographics.jsp` is reloaded.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `newdemographictemplate.inc`, which contains formatting for the demographic data. The template is described in “About the Included `newdemographictemplate.inc` Template” on page 2-18.
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

Events

The `editdemographics.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 3-27 provides information about these events and the business logic they invoke.

Table 3-27 `editdemographics.jsp` Events

Event	Webflow Response(s)
<code>button.back</code>	No business logic required. Loads <code>viewprofile.jsp</code> .
<code>button.save</code>	<code>UpdateDemographicInfoIP</code> <code>EditDemographicInfo</code> Pipeline

Table 3-28 briefly describes each of the Pipelines from Table 3-27. For more information about individual Pipeline Components, see “Pipeline Components” on page 3-72.

Table 3-28 `editdemographics.jsp` Associated Pipelines

Pipeline	Description
<code>EditDemographicInfo</code>	Contains <code>UpdateDemographicInfoPC</code> and is transactional.

Dynamic Data Display

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

Form Field Specification

The primary purpose of the `editdemographics.jsp` template is to allow customers to make changes to their demographic information using HTML form fields and radio buttons. It is also used to pass needed information to the Webflow.

The form fields used in the `editdemographics.jsp` template, most of which are imported from the `newdemographictemplate.inc` file, and a description for each of these form fields are listed in Table 3-29.

Note: If a form field is imported from another template, it is indicated in the description. Form fields without import information are in the `editdemographics.jsp` template.

Table 3-29 `editdemographics.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 (<code>editdemographics.jsp</code>), used by the Webflow.
"namespace"	Hidden	The namespace for the JSP; <code>sampleapp_user</code> in this JSP.
<code>HttpRequestConstants.CUSTOMER_GENDER</code>	Radio buttons	Identifies the customer as male or female. Imported.
<code>HttpRequestConstants.CUSTOMER_DATE_OF_BIRTH</code>	Textboxes	The customer's date of birth. Imported from <code>newdemographictemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_OCCUPATION</code>	Radio buttons	The customer's job description. Imported from <code>newdemographictemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_EMPLOYMENT_STATUS</code>	Radio buttons	Identifies if the customer has a job at the time of registration. Imported from <code>newdemographictemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_MARITAL_STATUS</code>	Radio buttons	Identifies the customer's marital status. Imported from <code>newdemographictemplate.inc</code> .
<code>HttpRequestConstants.CUSTOMER_EDUCATION_LEVEL</code>	Radio buttons	Identifies how much formal education the customer has completed. Imported from <code>newdemographictemplate.inc</code> .

3 Customer Profile Services

Table 3-29 editdemographics.jsp Form Fields

Parameter Name	Type	Description
HttpRequestConstants. CUSTOMER_INCOME_RANGE	Radio buttons	Identifies the customer's yearly income. Imported from newdemographictemplate.inc.
HttpRequestConstants. CUSTOMER_QUALITY	Radio buttons	Ranks customer from beginner to expert in using your product. Imported from newdemographictemplate.inc.

Note: Parameters that are literals in the JSP code are shown in quotes, while non-literals will require scriptlet syntax (such as `<%= HttpRequestConstants.CUSTOMER_QUALITY %>`) 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	<code>examples.wlcs.sampleapp.customer.webflow.DeleteCreditCardIP</code>
Description	Deletes a <code>CreditCard</code> from the <code>CreditCardMap</code> and creates a new <code>CustomerValue</code> object; then sets the <code>CreditCardMap</code> on <code>CustomerValue</code> and places it into the Pipeline session.
Required HttpServletRequest Parameters	<code>HttpRequestConstants.CREDITCARD_KEY</code> (code location: <code>profileeditcc.jsp</code> template.)
Required Pipeline Session properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session properties	<code>PipelineSessionConstants.CUSTOMER</code>
Removed Pipeline Session properties	None
Validation	Verifies that <code>HttpRequestConstants.CREDITCARD_KEY</code> is not <code>NULL</code> .
Exceptions	<code>InvalidInputException</code> , thrown if <code>HttpRequestConstants.CREDITCARD_KEY</code> is <code>NULL</code> . <code>InvalidSessionStateException</code> , thrown if the session is unavailable or has expired.

DeleteShippingAddressIP

Class Name	<code>examples.wlcs.sampleapp.customer.webflow.DeleteShippingAddressIP</code>
Description	Deletes a <code>ShippingAddress</code> from the <code>ShippingAddressMap</code> and creates a new <code>CustomerValue</code> object; then sets the <code>ShippingAddressMap</code> on <code>CustomerValue</code> and places it into the Pipeline session.
Required HttpServletRequest Parameters	<code>HttpRequestConstants.ADDRESS_KEY</code> (code location: <code>profileeditaddress.jsp</code> template.)
Required Pipeline Session properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session properties	<code>PipelineSessionConstants.CUSTOMER</code>
Removed Pipeline Session properties	None
Validation	Verifies that <code>HttpRequestConstants.ADDRESS_KEY</code> is not <code>NULL</code> .
Exceptions	<code>InvalidInputException</code> , thrown if <code>HttpRequestConstants.ADDRESS_KEY</code> is <code>NULL</code> . <code>InvalidSessionStateException</code> , thrown if the session is unavailable or has expired.

UpdateAccountInfoIP

Class Name	<code>examples.wlcs.sampleapp.customer.webflow.UpdateAccountInfoIP</code>
Description	Processes the customer's input from the <code>changepassword.jsp</code> . Creates a <code>CustomerValue</code> object in the Pipeline session containing the new information.
Required HttpServletRequest Parameters	<code>HttpRequestConstants.PASSWORD</code> <code>HttpRequestConstants.NEW_PASSWORD</code> <code>HttpRequestConstants.CONFIRM_PASSWORD</code> (code location: <code>changepassword.jsp</code> template.)
Required Pipeline Session properties	None
Updated Pipeline Session properties	<code>PipelineSessionConstants.PASSWORD</code>
Removed Pipeline Session properties	None
Validation	Validates the current password and verifies that the required fields contain values.
Exceptions	<code>InvalidInputException</code> , 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. <code>ProcessingException</code> , thrown in the case of a configuration error.

UpdateBasicInfoIP

Class Name	examples.wlcs.sampleapp.customer.webflow. UpdateBasicInfoIP
Description	Processes the customer's input from the <code>editprofile.jsp</code> . Creates a <code>CustomerValue</code> 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 HttpRequestConstants.CUSTOMER_EMAIL_OPT_IN (code location: <code>editprofile.jsp</code> template.)
Required Pipeline Session properties	PipelineSessionConstants.USER_NAME
Updated Pipeline Session properties	PipelineSessionConstants.CUSTOMER
Removed Pipeline Session properties	None
Validation	Verifies that the required fields contain values.
Exceptions	InvalidInputException, thrown if the required fields do not contain values. ProcessingException, thrown if the customer id in the pipeline session is invalid.

UpdateDemographicInfoIP

Class Name	examples.wlcs.sampleapp.customer.webflow. UpdateDemographicInfoIP
Description	Processes the customer's input from the editdemographics.jsp. Creates a CustomerValue object in the Pipeline session containing the new information.
Required HttpServletRequest Parameters	HttpRequestConstants.CUSTOMER_GENDER HttpRequestConstants.CUSTOMER_DATE_OF_BIRTH HttpRequestConstants.CUSTOMER_OCCUPATION HttpRequestConstants.CUSTOMER_EMPLOYMENT_STATUS HttpRequestConstants.CUSTOMER_MARITAL_STATUS HttpRequestConstants.CUSTOMER_EDUCATION_LEVEL HttpRequestConstants.CUSTOMER_INCOME_RANGE HttpRequestConstants.CUSTOMER_QUALITY HttpRequestConstants.HANDINESS (code location: newdemographictemplate.inc template.)
Required Pipeline Session properties	PipelineSessionConstants.USER_NAME
Updated Pipeline Session properties	PipelineSessionConstants.CUSTOMER PipelineSessionConstants.CREDITCARD_KEY
Removed Pipeline Session properties	None
Validation	Verifies that the status of invalidFieldisPresent is not true.
Exceptions	InvalidInputException, thrown if invalidFieldisPresent is true. InvalidSessionStateException, thrown when the session is unavailable or has expired. ProcessingException, thrown if the customer ID in the Pipeline session is invalid.

UpdatePaymentInfoIP

Class Name	examples.wlcs.sampleapp.customer.webflow. UpdatePaymentInfoIP
Description	Processes the customer's input from <code>profilenewcc.jsp</code> and <code>profileeditcc.jsp</code> . Creates a <code>CustomerValue</code> object in the Pipeline session containing the new information.
Required HttpServletRequest Parameters	<p><code>HttpRequestConstants.CREDITCARD_KEY</code> (code location: <code>profileedit.jsp</code> template.)</p> <hr/> <p><code>HttpRequestConstants.CUSTOMER_CREDITCARD_TYPE</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD HOLDER</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_NUMBER</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_DISPLAY_NUMBER</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_MONTH</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_YEAR</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS1</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_ADDRESS2</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_CITY</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_STATE</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_ZIPCODE</code> <code>HttpRequestConstants.CUSTOMER_CREDITCARD_COUNTRY</code> (code location: <code>editcctemplate.inc</code> template.)</p>
Required Pipeline Session properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session properties	<code>PipelineSessionConstants.CUSTOMER</code> <code>PipelineSessionConstants.CREDITCARD_KEY</code>
Removed Pipeline Session properties	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	<p><code>InvalidInputException</code>, thrown if the required fields do not contain values or the credit card number is less than the minimum required for the type.</p> <p><code>InvalidSessionStateException</code>, thrown if the session is unavailable or has expired.</p>
-------------------	---

UpdateShippingInfoIP

Class Name	<code>examples.wlcs.sampleapp.customer.webflow.UpdateShippingInfoIP</code>
Description	Processes the customer's input from the <code>profileeditaddress.jsp</code> . Creates a <code>CustomerValue</code> object in the Pipeline session containing the new information.
Required HttpServletRequest Parameters	<p><code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS1</code> <code>HttpRequestConstants.CUSTOMER_SHIPPING_ADDRESS2</code> <code>HttpRequestConstants.CUSTOMER_SHIPPING_CITY</code> <code>HttpRequestConstants.CUSTOMER_SHIPPING_STATE</code> <code>HttpRequestConstants.CUSTOMER_SHIPPING_ZIPCODE</code> <code>HttpRequestConstants.CUSTOMER_SHIPPING_COUNTRY</code> (code location: <code>editaddressstemplate.inc template</code>.)</p>
Required Pipeline Session properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session properties	<code>PipelineSessionConstants.CUSTOMER</code>
Removed Pipeline Session properties	None
Validation	Verifies that the required fields contain values.
Exceptions	<p><code>InvalidInputException</code>, thrown when the required fields do not contain values.</p> <p><code>InvalidSessionStateException</code>, thrown if the session is unavailable or is expired.</p>

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	<code>examples.wlcs.sampleapp.customer.pipeline.UpdateBasicInfoPC</code>
Description	Updates the Customer object for changes made by UpdateBasicInfoIP. This Pipeline component must stay in sync with the UpdateBasicInfoIP Input Processor.
Contained in	EditBasicInfo Pipeline
Required Pipeline Session properties	None
Updated Pipeline Session properties	None
Removed Pipeline Session properties	None
Type	Java class
JNDI Name	None
Exceptions	PipelineException, thrown when the Pipeline component is not able to set the customer's properties.

UpdateDemographicInfoPC

Class Name	<code>examples.wlcs.sampleapp.customer.pipeline.UpdateBasicInfoPC</code>
Description	Updates the <code>Customer</code> object for changes made by <code>UpdateDemographicInfoIP</code> . This Pipeline component must stay in sync with the <code>UpdateBasicDemographicIP</code> Input Processor.
Contained in	<code>EditDemographicInfo</code> Pipeline
Required Pipeline Session properties	None
Updated Pipeline Session properties	None
Removed Pipeline Session properties	None
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when the Pipeline component is not able to set the customer's properties.

UpdatePasswordPC

Class Name	<code>examples.wlcs.sampleapp.customer.pipeline.UpdatePasswordPC</code>
Description	Retrieves the <code>USER_NAME</code> and <code>PASSWORD</code> from the Pipeline session and updates the password for the user.
Contained in	UpdateAccountProfile Pipeline
Required Pipeline Session properties	<code>PipelineSessionConstants.USER_NAME</code> <code>PipelineSessionConstants.PASSWORD</code>
Updated Pipeline Session properties	None
Removed Pipeline Session properties	<code>PipelineSessionConstants.PASSWORD</code>
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when the Pipeline component is not able to set the customer's properties.

UpdatePaymentInfoPC

Class Name	<code>examples.wlcs.sampleapp.customer.pipeline.UpdatePaymentInfoPC</code>
Description	Updates the <code>Customer</code> object for changes made by <code>UpdatePaymentInfoIP</code> . This Pipeline component must stay in sync with the the <code>UpdatePaymentInfoIP</code> Input Processor.
Contained in	<code>DeleteCreditCard</code> , <code>NewCreditCard</code> , <code>NewCreditCardForPayment</code> , <code>UpdateCreditCard</code> , and <code>UpdateCreditCardForPayment</code> Pipelines
Required Pipeline Session properties	None
Updated Pipeline Session properties	None
Removed Pipeline Session properties	None
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when the Pipeline component is not able to set the customer's properties.

UpdateShippingInfoPC

Class Name	<code>examples.wlcs.sampleapp.customer.pipeline.UpdateShippingInfoPC</code>
Description	Updates the Customer object for changes made by <code>UpdateShippingInfoIP</code> . This Pipeline component must stay in sync with the the <code>UpdateShippingInfoIP</code> Input Processor.
Contained in	<code>ProfileNewAddress</code> , <code>ProfileEditAddress</code> , and <code>DeleteShippingAddressFromProfile</code> Pipelines
Required Pipeline Session properties	None
Updated Pipeline Session properties	None
Removed Pipeline Session properties	None
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when the Pipeline component is not able to set the customer's properties.

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 Registering Customers and Managing Customer services provide 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)
 - 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 Registering Customers and Managing Customer services, 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 the customer self-service pages in detail.

Note: For a description of the complete set of JSPs used in the Commerce services Web application and a listing of their locations in the directory structure, see the “Template Summary ” documentation.

A customer must be logged into your e-commerce site for the customer self-service options to be available. The customer self-service options appear in the left column created by the `leftside.inc` template. Figure 4-1 shows the `main.jsp` template (the home page for the product catalog) with the options available. For more information about the `main.jsp` template, see “The Product Catalog JSP Templates” in the *Guide to Building a Product Catalog*.

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

The Customer Self-Service section is only available if the customer is logged into the Web site. Customer Self-Service consists of order history, order status, and payment status.

Navigation Bar: Home | Search | View Cart | Log in

Store Catalog:

- Hardware
- Power Tools
- Measuring Tools
- Tool Sets
- Lawn/Garden Tools
- Other

Customer Self-Service Section:

Welcome Demo Customer
[View Profile](#)
[Logout](#)

View History
[Orders](#)
[Payments](#)

Quick Look-up:
 Enter keywords

Register Now and save \$10 on your order of at least \$50 placed today!!

Catalog data provided courtesy of [TPN Register](#), where supply meets demand.

Copyright © 1999-2001, 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 black lines and callout text are not part of the template; they are explanations of the template components.

Figure 4-2 Annotated orderhistory.jsp Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

The main body of the page contains dynamically-generated data about the customer's order history.

The footer is created by importing the `footer.inc` template.

Order History

Date	Order Number	Amount	
2001-03-28	1	\$164.75	View

1 - 1

< Back

Copyright © 1999-2001, BEA Systems Inc.

Location in the Directory Structure

You can find the `orderhistory.jsp` template file at the following location, where `$PORTAL_HOME` is the directory in which you installed Commerce services:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\order\orderhistory.jsp (Windows)
```

```
$PORTAL_HOME/applications/wlcsApp/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="webflow.tld" prefix="webflow" %>  
<%@ taglib uri="eb.tld" prefix="eb" %>  
<%@ taglib uri="i18n.tld" prefix="i18n" %>
```

Note: For more information about the Pipeline JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*. For more information on the Commerce services JSP tags, see “JSP Tag References” in the Campaign and Commerce services documentation.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\  
WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.axiom.units.*" %>
```

4 Customer Self-Service

```
<%@ 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.ebusiness.util.*" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ page import="com.beasys.commerce.webflow.PipelineSessionConstants" %>
<%@ page import="com.bea.p13n.appflow.webflow.WebflowJSPHelper" %>
```

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`).

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:
`<%@ include file="/commerce/includes/stylesheet.inc" %>`
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/header.inc" %>`
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/leftside.inc" %>`
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/footer.inc" %>`

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 Pipeline
button.viewOrderStatus	SelectOrderForViewingIP

Table 4-2 briefly describes each of the Pipelines from Table 4-1. For more information about individual Pipeline Components, see “Pipeline Components” on page 4-26.

Table 4-2 orderhistory.jsp Associated 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.

4 Customer Self-Service

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

Table 4-3 `orderhistory.jsp` Pipeline Session Properties

Attribute	Type	Description
<code>PipelineSessionConstant.SCROLLABLE_MODEL</code>	<code>com.beasys.commerce.ebusiness.util.ScrollableModel</code>	List of the orders available for the customer.

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

Listing 4-1 Retrieving the Order History Attribute

```
<webflow:getProperty id="orderHistory"
property="<%=PipelineSessionConstants.SCROLLABLE_MODEL%>"
type="com.beasys.commerce.ebusiness.util.ScrollableModel"
scope="session" namespace="sampleapp_main" />
```

Note: For more information on the `getProperty` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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
<code>createdDate()</code>	The date the customer's order was created.
<code>identifier()</code>	Key in the database for the order.
<code>getValue()</code>	The total price of that order. The price attribute is a money object.

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="<%=currentPage%>" id="orderValue" type="OrderValue" count="100">
  <tr>
    <td>
      <div class="tabletext"><%= orderValue.createdDate %></div>
    </td>
    <td>
      <div class="tabletext"><%= orderValue.identifier %></div>
    </td>
    <td>
      <div class="tabletext"><% Money total = orderValue.price; %>
      <i18n:getMessage bundleName="/commerce/currency" messageName="<%=
        total.getCurrency() %>" />&nbsp; <%=
        WebflowJSPHelper.priceFormat(total.getValue()) %>
      </div>
    </td>
    .
    .
    .
  </tr>
</wl:repeat>
```

Note: For more information on the WebLogic Commerce Server JSP tags, see “JSP Tag References” in the Campaign and Commerce services 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 black lines and callout text are not part of the template, but explanations of the components.

Figure 4-3 Annotated `orderstatus.jsp` Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

The main body of the page contains dynamically-generated data about a particular order the customer selected from the `orderhistory.jsp` template.

The footer is created by importing the `footer.inc` template.

Order Status

Confirmation number: 1
 Order status: Submitted
 Date ordered: 2001-03-28
 Splitting preferences: Ship as the items become available
 Special instructions: None
 Shipping address: One Winthrop Square, BOSTON, MA-02110, United States

ID	Description	Quantity	Subtotal
9-27168	saw-9-27168	1	\$ 151.95
Shipping & Handling			\$ 4.95
Total tax			\$ 7.85
Total due			\$ 164.75

Location in Commerce services Directory Structure

You can find the `orderstatus.jsp` template file at the following location, where `$PORTAL_HOME` is the directory in which you installed Commerce services:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\order\orderstatus.jsp (Windows)
```

```
$PORTAL_HOME/applications/wlcsApp/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="webflow.tld" prefix="webflow" %>  
<%@ taglib uri="i18n.tld" prefix="i18n" %>
```

Note: For more information on the WebLogic Commerce Server JSP tags, see “JSP Tag References” in the Campaign and Commerce services documentation. For more information about the Pipeline JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF (UNIX)
```

Java Package Imports

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

4 Customer Self-Service

```
<%@ page import="java.util.*" %>
<%@ page import="java.text.*" %>
<%@ 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.ebusiness.customer.*" %>
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>
<%@ page import="com.beasys.commerce.webflow.PipelineSessionConstants" %>
<%@ page import="com.bea.p13n.appflow.webflow.WebflowJSPHelper" %>
```

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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```

- footer.inc, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:
`<%@ include file="/commerce/includes/footer.inc" %>`

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. The dynamic content on orderstatus.jsp is obtained using a combination of WebLogic Server JSP tags, Pipeline JSP tags, and accessor methods/attributes.

First, the getProperty 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 Properties

Attribute	Type	Description
PipelineSessionConstant. SELECTED_ORDER	com.beasys.commerce. ebusiness.order.OrderValue	Contains information about the order selected by the customer.

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

Listing 4-3 Retrieving the Selected Order Attribute

```
<webflow:getProperty id="orderValue"  
property="<%=PipelineSessionConstants.SELECTED_ORDER%>"  
type="com.beasys.commerce.ebusiness.order.OrderValue"  
scope="session" namespace="sampleapp_main" />
```

Note: For more information on the getProperty JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

4 Customer Self-Service

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
<code>createdDate()</code>	The date the customer's order was created.
<code>identifier()</code>	Key in the database for the order; the order confirmation number.
<code>orderStatus()</code>	The status of the order.
<code>splittingPreference()</code>	The splitting preference for the order.
<code>specialInstructions()</code>	Any special instructions for the order.
<code>shippingAddress()</code>	The shipping address for the order.
<code>orderLines()</code>	A collection of the lines in the shopping cart that make up the customer's order.
<code>getTotal()</code>	In this instance, the total tax specified by the <code>OrderConstants.LINE_TAX</code> parameter. Note: The <code>getTotal()</code> method also allows you to combine different total types. For more information, see the <i>Javadoc</i> .
<code>adjustmentPresentations()</code>	Returns a list of adjustments or discounts applied to the overall order in the form of (<code>OrderAdjustmentPresentation</code>). See Table 4-7 for more information.

Table 4-7 provides more detailed information about the methods/attributes for `OrderAdjustPresentation`.

Table 4-7 OrderAdjustmentPresentation Accessor Methods

Method/Attribute	Description
<code>getReason()</code>	Describes why the discount applies to the customer.
<code>getComputation()</code>	The formula used to compute the discount.

Table 4-7 OrderAdjustmentPresentation Accessor Methods

Method/Attribute	Description
<code>getDiscount()</code>	This is the actual dollar amount of the discount.
<code>getAdjustmentType()</code>	Describes how the price adjustment is applied. For example, a discount is subtracted from the total price of the objects ordered, but a shipping tax is calculated from and added to the shipping price.

Table 4-8 describes the accessor methods/attributes available within the `shippingAddress` attribute of `OrderValue`.

Table 4-8 shippingAddress Accessor Methods

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

Table 4-9 describes the accessor methods/attributes available for each `OrderLine` of the `OrderLines` attribute.

Table 4-9 OrderLine Accessor Methods

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

4 Customer Self-Service

Table 4-9 OrderLine Accessor Methods

Method/Attribute	Description
<code>getAdjustmentPresentations()</code>	Returns a list of order presentation objects. See Table 4-7 for more information.

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="60%" cellpadding="4" cellspacing="0">
<tr>
  <td><div class="tabletext"><b>Confirmation number</b></div></td>
  <td><div class="tabletext"><%=orderValue.identifier%></div></td>
</tr>
.
.
.
<tr>
  <td valign="top"><div class="tabletext"><b>Shipping address</b></div>
  </td>
  <td valign="top">
    <div class="tabletext"><%=orderValue.shippingAddress.getStreet1()%>
    </div>
    <% if(orderValue.shippingAddress.getStreet2().length() != 0) { %>
    <div class="tabletext"><%=orderValue.shippingAddress.getStreet2()%>
    </div>
    <% } %>
    <div class="tabletext"><%=orderValue.shippingAddress.getCity()%></div>
    <div class="tabletext"><%String stateZip =
      orderValue.shippingAddress.getState()+ "-" +
      orderValue.shippingAddress.getPostalCode();%>
    </div>
    <div class="tabletext"><%=stateZip%>
    </div>
    <div class="tabletext"><%=orderValue.shippingAddress.getCountry()%>
    </div>
  </td>
</tr>
```

```
.
.
.
<wl:repeat set="<%=orderValue.orderLines.iterator()%>" id="orderLine"
type="OrderLine" count="100">
<tr>
  <td nowrap>
    <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" nowrap>
    <%
      // Calculate the line subtotal without discounts
      double orderLineTotal = (orderLine.getQuantity() *
        orderLine.getUnitPrice().getValue());
    %>
    <div class="tabletext">
      <il8n:getMessage bundleName="/commerce/currency" messageName="<%=
        orderLine.getUnitPrice().getCurrency() %>"/>&nbsp;&lt;%=
        WebflowJSPHelper.priceFormat( orderLineTotal ) %>
    </div>
  </td>
</tr>
.
.
.
</wl:repeat>
```

Note: For more information on the WebLogic Commerce Server JSP tags, see the “JSP Tag References” in the Campaign and Commerce services 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 black lines and callout text are not part of the template; they are explanations of the template components.

Figure 4-4 Annotated paymenthistory.jsp Template

The Commerce Templates header (`admin.inc`) contains useful information for the benefit of you and your development team.

The page header is created by importing the `header.inc` template.

The left column is created by importing the `leftside.inc` template.

The main body of the page contains dynamically-generated data about a customer's payments.

The footer is created by importing the `footer.inc` template.

Date	Transaction ID	Credit card	Amount
2001-03-28	1985804109839	XXXXXXXXXXXX1111	\$151.95

Location in the Directory Structure

You can find the `paymenthistory.jsp` template file at the following location, where `$PORTAL_HOME` is the directory in which you installed Commerce services:

```
%PORTAL_HOME\applications\wlcsApp\wlcs\  
commerce\order\paymenthistory.jsp (Windows)
```

```
$PORTAL_HOME/applications/wlcsApp/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="webflow.tld" prefix="webflow" %>  
<%@ taglib uri="i18n.tld" prefix="i18n" %>
```

Note: For more information on the WebLogic Commerce Server JSP tags, see “JSP Tag References” in the Campaign and Commerce services documentation. For more information about the Pipeline JSP tags, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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

```
PORTAL_HOME\applications\wlcsApp\wlcs\WEB-INF (Windows)
```

```
PORTAL_HOME/applications/wlcsApp/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.ebusiness.payment.*" %>  
<%@ page import="com.beasys.commerce.webflow.HttpRequestConstants" %>  
<%@ page import="com.beasys.commerce.webflow.PipelineSessionConstants" %>  
<%@ page import="com.bea.p13n.appflow.webflow.WebflowJSPHelper" %>
```

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.

This template is part of the `sampleapp_user` namespace in the Webflow.

Note: For more information about the default Webflow, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Included JSP Templates

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

- `admin.inc`, which shows the name of the current template and contains links to its *About* information, the JSP Template Index, and the Administration Tools. The `admin.inc` template should be removed from the production pages before they are moved to your live server.
- `stylesheet.inc`, which is a cascading stylesheet that defines global paragraph and text styles for the site. The import call is:

```
<%@ include file="/commerce/includes/stylesheet.inc" %>
```
- `header.inc`, which creates the page header. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/header.inc" %>
```
- `leftside.inc`, which creates the left column and secondary placeholder for advertising. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/leftside.inc" %>
```
- `footer.inc`, which creates the page footer. It is standard across many of the JSP templates provided by Commerce services. The import call is:

```
<%@ include file="/commerce/includes/footer.inc" %>
```

Events

There are no events on the `paymenthistory.jsp` template that trigger Input Processors or Pipelines in the Webflow. Table 4-10 briefly describes each of the Pipelines associated with the `paymenthistory.jsp` template. For more information about individual Pipeline Components, see “Pipeline Components” on page 4-26.

Table 4-10 paymenthistory.jsp Associated 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 getProperty 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 Properties

Attribute	Type	Description
PipelineSessionConstant. 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 getProperty JSP tag.

4 Customer Self-Service

Listing 4-5 Retrieving the Payment History Attribute

```
<webflow:getProperty id="paymentHistory"  
property="<%=PipelineSessionConstants.PAYMENT_HISTORY%" type="java.util.List"  
scope="request" namespace="sampleapp_main" />
```

Note: For more information on the `getProperty` JSP tag, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

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
<code>transactionDate()</code>	The date of the payment transaction.
<code>transactionId()</code>	Key in the database for the transaction; the payment confirmation number.
<code>creditCard()</code>	The status of the order.
<code>transactionAmount()</code>	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
<code>getDisplayNumber()</code>	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">
  <tr>
    <td align="left">
      <!-- Get transactionDate from PaymentTransactionValue -->
      <div class="tabletext"><%= paymentTransactionValue.transactionDate %>
      </div>
    </td>
    <td align="center">
      <!-- Get transactionId from PaymentTransactionValue -->
      <div class="tabletext"><%= paymentTransactionValue.transactionId %>
      </div>
    </td>
    <td align="center">
      <!-- Get credit card display from PaymentTransactionValue -->
      <div class="tabletext"><%=
        paymentTransactionValue.creditCard.getDisplayNumber() %>
      </div>
    </td>
    <td align="right">
      <!-- Get transactionAmount from PaymentTransactionValue -->
      <!-- The WebflowJSPHelper.priceFormat() converts a double to a String
        with two significant digits after the decimal-->
      <div class="tabletext"><il8n:getMessage bundleName="/commerce/currency"
messageName="<%= paymentTransactionValue.transactionAmount.getCurrency()
%>" />&nbsp;  <%=

```

4 Customer Self-Service

```
        WebflowJSPHelper.priceFormat(paymentTransactionValue.  
        transactionAmount.getValue()) %>  
    </div>  
</td>  
</tr>  
</wl:repeat>
```

Note: For more information on the WebLogic Commerce Server JSP tags, see “JSP Tag References” in the Campaign and Commerce services 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	<code>examples.wlcs.sampleapp.order.webflow.SelectOrderForViewingIP</code>
Description	Reads the order identifier and uses it to locate an <code>OrderValue</code> object from the <code>ORDER_HISTORY</code> attribute, then places the object in the Pipeline session.
Required HttpServletRequest Parameters	<code>HttpRequestConstants.ORDER_IDENTIFIER</code> (code location: <code>orderhistory.jsp</code> template.)
Required Pipeline Session Properties	<code>PipelineSessionConstants.ORDER_HISTORY</code>
Updated Pipeline Session Properties	<code>PipelineSessionConstants.ORDER_ADJUSTMENT</code> <code>PipelineSessionConstants.SELECTED_ORDER</code>
Removed Pipeline Session Properties	None
Validation	None
Exceptions	<code>ProcessingException</code> , thrown when the order identifier is not found in the HTTP request.

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	<code>examples.wlcs.sampleapp.order.pipeline.RefreshOrderHistoryPC</code>
Description	Uses the <code>USER_NAME</code> Pipeline session attribute to obtain the customer's order history.
Contained in	RefreshOrderHistory Pipeline
Required Pipeline Session Properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session Properties	<code>PipelineSessionConstants.ORDER_HISTORY</code>
Removed Pipeline Session Properties	None
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when required Pipeline session properties are not available, or if username does not exist in the Pipeline session.

RefreshPaymentHistoryPC

Class Name	<code>examples.wlcs.sampleapp.order.pipeline.RefreshPaymentHistoryPC</code>
Description	Uses the <code>USER_NAME</code> Pipeline session attribute to obtain the customer's payment history.
Contained in	RefreshPaymentHistory Pipeline
Required Pipeline Session Properties	<code>PipelineSessionConstants.USER_NAME</code>
Updated Pipeline Session Properties	<code>PipelineSessionConstants.PAYMENT_HISTORY</code>
Removed Pipeline Session Properties	None
Type	Java class
JNDI Name	None
Exceptions	<code>PipelineException</code> , thrown when required Pipeline session properties are not available, or if the username does not exist in the Pipeline session.

4 *Customer Self-Service*

Index

A

- accessor method(s)
 - billingAddress 3-48
 - contactAddress 3-9, 3-17
 - creditCard 3-10, 4-22
 - defaultCreditCard 3-48
 - defaultShippingAddress 3-33
 - OrderAdjustmentPresentation
 - 4-14
 - OrderLine 4-15
 - OrderValue 4-8, 4-14
 - PaymentTransactionValue
 - 4-22
 - shippingAddress 3-9, 4-15
 - shoppingCart 2-30
 - transactionAmount 4-23

B

- badlogin.jsp
 - default Webflow 2-9
 - directory locations 2-9
 - events 2-10
 - form field specification 2-10
 - included JSP templates 2-10
 - namespace 2-9
- business logic
 - newusercreation.jsp 2-29
 - orderhistory.jsp 4-7

C

- changepassword.jsp
 - about 3-51
 - default Webflow 3-53
 - directory locations 3-52
 - dynamic data display 3-55
 - events 3-55
 - form field specifications 3-56
 - included JSP templates 3-54
 - Java import statements 3-53
 - JSP tag libraries 3-52
 - namespace 3-53
 - parameters 3-56
 - pipeline components 3-55
- customer profile 3-1
- customer registration 2-1
- customer self-service 4-1
- customer support contact information xiii

D

- database schema 1-8
- documentation, where to find it xii

E

- editaddressstemplate.inc
 - about 3-30
 - getValidatedValue JSP tag
 - 3-30

- editcctemplate.inc
 - about 3-44
 - getValidatedValue JSP tag 3-45
- editdemographics.jsp
 - about 3-57
 - default Webflow 3-60
 - directory locations 3-59
 - events 3-62
 - form field specifications 3-62
 - included JSP templates 3-61
 - Java import statements 3-60
 - JSP tag libraries 3-59
 - namespace 3-60
 - parameters 3-63
 - pipeline components 3-62
- editprofile.jsp
 - about 3-12
 - default Webflow 3-15
 - directory locations 3-14
 - dynamic data display 3-16
 - events 3-16
 - form field specification 3-19
 - included JSP templates 3-15
 - Java import statements 3-14
 - Java scriptlets 3-18
 - JSP tag libraries 3-14
 - methods/attributes 3-17
 - namespace 3-15
 - parameters 3-19
 - pipeline components 3-16
- Events
 - badlogin.jsp 2-10
 - changepassword.jsp 3-55
 - editdemographics.jsp 3-62
 - editprofile.jsp 3-16
 - login.jsp 2-6
 - newuser.jsp 2-19
 - newusercreation.jsp 2-29
 - newuserforward.jsp 2-32
 - orderhistory.jsp 4-7

- paymenthistory.jsp 4-20
- profileeditaddress.jsp 3-31
- profileeditcc.jsp 3-46
- profilenewcc.jsp 3-39
- usercreationforward.jsp 2-34
- viewprofile.jsp 3-7

I

- input processors 2-35
 - CustomerProfileIP 2-35
 - DeleteCreditCardIP 3-65
 - DeleteShippingAddressIP 3-66
 - LoginCustomerIP 2-37
 - SelectOrderforViewingIP 4-25
 - UpdateAccountInfoIP 3-67
 - UpdateBasicInfoIP 3-68
 - UpdateDemographicInfoIP 3-69
 - UpdatePaymentInfoIP 3-70
 - UpdateShippingInfoIP 3-71

J

- JavaServer Page (JSP)
 - overview 2-2
- JavaServer Page templates
 - changepassword.jsp 3-51
 - editaddresstemplate.inc 3-30
 - editdemographics.jsp 3-57
 - editprofile.jsp 3-12
 - login.jsp 2-3
 - newcctemplate.inc 2-17
 - newdemographictemplate.inc 2-18
 - newuser.jsp 2-11
 - newusercreation.jsp 2-25
 - newuserforward.jsp 2-31
 - orderhistory.jsp 4-4
 - orderstatus.jsp 4-10
 - paymenthistory.jsp 4-18

- profileeditaddress.jsp 3-27
- profileeditcc.jsp 3-41
- profilenewaddress.jsp 3-21
- profilenewcc.jsp 3-35
- usercreationforward.jsp 2-33
- viewprofile.jsp 3-2

L

- login services 2-1
- login.jsp
 - about 2-3
 - default Webflow 2-4
 - directory locations 2-4
 - dynamic data 2-6
 - events 2-6
 - form field specification 2-6
 - included JSP templates 2-5
 - Java import statements 2-4
 - namespace 2-4

N

- namespaces
 - badlogin.jsp 2-9
 - changepassword.jsp 3-53
 - editdemographics.jsp 3-60
 - editprofile.jsp 3-15
 - login.jsp 2-4
 - newuser.jsp 2-15
 - newusercreation.jsp 2-28
 - newuserforward.jsp 2-32
 - orderhistory.jsp 4-6
 - orderstatus.jsp 4-12
 - paymenthistory.jsp 4-20
 - profileeditaddress.jsp 3-29
 - profileeditcc.jsp 3-44
 - profilenewaddress.jsp 3-23
 - profilenewcc.jsp 3-37
 - usercreationforward.jsp 2-34
 - viewprofile.jsp 3-5

- newaddresstemplate.inc
 - about 2-16
 - getValidatedValue JSP tag 2-16
- newcctemplate.inc
 - about 2-17
 - getValidatedValid JSP tag 2-17
- newdemographictemplate.inc
 - about 2-18
 - getValidatedValid JSP tag 2-18
- newuser.jsp
 - about 2-11
 - default Webflow 2-15
 - directory locations 2-14
 - events 2-19
 - form field specification 2-20
 - included JSP templates 2-15
 - Java import statements 2-14
 - JSP tag libraries 2-14
 - namespace 2-15
 - newaddresstemplate.inc 2-16
 - newcctemplate.inc 2-17
 - newdemographictemplate.inc 2-18
 - parameters 2-20
 - pipeline components 2-20
- newusercreation.jsp
 - about 2-25
 - default Webflow 2-27
 - directory locations 2-26
 - dynamic data display 2-29
 - events 2-29
 - included JSP templates 2-28
 - Java import statements 2-27
 - JSP tag libraries 2-27
 - namespace 2-28
- newuserforward.jsp
 - default Webflow 2-31
 - directory locations 2-31
 - events 2-32
 - included JSP templates 2-32

JSP tag libraries 2-31
namespace 2-32

O

orderhistory.jsp
 about 4-4
 default Webflow 4-6
 directory location 4-5
 dynamic data display 4-7
 events 4-7
 included JSP templates 4-6
 Java import statements 4-5
 JSP tag libraries 4-5
 methods/attributes 4-8
 namespace 4-6
 pipeline components 4-7
orderstatus.jsp
 about 4-10
 default Webflow 4-12
 directory locations 4-11
 dynamic data display 4-13
 included JSP templates 4-12
 Java import statements 4-11
 JSP tag libraries 4-11
 methods/attributes 4-14
 namespace 4-12
 pipeline components 4-13

P

paymenthistory.jsp
 about 4-18
 attributes 4-21
 default Webflow 4-20
 directory locations 4-19
 dynamic data display 4-21
 events 4-20
 included JSP templates 4-20
 Java import statements 4-19
 JSP tag libraries 4-19

 methods/attributes 4-22
 namespace 4-20
 pipeline components 4-20
pipeline components 2-39
 EncryptCreditCardPC 2-40
 RefreshOrderHistoryPC 4-26
 RefreshPaymentHistoryPC
 4-27
 RegisterUserPC 2-39
 UpdateBasicInfoPC 3-72
 UpdateDemographicInfoPC
 3-73
 UpdatePasswordPC 3-74
 UpdatePaymentInfoPC 3-75
 UpdateShippingInfoPC 3-76
printing product documentation xii
profileeditaddress.jsp
 about 3-27
 default Webflow 3-29
 directory locations 3-27
 dynamic data display 3-31
 editaddressstemplate.inc 3-30
 events 3-31
 form field specification 3-33
 included JSP templates 3-29
 Java import statements 3-28
 JSP tag libraries 3-28
 methods/attributes 3-32
 namespace 3-29
 parameters 3-33
 pipeline components 3-31
profileeditcc.jsp
 about 3-41
 default Webflow 3-43
 directory locations 3-42
 dynamic data display 3-47
 editccstemplate.inc 3-44
 events 3-46
 form field specifications 3-49
 included JSP templates 3-44
 Java import statements 3-43

JSP tag libraries 3-43
namespace 3-44
parameters 3-49
pipeline components 3-46

profilenewaddress.jsp
about 3-21
default Webflow 3-23
directory locations 3-22
events 3-24
form field specifications 3-25
included JSP templates 3-23
Java import statements 3-23
JSP tag libraries 3-22
namespace 3-23
parameters 3-25

profilenewcc.jsp
about 3-35
default Webflow 3-37
directory locations 3-36
events 3-39
form field specifications 3-39
included JSP templates 3-38
Java import statements 3-37
JSP tag libraries 3-37
namespace 3-37
parameters 3-40
pipeline components 3-39

R

Registering Customers and
Managing Customer
services
customer registration 2-1
customer self-service 4-1
default Webflow 1-3
related information xiii

S

schema

about 1-8
support
technical xiii

U

usercreationforward.jsp
default Webflow 2-33
directory locations 2-33
events 2-34
included JSP templates 2-34
Java import statements 2-33
JSP tag libraries 2-33
namespace 2-34

V

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