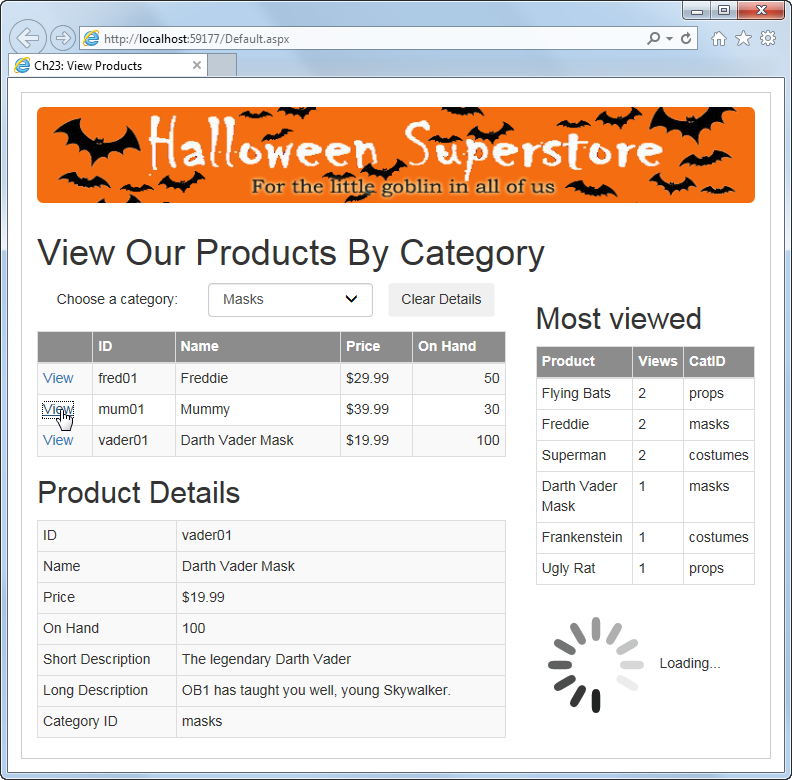
Chapter 23-Part 1 Test the View Products application and add an UpdateProgress control

In this exercise, you’ll test the View Products page that was presented in this chapter. That will give you a good idea of how the Ajax controls work. Then, you’ll simulate a service delay and add an UpdateProgress control to show that the data is being loaded.



Test the application

1. Open the XEx23ViewProducts web application that’s in your blackboard directory.
2. Run the application and click the View link for one or more of the products in the GridView control. Note that the product details appear below the GridView control, and another control appears to its right, without the page posting back to the server. Also note that sometimes there’s a small but noticeable delay.
3. Select a new category from the drop-down list, and notice that new products appear in the GridView control, again without the page posting back to the server. When you’re done testing, close the browser.

Simulate a service delay

1. Review the code in the ProductViewList.cs file, located in the Models folder. Uncomment the first line of code in the Add method, which looks like this:

//System.Threading.Thread.Sleep(2000);

That will cause an additional delay of 2 seconds every time you click the View link.

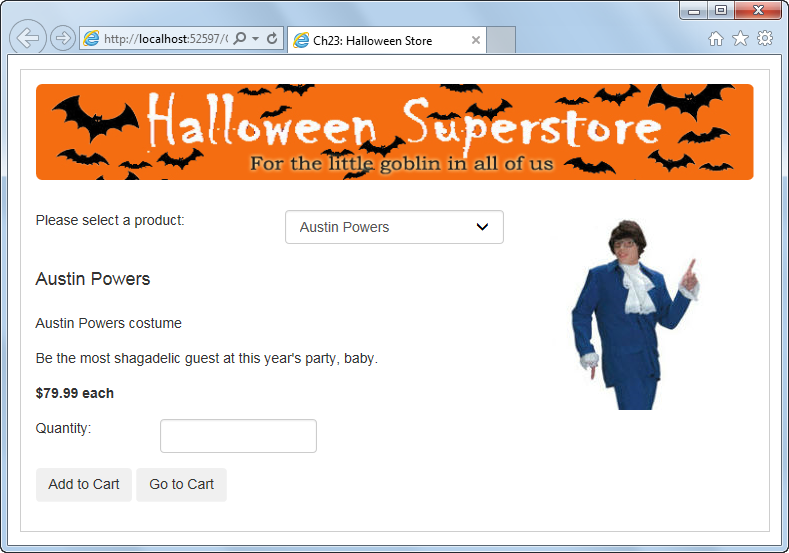
1. Run the application, click on the View link for one or more products, and notice the long delay each time the page is updated by Ajax. Then, close the browser.

Add an UpdateProgress control

1. Display the aspx code for the Default page, and scroll down to the last div element within the div element that defines the Bootstrap row for the page. It’s near the bottom of the page. Notice that this div element contains an UpdatePanel control with a ContentTemplate element, and the ContentTemplate element contains a GridView control.
2. Add an UpdateProgress control below the GridView control in the ContentTemplate element, as described in figure 23-10.
3. Add the HTML you want displayed during delays to the ProgressTemplate element of the UpdateProgress control. That HTML should consist of the spinner.gif image in the Images folder as well as text, as shown in the second part of figure 23-13.
4. Run the application again, and see that the HTML you added to the ProgressTemplate element in the UpdateProgress control appears during the 2-second delay.
5. Close the browser and comment out the line of code shown in step 4 to remove the delay. Then, give the page one last test.

Chapter 23-Part 2 Ajax enable the Halloween Store application

In this exercise, you’ll Ajax enable the Halloween Store application by adding a ScriptManager control and some UpdatePanel controls.



Test the Order page

1. Open the XEx23HalloweenStore web application that’s in your blackboard directory.
2. Run the application and select new products from the drop-down list on the Order page. Notice that the entire page refreshes each time you select a new product.
3. Add a product to the cart. Then, on the Cart page, click on the Empty Cart and Check Out buttons, and notice that the entire page refreshes each time. When you’re done, close the browser.

Add a ScriptManager and UpdatePanel controls

1. Add a ScriptManager control at the top of the master page, within the form element.
2. Add an UpdatePanel control at the top of the Order page, within the Content control. Then, add a ContentTemplate element to the UpdatePanel control. Finally, move the aspx code for the first row so it’s within the ContentTemplate element.
3. Run the application, and select new products from the drop-down list on the Order page. This time, each new product is displayed without a page refresh.
4. Click on the Go to Cart button to go to the Cart page, then click on the Continue Shopping button to go back to the Order page. Notice that the entire Order page refreshes. That shows that even an Ajax enabled page does a complete load each time it’s displayed from another page.
5. Select new products from the drop-down list to see that subsequent operations don’t cause the page to refresh. Then, close the browser.
6. Add an UpdatePanel control at the top of the Cart page, within the Content control. Then, add a ContentTemplate element to the UpdatePanel control. Finally, move the aspx code for the first row so it’s within the ContentTemplate element.
7. Run the application and add a product to the cart. Then, on the Cart page, click on the Remove Item button (without selecting an item) and then on the Empty Cart button. Note that the actions of these buttons occur without a page refresh.
8. Click on the Check Out button, and note that this still causes the entire page to refresh. Then, close the browser.

Add a trigger to an UpdatePanel control

1. Add a Triggers element below the ContentTemplate element in the UpdatePanel control on the Cart page. Then, add an AsyncPostBackTrigger control to the Triggers element, as shown in figure 23-8. The ControlID attribute of the control should be set to “btnCheckOut”.
2. Repeat steps 10 and 11. Now, all the actions occur without a page refresh.
3. When you’re through experimenting, close the browser and the project.