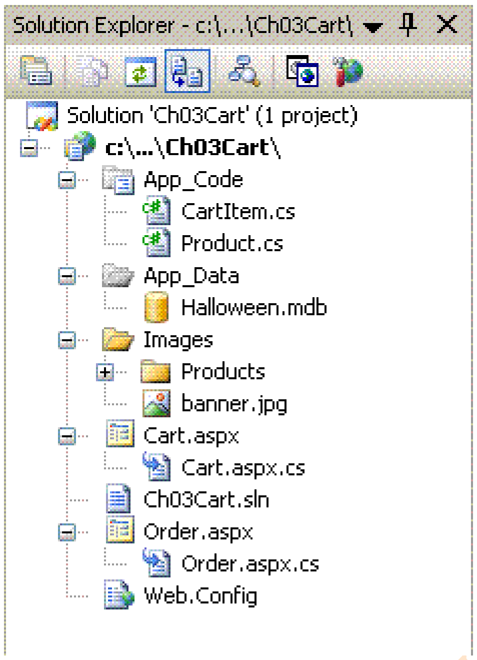
Multi-page page ASP.NET application

For that lecture we use a *Ch03Cart* web application.

The files and folders used by *Ch03Cart* application

The Solution Explorer for the *Ch03Cart* application



Special folders used in ASP.NET 2.0 applications

Folder	Description
App_Code	Non-page class files that are compiled together to create a single assembly
App_Data	Database files used by the application
App_Thems	Themes used by the application
Bin	Compiled code used by the application

The business classes in the *Ch03Cart* application

The Ch03Cart application required two business classes named *Product* and *CartItem*.

Two ways to open the Add New Item dialog box

- Right-click the *App_Code* folder in the Solution Explorer if it already exits, and then chose *Add New Item* from the shortcut menu.
- Click on the *App_Code* folder in the Solution Explorer to select it, and then choose the Website -→ *Add New Item* command.

How to add a new class to a web site

- From the *Add New Item* dialog box, select the *Class* template, name the class, and click the *Add* button. Visual Studio will create the class with namespace and class blocks and an empty constructor.
- If you try to add a class directly to the project instead of the *App_Code* folder, Visual Studio will warn you that the class should be placed in the *App_Code* folder and ask you if you'd like to place the class in the folder. If you click the *Yes* button, Visual Studio will create the *App_Code* folder if necessary and place the class in this folder.

How to add an existing class to a we site

- To add a class from another project to a web site, right-click the *App_Code* folder in the Solution Explorer and select *Add Existing* Item. Then locate the class file you want to add, select it, and click the Add button. The file is copied to your project.

How to use a class that's part of a class library

- Right-click the project in the Solution Explorer and select *Add Reference* from the shortcut menu. Click the *Browser* tab in the dialog box that's displayed, and then locate and select the dll file for the class library you want to use. The class library is added to the project's *Bin* folder, which is created if it doesn't already exist. To use the classes in the class library without qualification, add a using statement for the class library.

How to redirect or transfer to another page

The Transfer method of the *HttpServerUtility* class

Method	Description
Transfer(URL)	Terminates the execution of the current page and transfers controls to the page at the specified URL

Code that transfers control to another page

Server.Transfer("Cart.aspx")

The Redirect method of the HttpResponse class

Method	Description
Redirect(URL)	Redirects the client to the specified URL and terminates the execution of the current page

Code that redirects the client to another page

Response.Redirect("Cart.aspx")

When you use the *Redirect* method, the server sends a special message called an ***HTTP redirect message*** to the browser. When the browser receives this message, it sends an HTTP request to the server to request the new page. The server then processed the new page and sends it back to the browser. Because this involves a round trip to the browser, it's less efficient than the Transfer method.

Cross-page posting

This is a third way to transfer to a different web page. To use cross-page posting, you specify the URL of another page in the *PostBackUrl* property of a button control. Then, when the user clicks the button, an HTTP Post message that contains the URL specified by the *PostBackUrl* property is sent back to the server. As a result, the page with that URL is loaded and executed instead of the page that was originally displayed.

The PostBackUrl property of the Button control

Property	Description
PostBackUrl	Specifies the URL of the page that should be requested when the user clicks the button

The Page class

Property	Description
PreviousPage	Returns a *Page* object that represents the previous page
Method	**Description**
FindControl(id)	Returns a *Control* object with the specified id. If the control is a text box, you can cast the *Control* object to *TextBox*, then use the *Text* property to access the data

The aspx code for a button that posts to a different page

```
<asp:Button ID="btnCart" runat="server" CausesValidation="False"
        PostBackUrl="~/Cart.aspx" Text="Go to Cart" />
```

Code that retrieves data from the previous page

```
protected void Page_Load(object sender, EventArgs e)
{
   if (PreviousPage != null)
   {
      TextBox txtQuantity = (TextBox) PreviousPage.FindControl("txtQuntity");

      lblQuantity.Text = txtQuantity;
   }
```

}

Description

- ***Cross-page*** posting lets you use the *PostBackUrl* property of a button to specify the page that should be requested when the user clicks the button.
- When you post back to another page, the previous page is available via the *PreviousPage* property. Then you can use the *FindControl* method to retrieve data entered by the user.
- In general, cross-page posting is more efficient than the *Server.Transfer* and *Response.Redirect* methods. However, cross-page posting makes it more difficult to retrieve data from the original page.
- If you don't need to retrieve data from the original page, cross-page posting is clearly better than the *Server.Transfer* and *Response.Redirect* methods.

When you code within a *Transfer* or redirect methods, an absolute URL lets you display a page at another web site.

Statements that absolute and relative URLs

Response.Redirect("http://www.utech.edu.jm/Default.aspx")
Response.Redirect("http://www.utech.edu.jm/Studets/Search.aspx")

Statements that use relative URLs that are based on the current directory

Response.Redirect("Checkout.aspx")
Response.Redirect("Login/Register.aspx")

Statements that use relative URLs that navigate up the directory structure

Response.Redirect("../Register.aspx")
Response.Redirect("../../Register.aspx")
Response.Redirect("/Register.aspx")
Response.Redirect("/Login/Register.aspx")

Server control attributes that use URLs that are based on the root directory of the current web site

PostBackUrl = "~/Cart.aspx"
ImageUrl = "~/Image/banner.jpg"

How to use a data sources

To connect to a database and work with its data, ASP.NET 2.0 offers a control called a ***data source***. ASP.NET provides several data sources controls in the *Data* group of the Toolbox. The simplest of these controls is *AccessDataSource*, which is designed to retrieve data from a Microsoft Access database file.

How to create an Access data source

1. In the Web Forms Designer, open the *Data* group of the Toolbox and drug the *AccessDataSource* control to the form.
2. Select *Configure Data Source* from the smart tag menu of the data source control to display the *Configure Data Source* dialog box.
3. Identify the Access database that you want to use in the *App_Data* folder and click *Next*. Then, complete the *Configure Data Source* wizard.

Before you can create an Access data source, you must add the Access database file to the *App_Data* folder. The *Ch03Cart* application uses an *AccessDataSource* control to provide the list of products that's displayed in the drop-down list. This control reads data from a Microsoft Access database file. Although data source controls are visible in the Web Forms Designer, they don't render any HTML to the browser. As a result, they aren't visible when the application runs.

The aspx code for an Access data source control

```
<asp:AccessDataSource ID="AccessDataSource1" runat="server"
  DataFile="~/App_Data/Halloween.mdb"
  SelectCommand="SELECT [ProductID], [Name], [ShortDescription],
                        [LongDescription],
                        [ImageFile],  [UnitPrice]
                 FROM [Products] ORDER BY [Name]">
</asp:AccessDataSource>
```

Once you've created a data source, you can bind a drop-down list to it as shown below:

```
<asp:DropDownList ID="ddlProducts" runat="server" Width="150px"
  DataSourceID="AccessDataSource1" DataTextField="Name"
  DataValueField="ProductID" AutoPostBack="True">
</asp:DropDownList>
```

Attributes for binding a drop-down list

Attribute	Description
DataSourceID	The *ID* of the data source that the drop-down list should be bound to
DataTextField	The name of the data source field that should be displayed in the drop-down list
DataValueField	The name of the data source field whose value should be returned by the *SelectedValue* property of the drop-down list

The retrieving date and manipulate it in .NET technology will be shown later in that course in details, because it is one of the corn issues. In this lecture we will show the easiest way how to do it.

Code that gets product information for the selected product

```
DataView productsTable = (DataView)
        AccessDataSource1.Select(DataSourceSelectArguments.Empty);

productsTable.RowFilter = "ProductID = '" + ddlProducts.SelectedValue + "'";

DataRowView row = (DataRowView) productsTable[0];

Product p = new Product();

p.ProductID = row["ProductID"].ToString();
p.Name = row["Name"].ToString();
p.ShortDescription = row["ShortDescription"].ToString();
p.LongDescription = row["LongDescription"].ToString();
p.UnitPrice = (decimal) row["UnitPrice"];
p.ImageFile = row["ImageFile"].ToString();
```

The Select method of the *AccessDataSource* class

Method	Description
Select(selectOptions)	Returns an *IEnumerable* object that contains the rows retrieved from the underlining Access database. To get all the rows, *selectOptions* parameter should be *DataSourceSelectArguments.Empty*.

The *DataView* class

Property	Description
RowFilter	A string that is used to filter the rows retrieved from the Access database
Indexer	**Description**
[index]	Returns a *DataRowView* object for the row at the specified index position

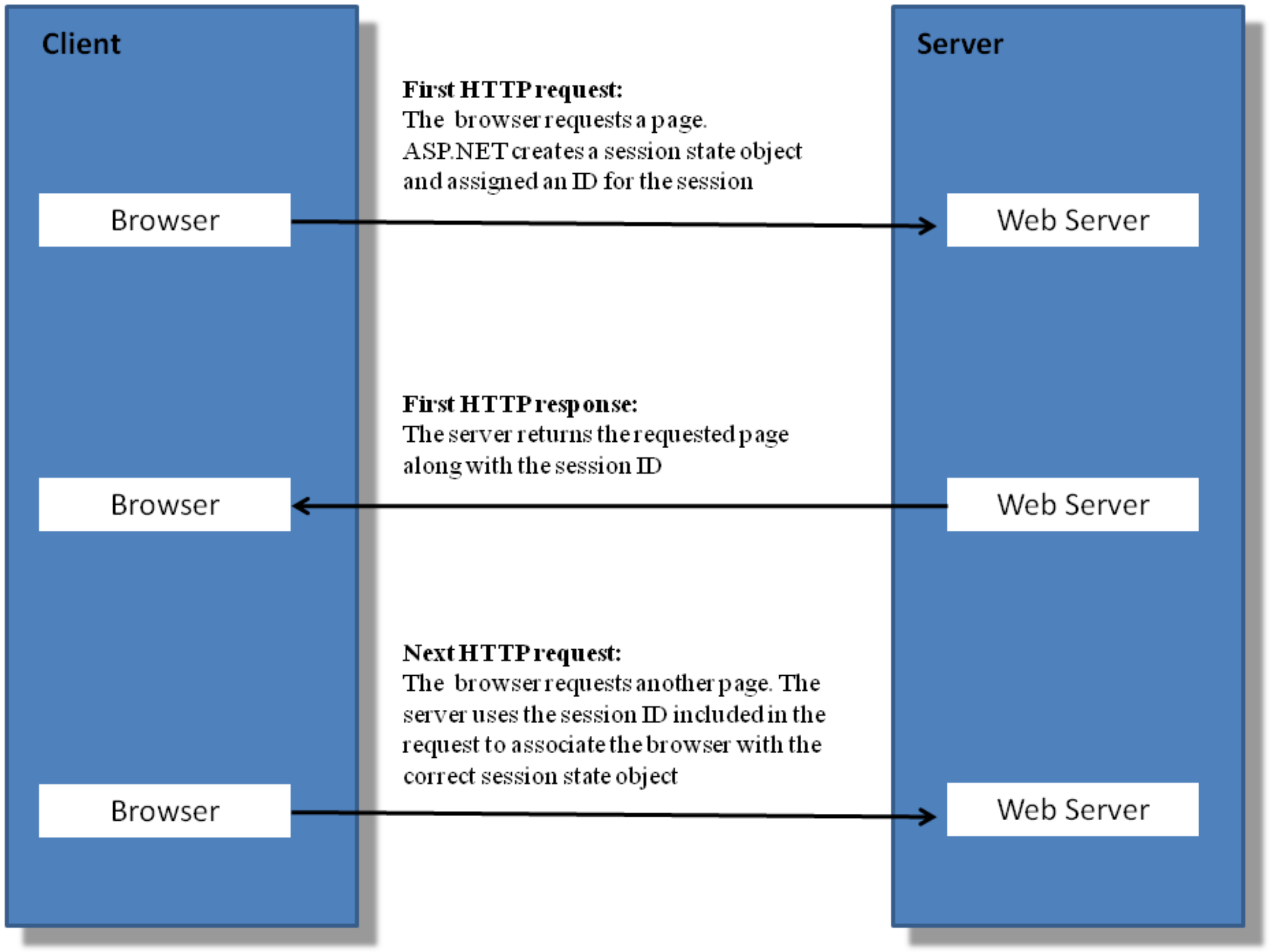
The *DataRowView* class

Indexer	Description
[index]	Returns the value of the columns at the specified index position as an object
[name]	Returns the value of the column with the specified name as an object

Using session state

HTTP is a stateless protocol. ASP.NET uses ***session state*** to keep track of each session and that you can use session state to maintain program values across executions of an application.

How ASP.NET maintains the state of a session


Client
Server
First HTTP request:
The browser requests a page.
ASP.NET creates a session state object and assigned an ID for the session
Browser
Web Server
First HTTP response:
The server returns the requested page along with the session ID
Browser
Web Server
Next HTTP request:
The browser requests another page. The server uses the session ID included in the request to associate the browser with the correct session state object
Browser
Web Server

Description

- ASP.NET uses session state to track the state of each user of an application. To do that, it creates a ***session state object***.
- The session state object includes a session ID that's sent back to the browser as a ***cookie***. Then, the browser automatically returns the session ID cookie to the server with each request so the server can associate the browser with an existing session state object.
- If you want your application to work on browsers that don't support cookies, you can configure ASP.NET to encode the session ID in the URL for each page of the application.
- You can use the session state object to store and retrieve items across executions of an application.

Typical uses for session state

- **To keep information about the user**, such as the user's name or whether the user has registered.
- **To save objects the user is working with**, such as a shopping cart or a customer record.
- **To keep track of pending operations**, such as what steps the user has completed while placing an order.

Using session state for storing a data

To do that, you use the member of this object, which is created from the *HttpSessionState* class. To access this object from a web form, you use the *Session* property of the page.

Common members of the *HttpSessionState* class

Property	**Description**
SessionID	The unique ID of the session
Count	The number of items in the session state collection
Indexer	**Description**
[name]	The value of the session state item with the specific name
Method	**Description**
Add(name, value)	Adds an item to the session state collection
Clear()	Removes all items from the session state collection
Remove(name)	Removes the item with the specified name from the session state collection

A statement that adds or updates a session state item

Session["Cart"] = cart;

Another way to add or update a session state item

Session.Add("Cart", cart);

A statement that retrieves the value of a session state item

SortedList cart = (SortedList) Session["Cart"]

A statement that removes an item from session state

Session.Remove ("Cart");

A statement that retrieves the value of a session state item from a class that doesn't inherit *System.Web.UI.Page*

```
SortedList cart = (SortedList) HttpContext.Current.Session["Cart"];
```

The code for the *Product* class

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

/// <summary>
/// Summary description for Product
/// </summary>
public class Product
{
   public string ProductID;
   public string Name;
   public string ShortDescription;
   public string LongDescription;
   public decimal UnitPrice;
   public string ImageFile;
}
```

The code for the *CartItem* class

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

/// <summary>
/// Summary description for CartItem
/// </summary>
public class CartItem
{
   public Product Product;
   public int Quantity;

   public string Display()
   {
      string displayString =
         Product.Name + " (" + Quantity.ToString()
                  + " at " + Product.UnitPrice.ToString("c") + " each)";

      return displayString;
   }
}
```

The aspx code for the *Order* page *(Order.aspx)*

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Order.aspx.cs" Inherits="Order" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
              "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Shopping Cart</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/banner.jpg" /><br /><br />
      <asp:Label ID="Label1" runat="server" Text="Please select a product:"></asp:Label>
      <asp:DropDownList ID="ddlProducts" runat="server" Width="150px"
         DataSourceID="AccessDataSource1" DataTextField="Name"
         DataValueField="ProductID" AutoPostBack="True">
      </asp:DropDownList>
      <asp:AccessDataSource ID="AccessDataSource1" runat="server"
         DataFile="~/App_Data/Halloween.mdb"
         SelectCommand="SELECT [ProductID], [Name], [ShortDescription],
            [LongDescription], [ImageFile], [UnitPrice]
            FROM [Products] ORDER BY [Name]">
      </asp:AccessDataSource>
      <br />
      <table>
         <tr>
            <td style="width: 250px; height: 22px">
               <asp:Label ID="lblName" runat="server"
                  Font-Bold="False" Font-Size="Larger">
               </asp:Label>
            </td>
            <td rowspan="4" style="width: 20px">
            </td>
            <td rowspan="4" valign="top">
               <asp:Image ID="imgProduct" runat="server" Height="200" />
            </td>
         </tr>
         <tr>
            <td style="width: 250px">
               <asp:Label ID="lblShortDescription" runat="server">
               </asp:Label>
            </td>
         </tr>
         <tr>
            <td style="width: 250px">
               <asp:Label ID="lblLongDescription" runat="server">
               </asp:Label>
            </td>
         </tr>
         <tr>
            <td style="width: 250px; height: 53px;">
               <asp:Label ID="lblUnitPrice" runat="server"
                  Font-Bold="True" Font-Size="Larger">
               </asp:Label>
               <asp:Label ID="Label2" runat="server" Font-Bold="True"
                  Font-Size="Larger" Text="each">
               </asp:Label>
            </td>
         </tr>
      </table>
      <br />
      <asp:Label ID="Label3" runat="server" Text="Quantity:"
         Width="80px" BorderWidth="0px"></asp:Label>
      <asp:TextBox ID="txtQuantity" runat="server" Width="80px">
      </asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
         ControlToValidate="txtQuantity" Display="Dynamic"
         ErrorMessage="Quantity is a required field.">
      </asp:RequiredFieldValidator>
      <asp:RangeValidator ID="RangeValidator1" runat="server"
         ControlToValidate="txtQuantity" Display="Dynamic"
         ErrorMessage="Quantity must range from 1 to 500."
         MaximumValue="500" MinimumValue="1" Type="Integer"></asp:RangeValidator><br /><br
/>
      <asp:Button ID="btnAdd" runat="server" Text="Add to Cart"
         OnClick="btnAdd_Click" />&nbsp;
      <asp:Button ID="btnCart" runat="server" CausesValidation="False"
        PostBackUrl="~/Cart.aspx" Text="Go to Cart" />
    </div>
    </form>
</body>
</html>
```

The code-behind file for the *Order* page *(Order.aspx.cs)*

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Collections;

public partial class Order : System.Web.UI.Page
{
  private Product selectedProduct;

  protected void Page_Load(object sender, EventArgs e)
  {
    if (!IsPostBack)
      ddlProducts.DataBind();

    selectedProduct = this.GetSelectedProduct();
    lblName.Text = selectedProduct.Name;
    lblShortDescription.Text = selectedProduct.ShortDescription;
    lblLongDescription.Text = selectedProduct.LongDescription;
    lblUnitPrice.Text = selectedProduct.UnitPrice.ToString("c");
    imgProduct.ImageUrl = "Images/Products/" + selectedProduct.ImageFile;
  }

  private Product GetSelectedProduct()
  {
    DataView productsTable = (DataView)
      AccessDataSource1.Select(DataSourceSelectArguments.Empty);

    productsTable.RowFilter = "ProductID = '" + ddlProducts.SelectedValue + "'";

    DataRowView row = (DataRowView) productsTable[0];
    Product p = new Product();

    p.ProductID = row["ProductID"].ToString();
    p.Name = row["Name"].ToString();
    p.ShortDescription = row["ShortDescription"].ToString();
    p.LongDescription = row["LongDescription"].ToString();
    p.UnitPrice = (decimal) row["UnitPrice"];
    p.ImageFile = row["ImageFile"].ToString();

    return p;
}

protected void btnAdd_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        CartItem item = new CartItem();

        item.Product = selectedProduct;
        item.Quantity = Convert.ToInt32(txtQuantity.Text);

        this.AddToCart(item);

        Response.Redirect("Cart.aspx");
    }
}

private void AddToCart(CartItem item)
{
    SortedList cart = this.GetCart();

    string productID = selectedProduct.ProductID;

    if (cart.ContainsKey(productID))
    {
        CartItem existingItem = (CartItem) cart[productID];

        existingItem.Quantity += item.Quantity;
    }
    else
        cart.Add(productID, item);
}

private SortedList GetCart()
{
    if (Session["Cart"] == null) { Session.Add("Cart", new SortedList()); }
```

```
        return (SortedList) Session["Cart"];
    }
}
```

The aspx file for the *Cart* page *(Cart.aspx)*

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Cart.aspx.cs" Inherits="Cart" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Shopping Cart</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/banner.jpg" /><br />
    <br />
    Your shopping cart:<br />
    <table border="0" cellpadding="0" cellspacing="0" style="width: 500px">
      <tr>
        <td style="width: 286px; height: 153px">
          <asp:ListBox ID="lstCart" runat="server" Height="135px" Width="267px"></asp:ListBox>
        </td>
        <td style="height: 153px">
          <asp:Button ID="btnRemove" runat="server" Text="Remove Item" Width="100px"
                    OnClick="btnRemove_Click" /><br />
          <br />
          <asp:Button ID="btnEmpty" runat="server" Text="Empty Cart" Width="100px"
                    OnClick="btnEmpty_Click" />
        </td>
      </tr>
    </table>
    <br />
    <asp:Button ID="btnContinue" runat="server" PostBackUrl="~/Order.aspx" Text="Continue
    Shopping" />&nbsp;
    <asp:Button ID="btnCheckOut" runat="server" Text="Check Out" OnClick="btnCheckOut_Click"
/><br />
    <br />
    <asp:Label ID="lblMessage" runat="server"></asp:Label>

  </div>
  </form>
```

```
</body>
</html>
```

The code-behind file for the *Cart* page *(Cart.aspx.cs)*

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Cart : System.Web.UI.Page
{
   private SortedList cart;

   protected void Page_Load(object sender, EventArgs e)
   {
      this.GetCart();

      if (!IsPostBack)
         this.DisplayCart();
   }

   private void GetCart()
   {
      if (Session["Cart"] == null) { Session.Add("Cart", new SortedList()); }

      cart = (SortedList) Session["Cart"];
   }

   private void DisplayCart()
   {
      lstCart.Items.Clear();

      CartItem item;

      foreach (DictionaryEntry entry in cart)
      {
         item = (CartItem) entry.Value;
         lstCart.Items.Add(item.Display());
      }
```

```
    }

    protected void btnRemove_Click(object sender, EventArgs e)
    {
        if (lstCart.SelectedIndex > -1 && cart.Count > 0)
        {
            cart.RemoveAt(lstCart.SelectedIndex);
            this.DisplayCart();
        }
    }

    protected void btnEmpty_Click(object sender, EventArgs e)
    {
        cart.Clear();
        lstCart.Items.Clear();
        lblMessage.Text = "";
    }

    protected void btnCheckOut_Click(object sender, EventArgs e)
    {
        lblMessage.Text = "Sorry, that function hasn't been "
                + "implemented yet.";
    }
}
```