

# Using Ajax with WebSphere Commerce

Skill Level: Intermediate

[Marco Deluca \(madeluca@ca.ibm.com\)](mailto:madeluca@ca.ibm.com)

Solution Architect

IBM Canada

[Frances Mullally \(mullally@ca.ibm.com\)](mailto:mullally@ca.ibm.com)

Information Developer

EMC

12 Jul 2006

This tutorial provides a quick series of tasks to use Ajax technology with WebSphere® Commerce. It also provides instructions on how to enable dynamic caching with Ajax using the full test environment.

## Section 1. Introduction

Asynchronous JavaScript and XML (AJAX) is a group of technologies used together as a Web development technique for creating interactive Web applications. With Ajax, instead of loading a Web page, a browser loads an Ajax engine (written in JavaScript), which displays the page the user sees, and then communicates with the server. The Ajax engine allows a user's interaction with the application to happen simultaneously, irregardless of the actions of the server.

When used with WebSphere Commerce, Ajax eliminates the time customers spend staring at a browser window and an hourglass icon, waiting for the server to do something. A customer makes their request through the browser and the Ajax engine responds and gets the information from the WebSphere Commerce Server. This tutorial provides widgets developers can use to enhance the functionality of their WebSphere Commerce v5.6.1 application. The tutorial provides instructions on how to enable dynamic caching with Ajax using the full test environment.

This tutorial shows examples where you can use Ajax with WebSphere Commerce.

The examples display simple and complex scenarios where Ajax is used to serve customers better by eliminating page refresh, providing customers with immediate results:

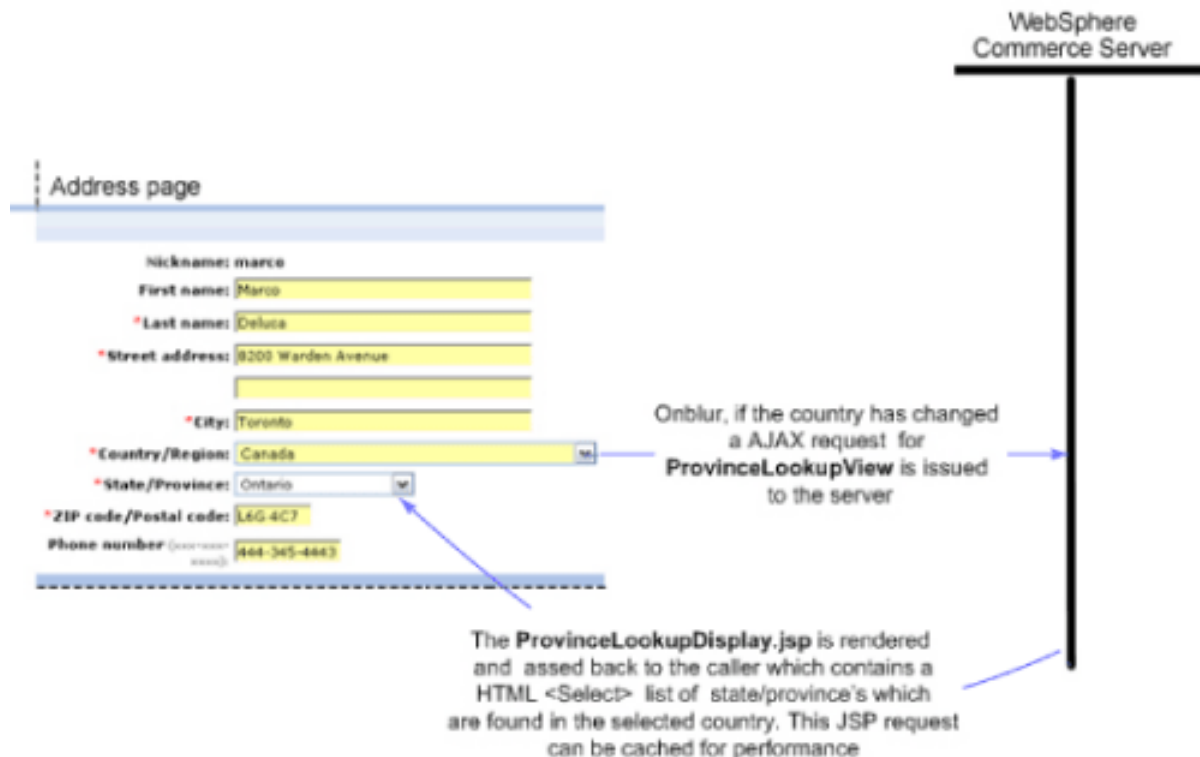
- Scenario one - Menu population
- Scenario two - Credit check
- Scenario three - Quick SKU lookup

Through the scenarios, you will see how display (css) and Java script are used to invoke instantaneous server calls without traditional Web page refreshing. Caching Ajax requests for performance improvement is also examined. Note: The scenarios in this tutorial may not directly apply to your business needs. The aim of this tutorial is to provide a sample package to form the basis for your Ajax implementations.

### **Scenario one - Menu population**

This scenario displays how to populate a menu given user input. When a country or region is selected, the state or province dropdown menu is populated via an Ajax call to the server to retrieve the state or province names within the selected country. If your store only services one or two countries, it is more efficient to bypass an Ajax call altogether and load all state/provinces on the initial page load. Caching is demonstrated in this example to speed up performance. Figure 1 shows the State/Province field instantaneously displaying options once a Country/Region has been selected.

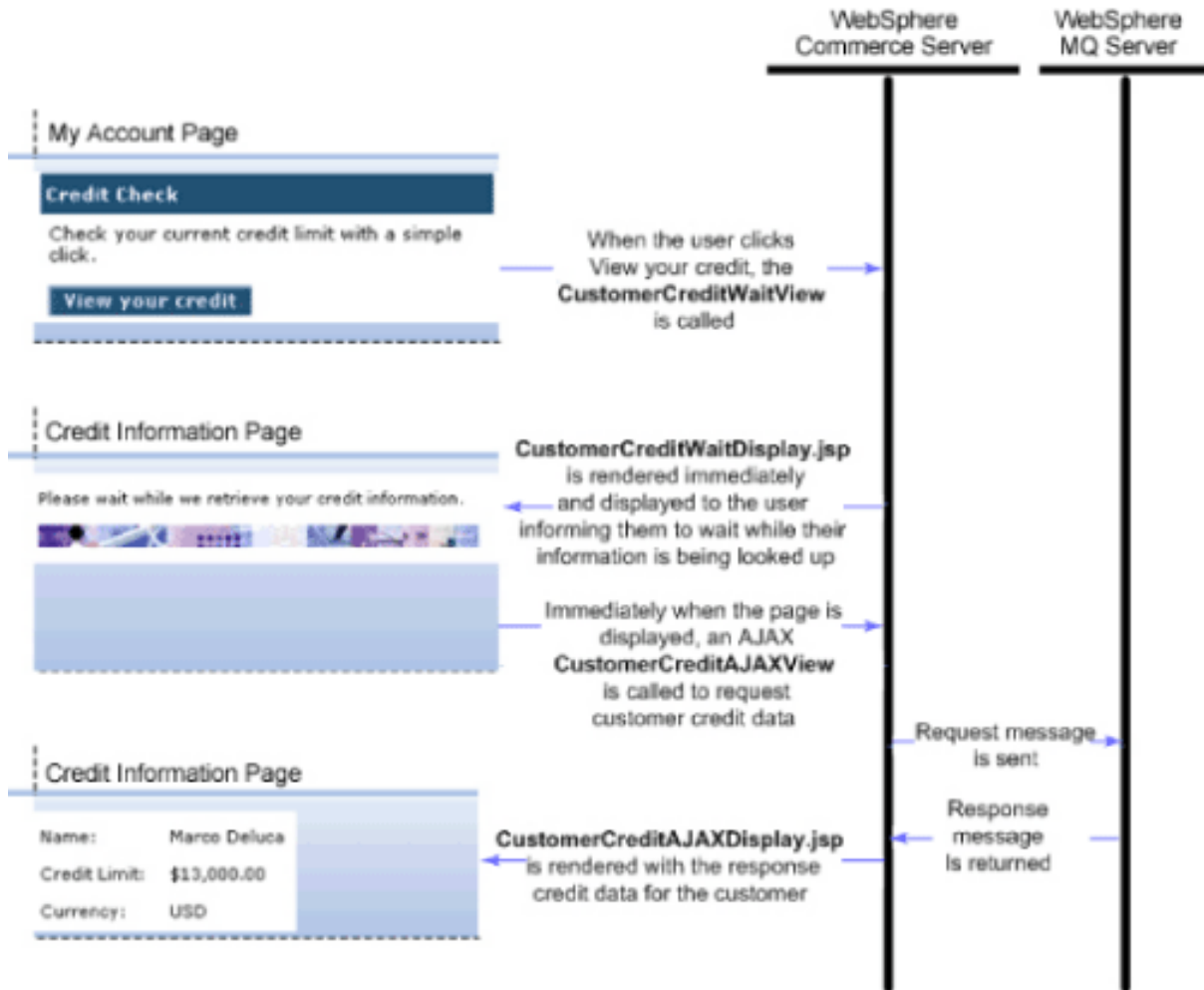
### **Figure 1. Menu population**



### Scenario two - Credit check

This scenario demonstrates a smooth and friendly way to inform your user to wait while you retrieve data for them from a backend system. Web Service and MQ messaging calls are not always instantaneous and require users to wait. In the traditional method, a user clicks a link and waits for a synchronous message to be sent and received before the Web page returns to the user. During busy times, requests may take longer than expected, which can prompt users to click the Refresh button on their browser. This causes the server to handle yet another request, and slows down the process. In this scenario, the Web page is redrawn and instantaneously informs the user to wait. Next, a request is issued to the MQ server to get data (in this case, Credit data) for a particular user. When the result returns, it is instantly displayed. If a set timeout or error occurs, an error message and refresh button are displayed to the user. Figure 2 shows what a customer will see when their credit check information is being retrieved from the server.

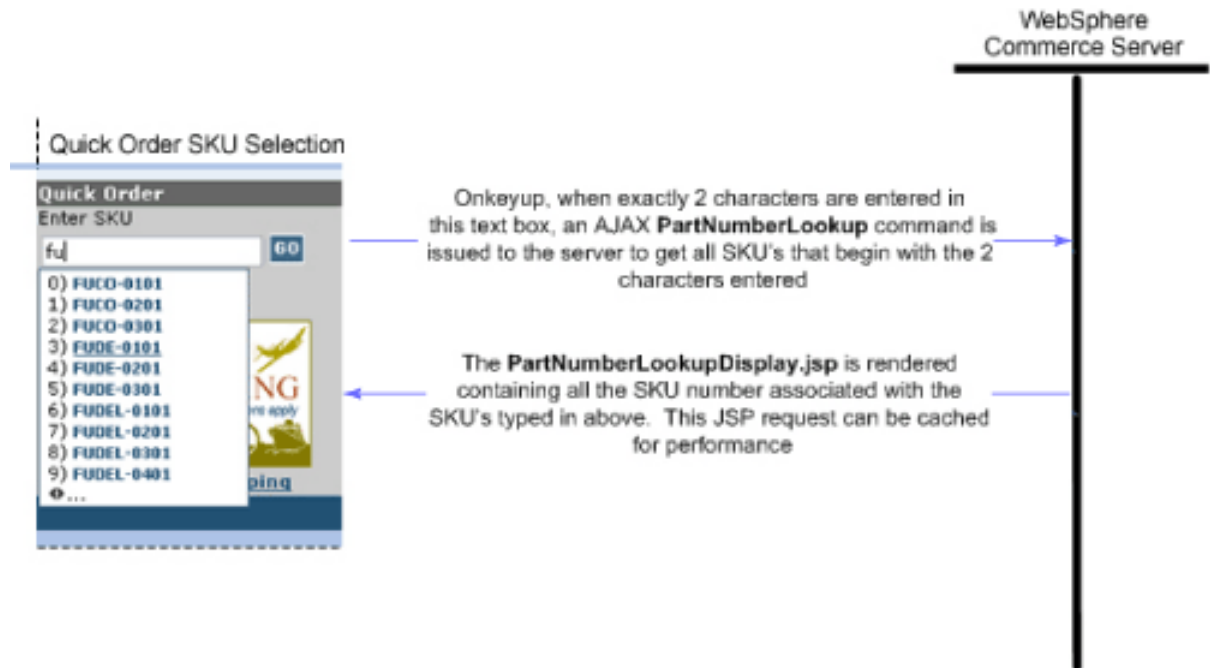
**Figure 2. Credit check**



### Scenario three - Quick SKU lookup

This scenario is more complex and applicable to smaller catalogs. In this scenario, the user types in two characters (configurable #) into the SKU text area, and an Ajax call is made to assist the user in finding the SKU. Each extra character entered narrows the list of SKUs without further requests being made to the server. These lookup results are cached for performance. Figure 3 shows the options customers will see when they enter in partial SKU number.

**Figure 3. Quick SKU lookup**



## Section 2. Deploying the assets

Deploying the assets involves moving all of the Ajax assets onto your system. To move the Ajax assets onto your system, perform the following steps:

1. Import the commands and beans into your WebSphere Commerce system.
2. Load the database assets.
3. Load the access control policies.
4. Load the credit check request and response information.

### Importing Ajax assets into WebSphere Commerce

The first step to using Ajax with WebSphere Commerce involves importing all of the commands, beans, and store assets into your WebSphere Commerce Developer workspace. To import the store assets, perform the following steps:

1. Download the [AJAX.zip](#) to your machine where WebSphere Commerce

resides.

2. Extract the files into a directory of your choosing.
3. In WebSphere Studio Application Developer, open the Java perspective, **Window -> Open Perspective -> Java**.
4. Right-click the **Stores** and select **Import**. The Import wizard opens.
5. From the Import Select list, select **File system** to import resources from the local file system and click **Next**.
6. Click **Browse** and navigate to the sample code, and select the following directory: *yourDirectory*\Stores, where *yourDirectory* is the directory into which you downloaded the tutorial.
7. Click **OK**.
8. Click **Select All**.
9. Click **Finish**.

To import the commands and beans, perform the following steps:

1. Expand the WebSphereCommerceServerExtensionsLogic project.
2. Right-click the **src** folder under the Extensions Logic project, and select **Import**. The Import wizard opens.
3. From the Import Select list, select **File system** to import resources from the local file system and click **Next**.
4. Click **Browse** and navigate to the sample code, and select the following directory:  
*yourDirectory*\WebSphereCommerceServerExtensionLogic\src,  
where *yourDirectory* is the directory into which you downloaded the tutorial.
5. Click **OK**.
6. Click **Select All**.
7. Click **Finish**.

### Loading database assets

After you have imported all of the commands and beans into your system, you must

load the database assets. The following steps will load the AJAX.sql script located in the *yourDirectory*\db\sql directory. These scripts load the Ajax resources for the SKU lookup, province and country codes, and credit check into the database tables. To run the AJAX.sql script:

1. Launch a command prompt.
2. Go to the *WC\_installdir*\bin directory, where *WC\_installdir* is the directory where you installed WebSphere Commerce.
3. If you are using the Cloudscape database:
  1. Type: `ij`. This loads your Cloudscape database.
  2. Type: `connect '../db/mall'`;
  3. Type: `run 'yourdirectory\db\sql\AJAX.sql'`;
  4. Type: `quit`;
4. If you are using a DB2 database:
  1. Open a DB2 Command Window.
  2. Type: `db2 connect to database_name user user_name using user_password`
  3. Type: `db2 -tvf "yourdirectory\db\sql\AJAX.sql"`
  4. Type: `db2 terminate`

### Loading access control policies

After the commands, beans, and database assets have been loaded into your system, you must load the access control policies on your system. These policies determine the Resource Groups, Resource Categories, and Policy Groups that will be affected by the SKU lookup function. To load the access control policies, do the following steps:

1. Copy the *yourDirectory*\xml\policies directory to your *WC\_installdir*\xml directory.
2. From the command prompt, go to the *WC\_installdir*\bin directory.
3. If you are using the Cloudscape database type: `acpload AJAX_AC_Policies.xml`

4. If you are using a DB2 database type: `acpload database_name user_name user_password AJAX_AC_Policies.xml shemea_name`
5. Check the log file under the `WC_installdir\logs` directory, `messages.txt`, to make sure that the access policies were loaded successfully.

### Loading credit check request and response information

If you are going to be using the credit check scenario, you must load the user template and credit account information onto your system. The user template information defines the inbound mapping xml to name value pair mapping for the CustomerCreditInfo command. The CreditAccount.dtd file defines the credit account message. CreditAccount.xml is a sample of how the message looks.

To copy the user template to your machine, do the following steps:

1. Open the file `yourdirectory\xml\messaging\user_template_merge.xml` in a text editor.
2. Copy the contents between the `<ECTemplate>` tags into the `WC_installdir\xml\messaging\user_template.xml` file.
3. Save the file.

To copy the credit account message definition file, copy the `yourdirectory\xml\messaging\CreditAccount.dtd` file to your `WC_installdir\xml\messaging` directory.

---

## Section 3. Integrating Ajax assets with the ConsumerDirect store

Now that all of the assets are copied over, you must merge new information into the existing ConsumerDirect store assets. Merge the new information by performing the following tasks:

1. Update your instance configuration.

2. Merge your store JSP files.

## Updating your instance configuration

You must update your instance configuration to recognize the CreditAccount.dtd file:

1. Open the `WC_install_dir\conf\xml\config.xml` directory on your WebSphere Commerce machine.
2. Add the CreditAccount.dtd file name to the `Messaging\EcInboundMessageDtdFiles = " "` attribute.

## Merging your store JSP files

After you have completed all of the previous tasks, you are ready to integrate the assets you previously copied over to your ConsumerDirect store.

**Note:** The following steps will overwrite your JSP files. If you do not wish to overwrite your JSP files, conduct a compare between the files provided and your JSPs and insert the additional code where necessary.

To integrate the assets:

1. Copy the files in `yourdirectory\Stores_Original_JSP_Modifications\ProvinceState_JSP_` into your `WC_installdir\workspace\Stores\Web Content\ConsumerDirect` directory.
2. Copy the files in `yourdirectory\Stores_Original_JSP_Modifications\CreditCheck_JSP_Mo` into your `WC_installdir\workspace\Stores\Web Content\ConsumerDirect` directory.
3. Copy the files in `yourdirectory\Stores_Original_JSP_Modifications\PartNumber_JSP_Mod` into your `WC_installdir\workspace\Stores\Web Content\ConsumerDirect` directory.

## Adding information for the Quick SKU lookup scenario

If you want to use the Quick SKU lookup functionality, you must add the following two lines to all of your JSP pages with a `<head>` tag to get this functionality to work:

1. Open your JSP file and locate the `<head>` tag.

2. Insert the following code between the <head> and </head> tags.

```
<link rel="stylesheet" href="<c:out value="\${jspStoreImgDir}css/AJAX.css" />"
type="text/css" />

<script type="text/javascript" language="javascript" src="<c:out
value="\${jspStoreImgDir}"
/>javascript/AJAX_Util.js"></script>
```

---

## Section 4. Testing Ajax

To test the Ajax functionality with WebSphere Commerce:

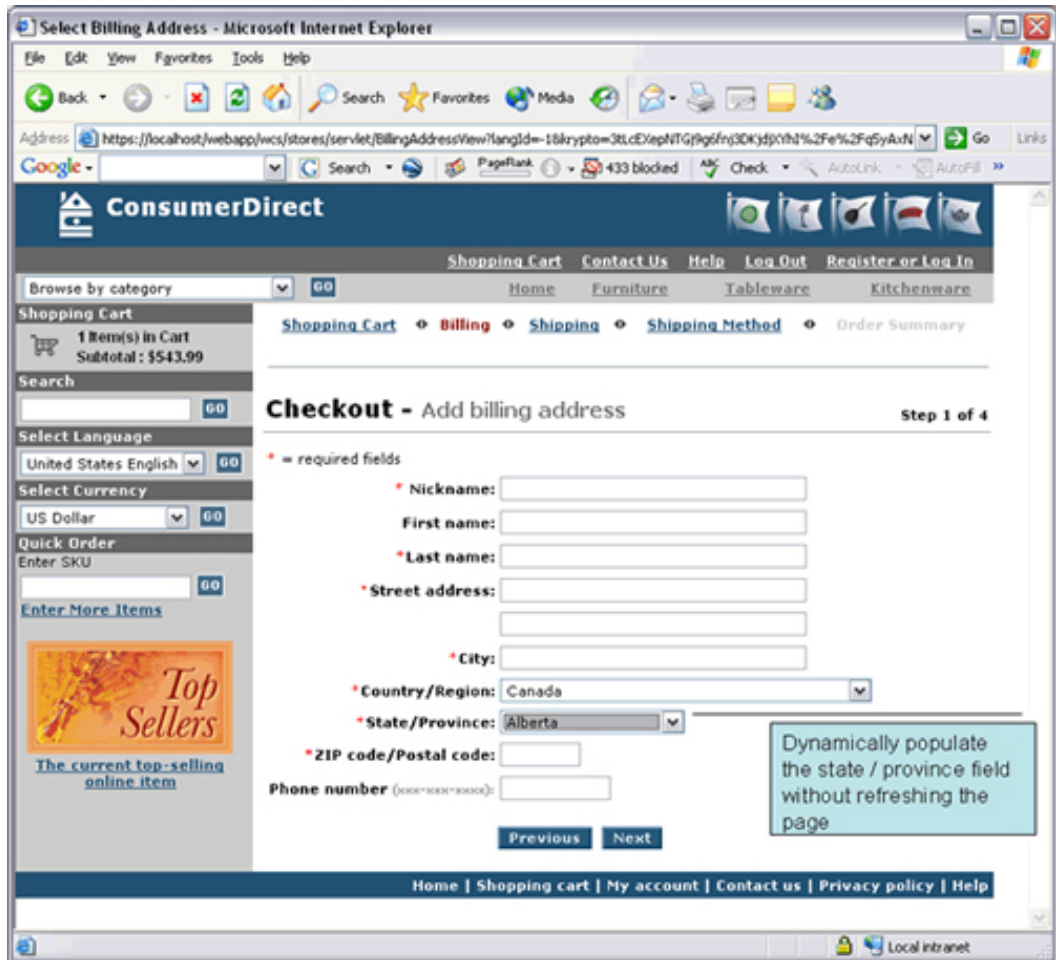
1. Start the WebSphere Commerce development environment.
2. Open the store by typing the following URL into your browser:  
`http://localhost/webapp/wcs/stores/servlet/ConsumerDirect/index.jsp.`

### Testing scenario 1 - State/Province menu population

To test the State/Province menu population scenario:

1. Add a product to your shopping cart.
2. Click **Checkout**. The Checkout- Add billing address page displays.
3. Select **Canada** or **United States** in the Country/Region field.
4. Tab off the field. The State/Province field should be populated as shown in Figure 4.

#### Figure 4. Checkout page



## Testing scenario 2 - Credit check

To test the Credit check scenario:

1. Click **Register** or **Log In**.
2. Register a customer or log in with an existing customer.
3. Once you are logged in, you will be on the My Account page. You will see the new option called Credit Check as shown in Figure 5.
4. Click **View your credit**. You will see a status message as shown in Figures 6,7, and 8.

### Figure 5. My account page

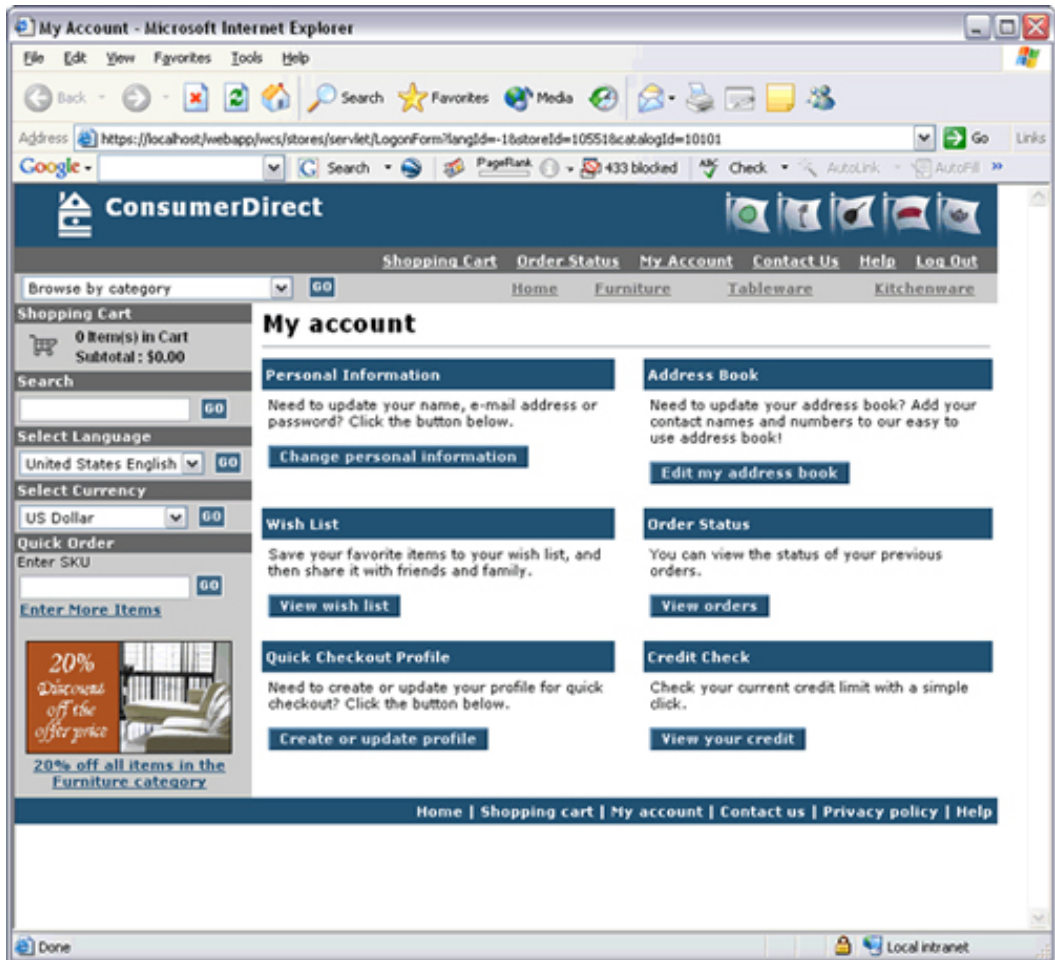


Figure 6. Credit Information page with waiting message

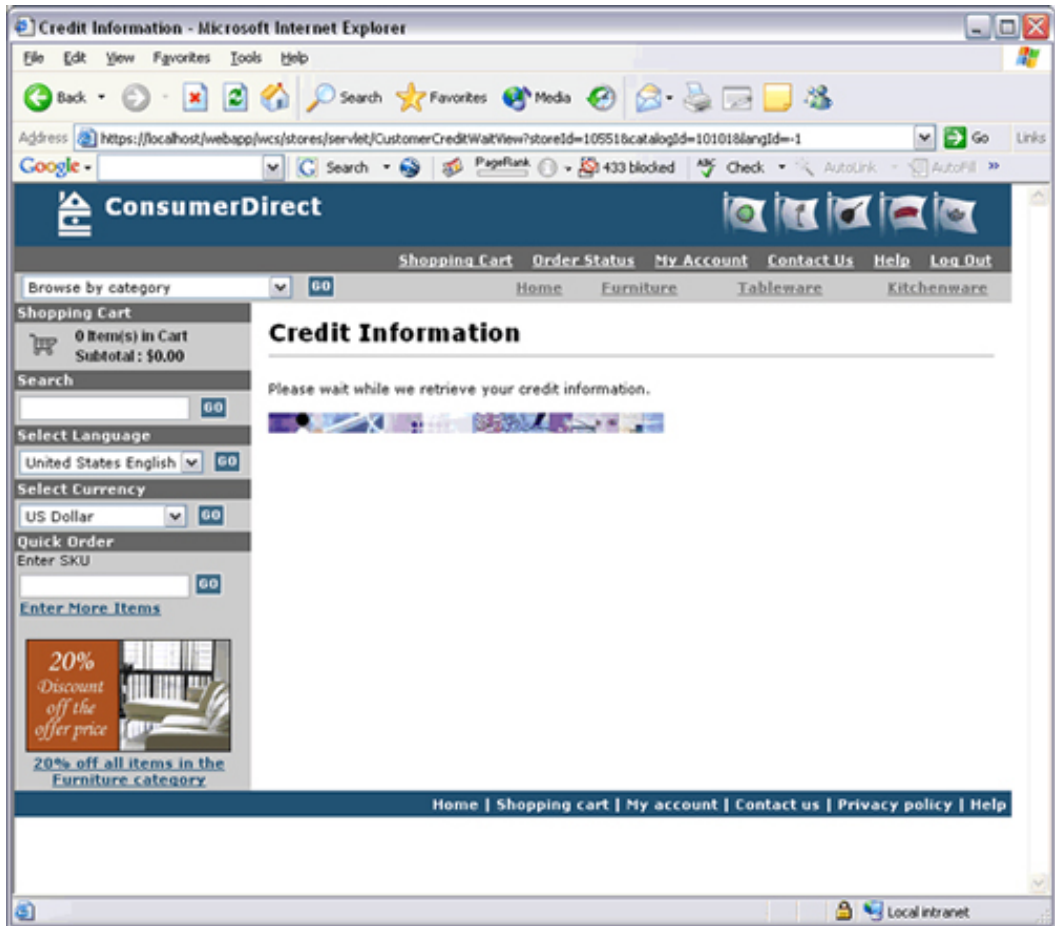


Figure 7. Credit information page with status message

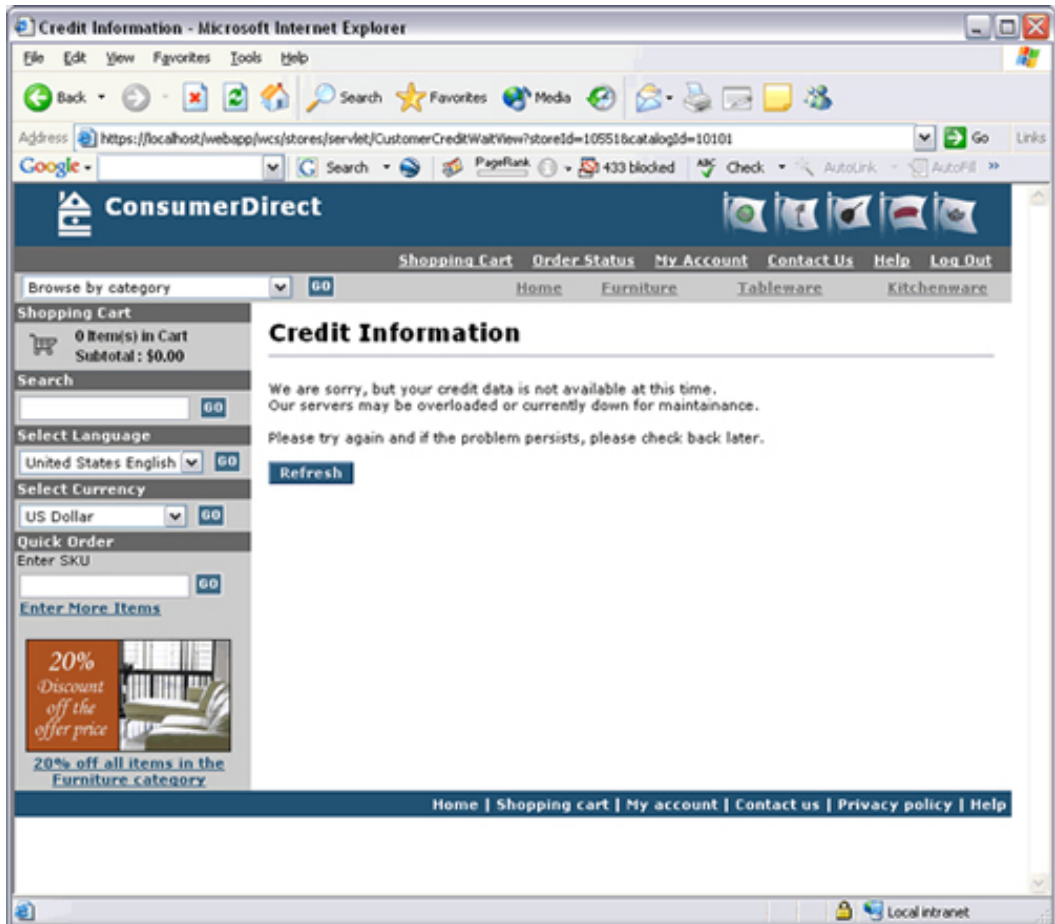
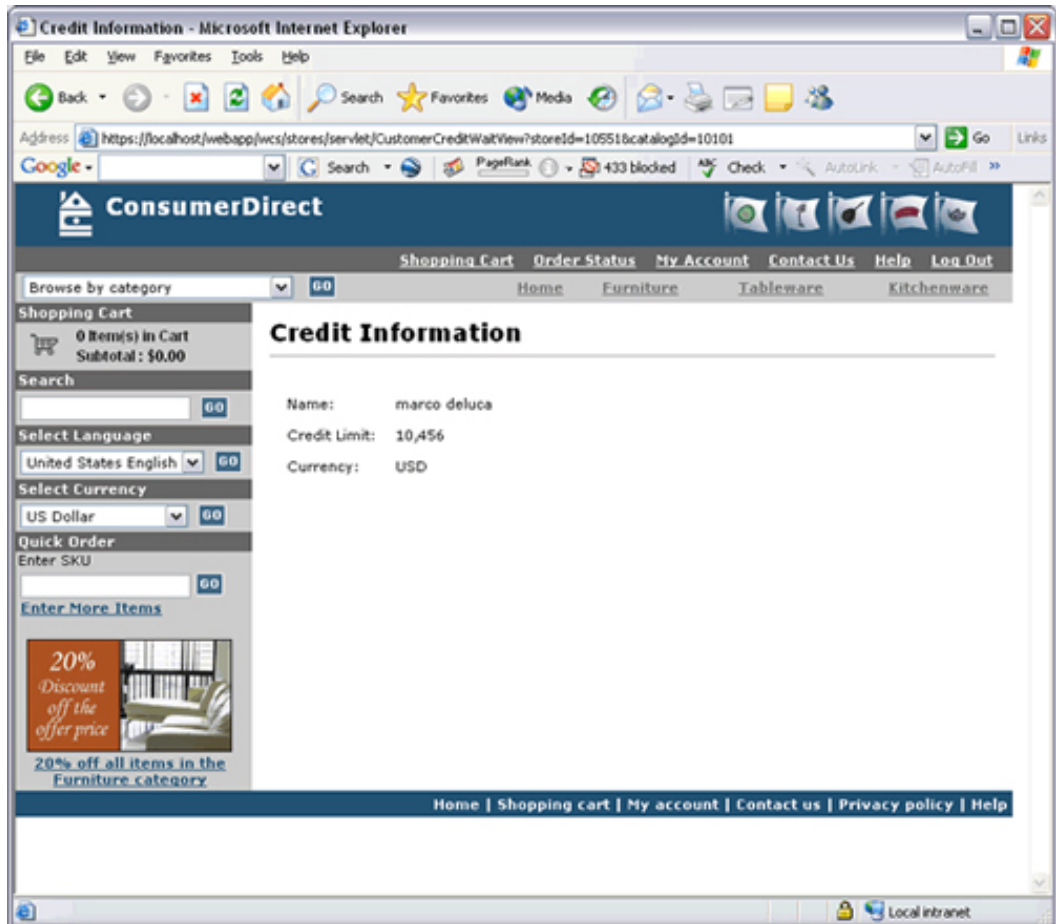


Figure 8. Credit Information page with results

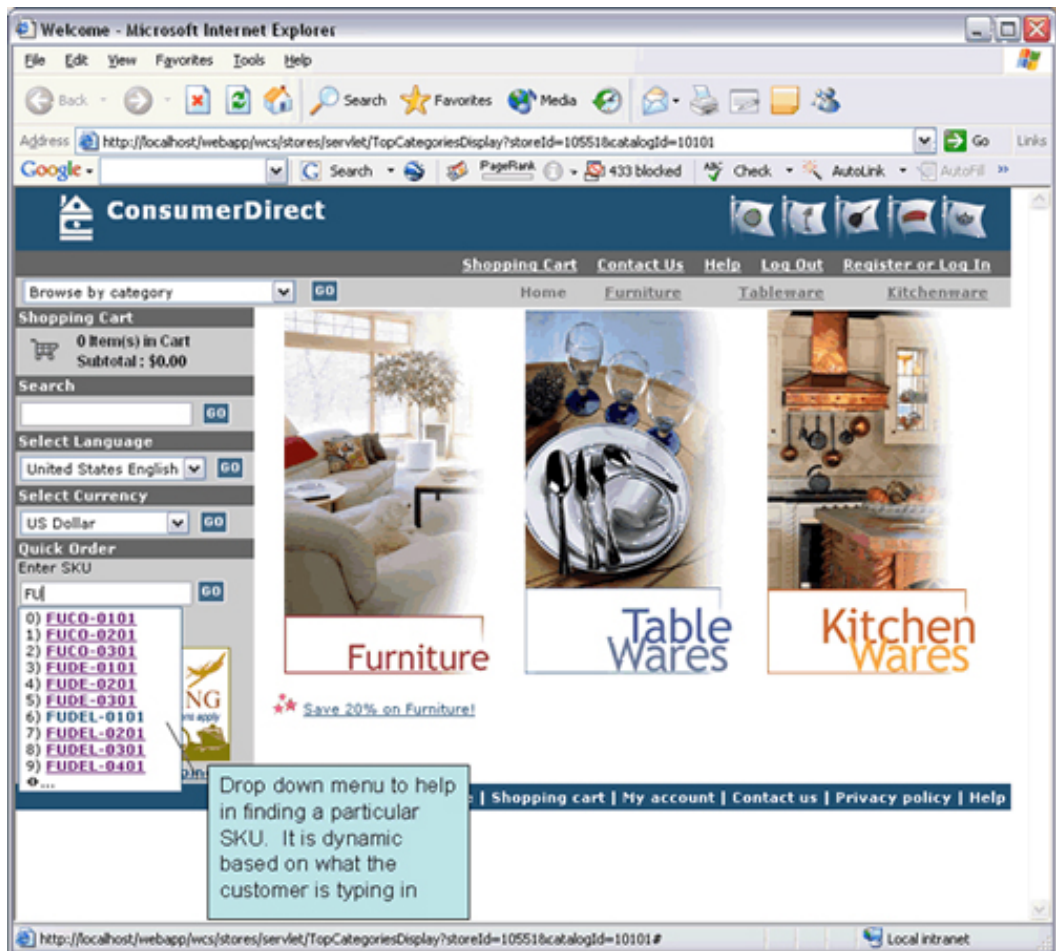


### Testing scenario 3 - Quick SKU lookup

To test the Quick SKU lookup scenario:

1. Open the WebSphere Commerce Accelerator.
2. Select the ConsumerDirect starter store.
3. Go to the **Store -> Change Flow** menu, and select the **Orders** tab.
4. Check the **Quick order** box.
5. Click **OK** to close the dialogue.
6. Go to the ConsumerDirect starter store. You see the Quick Order section.
7. In the Enter SKU field, type in **FU**. A drop-down menu with SKUs displays as shown in Figure 9.

**Figure 9. ConsumerDirect home page**



## Section 5. Enabling dynamic caching on full test environment (optional)

Caching the servlet or JSP file results improves application performance. WebSphere Application Server consolidates several caching activities, including servlets, Web services, and WebSphere commands into one service called the dynamic cache. These caching activities work together to improve application performance, and share many configuration parameters, which are set in an application server's dynamic cache service.

You can use the dynamic cache to improve the performance of servlet and JSP files by serving requests from an in-memory cache. Cache entries contain servlet output, results of servlet execution and metadata.

To enable dynamic caching when using Ajax, you must be on the full test environment. It is also recommended that you have tested Ajax on your system before doing this step.

**Note:** If you are testing the Ajax functionality, you can skip this step. However, it is required when you move to production, as it fully enhances the performance capabilities of Ajax with your WebSphere Commerce system.

To enable dynamic caching on your WebSphere Commerce system, see [Enabling and disabling Dynamic Caching in WebSphere Commerce Developer](#) in the WebSphere Commerce Information Center.

### Merging the cachespec.xml file

You must merge the information in the cachespec.xml file in the Ajax directory into the WebSphere Commerce cachespec.xml file. To merge the information in the two files, perform the following steps:

1. Open the `AJAX\cachespec.xml` in a text editor.
2. Merge with your existing cachespec.xml file under  
`install_dir\workspace\Stores\Web  
Content\WEB-INF\cachespec.xml`.

**Note:** If this file does not exist, copy the `AJAX\cachespec.xml` file into your WEB-INF directory.

---

## Section 6. Conclusion

By performing a quick series of tasks, you can use Ajax to add courtesy messages to customers for credit checks, enable customers to easily find products by performing quick SKU searches, and populate menus instantaneously. Customers can make their request through the browser and the Ajax engine will respond and get the information from the WebSphere Commerce Server. You can use Ajax in many other areas of WebSphere Commerce to help your On Demand business thrive, and remain a top competitor in your retail marketplace.

## Downloads

| Description                   | Name     | Size | Download method      |
|-------------------------------|----------|------|----------------------|
| Sample code for this tutorial | AJAX.zip | 65KB | <a href="#">HTTP</a> |

[Information about download methods](#)

## Resources

- Visit the [WebSphere Commerce Information Center](#) for more information on all of the features you can use with WebSphere Commerce.
- Visit the [developerWorks WebSphere Commerce zone](#) for technical and how-to information.
- Participate in [WebSphere Commerce forums](#).

## About the authors

Marco Deluca



**Marco Deluca** is a Business Solution Architect with the WebSphere Commerce Solution Enablement team and has five years of experience developing and creating solutions for WebSphere Commerce.

---

Frances Mullally

**Frances Mullally** is an Information Developer with WebSphere Commerce. She has experience in the electronic commerce, retail, and health care industries as a technical writer and trainer.