CHAPTER

Coding

1. Add CartItem as a Model Class

```
using System.ComponentModel.DataAnnotations;

namespace WingtipToys.Models
{
    public class CartItem
    {
        [Key]
        public string ItemId { get; set; }

        public string CartId { get; set; }

        public int Quantity { get; set; }

        public System.DateTime DateCreated { get; set; }

        public int ProductId { get; set; }

        public virtual Product Product { get; set; }

    }
}
```

2. Update the Product Context

```
using System.Data.Entity;
```

```
namespace WingtipToys.Models
{
    public class ProductContext : DbContext
    {
        public ProductContext()
            : base("WingtipToys")
        {
        }

        public DbSet<Category> Categories { get; set; }
        public DbSet<Product> Products { get; set; }
        public DbSet<CartItem> ShoppingCartItems { get; set; }
    }
}
```

3. Managing the Shopping Cart Business Logic

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using WingtipToys.Models;

namespace WingtipToys.Logic
{
  public class ShoppingCartActions : IDisposable
  {
    public string ShoppingCartId { get; set; }

    private ProductContext _db = new ProductContext();

    public const string CartSessionKey = "CartId";

    public void AddToCart(int id)
    {
      // Retrieve the product from the database.
      ShoppingCartId = GetCartId();

      var cartItem = _db.ShoppingCartItems.SingleOrDefault(
          c => c.CartId == ShoppingCartId
          && c.ProductId == id);
      if (cartItem == null)
      {
        // Create a new cart item if no cart item exists.
        cartItem = new CartItem
        {
          ItemId = Guid.NewGuid().ToString(),
          ProductId = id,
          CartId = ShoppingCartId,
          Product = _db.Products.SingleOrDefault(
           p => p.ProductID == id),
          Quantity = 1,
          DateCreated = DateTime.Now
        };

        _db.ShoppingCartItems.Add(cartItem);
      }
      else
      {
        // If the item does exist in the cart,
        // then add one to the quantity.
        cartItem.Quantity++;
      }
      _db.SaveChanges();
    }

    public void Dispose()
    {
      if (_db != null)
      {
        _db.Dispose();
        _db = null;
      }
    }

    public string GetCartId()
    {
      if (HttpContext.Current.Session[CartSessionKey] == null)
      {
        if
(!string.IsNullOrWhiteSpace(HttpContext.Current.User.Identity.Name))
        {
          HttpContext.Current.Session[CartSessionKey] =
HttpContext.Current.User.Identity.Name;
        }
        else
        {
          // Generate a new random GUID using System.Guid class.
          Guid tempCartId = Guid.NewGuid();
          HttpContext.Current.Session[CartSessionKey] =
tempCartId.ToString();
        }
      }
      return HttpContext.Current.Session[CartSessionKey].ToString();
    }

    public List<CartItem> GetCartItems()
    {
      ShoppingCartId = GetCartId();

      return _db.ShoppingCartItems.Where(
          c => c.CartId == ShoppingCartId).ToList();
    }
  }
}
```

4. Creating the Add-To-Cart Functionality

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Diagnostics;
using WingtipToys.Logic;

namespace WingtipToys
{
  public partial class AddToCart : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      string rawId = Request.QueryString["ProductID"];
      int productId;
      if (!String.IsNullOrEmpty(rawId) && int.TryParse(rawId, out
productId))
      {
        using (ShoppingCartActions usersShoppingCart = new
ShoppingCartActions())
        {
          usersShoppingCart.AddToCart(Convert.ToInt16(rawId));
        }

      }
      else
      {
        Debug.Fail("ERROR : We should never get to AddToCart.aspx without
a ProductId.");
        throw new Exception("ERROR : It is illegal to load AddToCart.aspx
without setting a ProductId.");
      }
      Response.Redirect("ShoppingCart.aspx");
    }
  }
}
```

5. Creating the Shopping Cart UI

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="ShoppingCart.aspx.cs"
Inherits="WingtipToys.ShoppingCart" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <div id="ShoppingCartTitle" runat="server"
class="ContentHead"><h1>Shopping Cart</h1></div>
    <asp:GridView ID="CartList" runat="server"
AutoGenerateColumns="False" ShowFooter="True" GridLines="Vertical"
CellPadding="4"
        ItemType="WingtipToys.Models.CartItem"
SelectMethod="GetShoppingCartItems"
        CssClass="table table-striped table-bordered" >
        <Columns>
        <asp:BoundField DataField="ProductID" HeaderText="ID"
SortExpression="ProductID" />
        <asp:BoundField DataField="Product.ProductName"
HeaderText="Name" />
        <asp:BoundField DataField="Product.UnitPrice" HeaderText="Price
(each)" DataFormatString="{0:c}"/>
        <asp:TemplateField   HeaderText="Quantity">
                <ItemTemplate>
                    <asp:TextBox ID="PurchaseQuantity" Width="40"
runat="server" Text="<%#: Item.Quantity %>"></asp:TextBox>
                </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Item Total">
                <ItemTemplate>
                    <%#: String.Format("{0:c}",
((Convert.ToDouble(Item.Quantity)) *
Convert.ToDouble(Item.Product.UnitPrice)))%>
                </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Remove Item">
                <ItemTemplate>
                    <asp:CheckBox id="Remove"
runat="server"></asp:CheckBox>
                </ItemTemplate>
        </asp:TemplateField>
        </Columns>
    </asp:GridView>
    <div>
        <p></p>
        <strong>
            <asp:Label ID="LabelTotalText" runat="server" Text="Order Total:
"></asp:Label>
            <asp:Label ID="lblTotal" runat="server"
EnableViewState="false"></asp:Label>
        </strong>
    </div>
    <br />
</asp:Content>
```

6. Retrieving the Shopping Cart Items

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WingtipToys.Models;
using WingtipToys.Logic;

namespace WingtipToys
{
  public partial class ShoppingCart : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    public List<CartItem> GetShoppingCartItems()
    {
      ShoppingCartActions actions = new ShoppingCartActions();
      return actions.GetCartItems();
    }
  }
}
```

7. Adding Products to the Shopping Cart

```
<%@ Page Title="Products" Language="C#"
MasterPageFile="~/Site.Master" AutoEventWireup="true"
         CodeBehind="ProductList.aspx.cs"
Inherits="WingtipToys.ProductList" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <section>
        <div>
            <hgroup>
    <h2><%: Page.Title %></h2>
  </hgroup>

  <asp:ListView ID="productList" runat="server"
    DataKeyNames="ProductID" GroupItemCount="4"
    ItemType="WingtipToys.Models.Product"
SelectMethod="GetProducts">
    <EmptyDataTemplate>
      <table runat="server">
        <tr>
          <td>No data was returned.</td>
        </tr>
      </table>
    </EmptyDataTemplate>
    <EmptyItemTemplate>
      <td runat="server" />
    </EmptyItemTemplate>
    <GroupTemplate>
      <tr id="itemPlaceholderContainer" runat="server">
        <td id="itemPlaceholder" runat="server"></td>
      </tr>
    </GroupTemplate>
    <ItemTemplate>
      <td runat="server">
        <table>
          <tr>
            <td>
              <a
href="ProductDetails.aspx?productID=<%#:Item.ProductID%>">
                <img
src="/Catalog/Images/Thumbs/<%#:Item.ImagePath%>"
                  width="100" height="75" style="border: solid"
/></a>
            </td>
          </tr>
          <tr>
            <td>
              <a
href="ProductDetails.aspx?productID=<%#:Item.ProductID%>">
                <span>
                  <%#:Item.ProductName%>
                </span>
              </a>
```

```
                                  <br />
                                  <span>
                                      <b>Price: </b><%#:String.Format("{0:c}", 
Item.UnitPrice)%>
                                  </span>
                                  <br />
                                  <a 
href="/AddToCart.aspx?productID=<%#:Item.ProductID %>">
                                      <span class="ProductListItem">
                                          <b>Add To Cart<b>
                                      </span>
                                  </a>
                              </td>
                          </tr>
                          <tr>
                              <td>&nbsp;</td>
                          </tr>
                      </table>
                      </p>
                  </td>
              </ItemTemplate>
              <LayoutTemplate>
                  <table runat="server" style="width:100%;">
                      <tbody>
                          <tr runat="server">
                              <td runat="server">
                                  <table id="groupPlaceholderContainer" 
runat="server" style="width:100%">
                                      <tr id="groupPlaceholder" runat="server"></tr>
                                  </table>
                              </td>
                          </tr>
                          <tr runat="server">
                              <td runat="server"></td>
                          </tr>
                          <tr></tr>
                      </tbody>
                  </table>
              </LayoutTemplate>
          </asp:ListView>
      </div>
  </section>
</asp:Content>
```

8. Calculating and Displaying the Order Total

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using WingtipToys.Models;

namespace WingtipToys.Logic
{
  public class ShoppingCartActions : IDisposable
  {
    public string ShoppingCartId { get; set; }

    private ProductContext _db = new ProductContext();

    public const string CartSessionKey = "CartId";

    public void AddToCart(int id)
    {
      // Retrieve the product from the database.
      ShoppingCartId = GetCartId();

      var cartItem = _db.ShoppingCartItems.SingleOrDefault(
          c => c.CartId == ShoppingCartId
          && c.ProductId == id);
      if (cartItem == null)
      {
        // Create a new cart item if no cart item exists.
        cartItem = new CartItem
        {
          ItemId = Guid.NewGuid().ToString(),
          ProductId = id,
          CartId = ShoppingCartId,
          Product = _db.Products.SingleOrDefault(
           p => p.ProductID == id),
          Quantity = 1,
          DateCreated = DateTime.Now
        };

        _db.ShoppingCartItems.Add(cartItem);
      }
      else
      {
        // If the item does exist in the cart,
        // then add one to the quantity.
        cartItem.Quantity++;
      }
      _db.SaveChanges();
    }

    public void Dispose()
    {
      if (_db != null)
      {
        _db.Dispose();
        _db = null;
      }
    }

    public string GetCartId()
    {
      if (HttpContext.Current.Session[CartSessionKey] == null)
      {
        if
(!string.IsNullOrWhiteSpace(HttpContext.Current.User.Identity.Name))
        {
          HttpContext.Current.Session[CartSessionKey] =
HttpContext.Current.User.Identity.Name;
        }
        else
        {
          // Generate a new random GUID using System.Guid class.
          Guid tempCartId = Guid.NewGuid();
          HttpContext.Current.Session[CartSessionKey] =
tempCartId.ToString();
        }
      }
      return HttpContext.Current.Session[CartSessionKey].ToString();
    }
    public List<CartItem> GetCartItems()
    {
      ShoppingCartId = GetCartId();

      return _db.ShoppingCartItems.Where(
          c => c.CartId == ShoppingCartId).ToList();
    }

    public decimal GetTotal()
    {
      ShoppingCartId = GetCartId();
      // Multiply product price by quantity of that product to get
      // the current price for each of those products in the cart.
      // Sum all product price totals to get the cart total.
      decimal? total = decimal.Zero;
      total = (decimal?)(from cartItems in _db.ShoppingCartItems
                         where cartItems.CartId == ShoppingCartId
                         select (int?)cartItems.Quantity *
                         cartItems.Product.UnitPrice).Sum();
      return total ?? decimal.Zero;
    }
  }
}
```

9. Modify the Shopping Cart Display

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WingtipToys.Models;
using WingtipToys.Logic;

namespace WingtipToys
{
  public partial class ShoppingCart : System.Web.UI.Page
  {
```

```
  protected void Page_Load(object sender, EventArgs e)
  {
    using (ShoppingCartActions usersShoppingCart = new
ShoppingCartActions())
    {
      decimal cartTotal = 0;
      cartTotal = usersShoppingCart.GetTotal();
      if (cartTotal > 0)
      {
        // Display Total.
        lblTotal.Text = String.Format("{0:c}", cartTotal);
      }
      else
      {
        LabelTotalText.Text = "";
        lblTotal.Text = "";
        ShoppingCartTitle.InnerText = "Shopping Cart is Empty";
      }
    }
  }

  public List<CartItem> GetShoppingCartItems()
  {
    ShoppingCartActions actions = new ShoppingCartActions();
    return actions.GetCartItems();
  }
 }
}
```

10. Adding Update and Checkout Buttons to the Shopping Cart

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="ShoppingCart.aspx.cs"
Inherits="WingtipToys.ShoppingCart" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
```

```
    <div id="ShoppingCartTitle" runat="server"
class="ContentHead"><h1>Shopping Cart</h1></div>
    <asp:GridView ID="CartList" runat="server"
AutoGenerateColumns="False" ShowFooter="True" GridLines="Vertical"
CellPadding="4"
        ItemType="WingtipToys.Models.CartItem"
SelectMethod="GetShoppingCartItems"
        CssClass="table table-striped table-bordered" >
        <Columns>
        <asp:BoundField DataField="ProductID" HeaderText="ID"
SortExpression="ProductID" />
        <asp:BoundField DataField="Product.ProductName"
HeaderText="Name" />
        <asp:BoundField DataField="Product.UnitPrice" HeaderText="Price
(each)" DataFormatString="{0:c}"/>
        <asp:TemplateField   HeaderText="Quantity">
                <ItemTemplate>
                    <asp:TextBox ID="PurchaseQuantity" Width="40"
runat="server" Text="<%#: Item.Quantity %>"></asp:TextBox>
                </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Item Total">
                <ItemTemplate>
                    <%#: String.Format("{0:c}",
((Convert.ToDouble(Item.Quantity)) *
Convert.ToDouble(Item.Product.UnitPrice)))%>
                </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Remove Item">
                <ItemTemplate>
                    <asp:CheckBox id="Remove"
runat="server"></asp:CheckBox>
                </ItemTemplate>
        </asp:TemplateField>
        </Columns>
    </asp:GridView>
    <div>
        <p></p>
        <strong>
            <asp:Label ID="LabelTotalText" runat="server" Text="Order Total:
"></asp:Label>
            <asp:Label ID="lblTotal" runat="server"
EnableViewState="false"></asp:Label>
```

```
        </strong>
    </div>
  <br />
    <table>
    <tr>
      <td>
        <asp:Button ID="UpdateBtn" runat="server" Text="Update"
OnClick="UpdateBtn_Click" />
      </td>
      <td>
        <!--Checkout Placeholder -->
      </td>
    </tr>
    </table>
</asp:Content>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WingtipToys.Models;
using WingtipToys.Logic;
using System.Collections.Specialized;
using System.Collections;
using System.Web.ModelBinding;

namespace WingtipToys
{
  public partial class ShoppingCart : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      using (ShoppingCartActions usersShoppingCart = new
ShoppingCartActions())
      {
        decimal cartTotal = 0;
        cartTotal = usersShoppingCart.GetTotal();
        if (cartTotal > 0)
        {
          // Display Total.
          lblTotal.Text = String.Format("{0:c}", cartTotal);
        }
        else
        {
          LabelTotalText.Text = "";
          lblTotal.Text = "";
          ShoppingCartTitle.InnerText = "Shopping Cart is Empty";
          UpdateBtn.Visible = false;
        }
      }
    }

    public List<CartItem> GetShoppingCartItems()
    {
      ShoppingCartActions actions = new ShoppingCartActions();
      return actions.GetCartItems();
    }

    public List<CartItem> UpdateCartItems()
    {
      using (ShoppingCartActions usersShoppingCart = new
ShoppingCartActions())
      {
        String cartId = usersShoppingCart.GetCartId();

        ShoppingCartActions.ShoppingCartUpdates[] cartUpdates = new
ShoppingCartActions.ShoppingCartUpdates[CartList.Rows.Count];
        for (int i = 0; i < CartList.Rows.Count; i++)
        {
          IOrderedDictionary rowValues = new OrderedDictionary();
          rowValues = GetValues(CartList.Rows[i]);
          cartUpdates[i].ProductId =
Convert.ToInt32(rowValues["ProductID"]);

          CheckBox cbRemove = new CheckBox();
          cbRemove = (CheckBox)CartList.Rows[i].FindControl("Remove");
          cartUpdates[i].RemoveItem = cbRemove.Checked;

          TextBox quantityTextBox = new TextBox();
          quantityTextBox =
(TextBox)CartList.Rows[i].FindControl("PurchaseQuantity");
          cartUpdates[i].PurchaseQuantity =
Convert.ToInt16(quantityTextBox.Text.ToString());
        }
        usersShoppingCart.UpdateShoppingCartDatabase(cartId, cartUpdates);
        CartList.DataBind();
        lblTotal.Text = String.Format("{0:c}", usersShoppingCart.GetTotal());
        return usersShoppingCart.GetCartItems();
      }
    }

    public static IOrderedDictionary GetValues(GridViewRow row)
    {
      IOrderedDictionary values = new OrderedDictionary();
      foreach (DataControlFieldCell cell in row.Cells)
      {
        if (cell.Visible)
        {
          // Extract values from the cell.
          cell.ContainingField.ExtractValuesFromCell(values, cell,
row.RowState, true);
        }
      }
      return values;
    }

    protected void UpdateBtn_Click(object sender, EventArgs e)
    {
      UpdateCartItems();
    }
  }
}
```

11. Updating and Removing Shopping Cart Items

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using WingtipToys.Models;

namespace WingtipToys.Logic
```

```
{
  public class ShoppingCartActions : IDisposable
  {
    public string ShoppingCartId { get; set; }

    private ProductContext _db = new ProductContext();

    public const string CartSessionKey = "CartId";

    public void AddToCart(int id)
    {
      // Retrieve the product from the database.
      ShoppingCartId = GetCartId();

      var cartItem = _db.ShoppingCartItems.SingleOrDefault(
          c => c.CartId == ShoppingCartId
          && c.ProductId == id);
      if (cartItem == null)
      {
        // Create a new cart item if no cart item exists.
        cartItem = new CartItem
        {
          ItemId = Guid.NewGuid().ToString(),
          ProductId = id,
          CartId = ShoppingCartId,
          Product = _db.Products.SingleOrDefault(
           p => p.ProductID == id),
          Quantity = 1,
          DateCreated = DateTime.Now
        };

        _db.ShoppingCartItems.Add(cartItem);
      }
      else
      {
        // If the item does exist in the cart,
        // then add one to the quantity.
        cartItem.Quantity++;
      }
      _db.SaveChanges();
    }

    public void Dispose()
    {
      if (_db != null)
      {
        _db.Dispose();
        _db = null;
      }
    }

    public string GetCartId()
    {
      if (HttpContext.Current.Session[CartSessionKey] == null)
      {
        if
(!string.IsNullOrWhiteSpace(HttpContext.Current.User.Identity.Name))
        {
          HttpContext.Current.Session[CartSessionKey] =
HttpContext.Current.User.Identity.Name;
        }
        else
        {
          // Generate a new random GUID using System.Guid class.
          Guid tempCartId = Guid.NewGuid();
          HttpContext.Current.Session[CartSessionKey] =
tempCartId.ToString();
        }
      }
      return HttpContext.Current.Session[CartSessionKey].ToString();
    }

    public List<CartItem> GetCartItems()
    {
      ShoppingCartId = GetCartId();

      return _db.ShoppingCartItems.Where(
          c => c.CartId == ShoppingCartId).ToList();
    }

    public decimal GetTotal()
    {
      ShoppingCartId = GetCartId();
      // Multiply product price by quantity of that product to get
      // the current price for each of those products in the cart.
      // Sum all product price totals to get the cart total.
      decimal? total = decimal.Zero;
      total = (decimal?)(from cartItems in _db.ShoppingCartItems
                    where cartItems.CartId == ShoppingCartId
                    select (int?)cartItems.Quantity *
                    cartItems.Product.UnitPrice).Sum();
      return total ?? decimal.Zero;
    }

    public ShoppingCartActions GetCart(HttpContext context)
    {
      using (var cart = new ShoppingCartActions())
      {
        cart.ShoppingCartId = cart.GetCartId();
        return cart;
      }
    }

    public void UpdateShoppingCartDatabase(String cartId,
ShoppingCartUpdates[] CartItemUpdates)
    {
      using (var db = new WingtipToys.Models.ProductContext())
      {
        try
        {
          int CartItemCount = CartItemUpdates.Count();
          List<CartItem> myCart = GetCartItems();
          foreach (var cartItem in myCart)
          {
            // Iterate through all rows within shopping cart list
            for (int i = 0; i < CartItemCount; i++)
            {
              if (cartItem.Product.ProductID == CartItemUpdates[i].ProductId)
              {
                if (CartItemUpdates[i].PurchaseQuantity < 1 ||
CartItemUpdates[i].RemoveItem == true)
                {
                  RemoveItem(cartId, cartItem.ProductId);
                }
                else
                {
                  UpdateItem(cartId, cartItem.ProductId,
CartItemUpdates[i].PurchaseQuantity);
                }
              }
            }
          }
        }
        catch (Exception exp)
        {
          throw new Exception("ERROR: Unable to Update Cart Database - " +
exp.Message.ToString(), exp);
        }
      }
    }

    public void RemoveItem(string removeCartID, int removeProductID)
    {
      using (var _db = new WingtipToys.Models.ProductContext())
      {
        try
        {
          var myItem = (from c in _db.ShoppingCartItems where c.CartId ==
removeCartID && c.Product.ProductID == removeProductID select
c).FirstOrDefault();
          if (myItem != null)
          {
            // Remove Item.
            _db.ShoppingCartItems.Remove(myItem);
            _db.SaveChanges();
          }
        }
        catch (Exception exp)
        {
          throw new Exception("ERROR: Unable to Remove Cart Item - " +
exp.Message.ToString(), exp);
        }
      }
    }

    public void UpdateItem(string updateCartID, int updateProductID, int
quantity)
    {
      using (var _db = new WingtipToys.Models.ProductContext())
      {
        try
        {
          var myItem = (from c in _db.ShoppingCartItems where c.CartId ==
updateCartID && c.Product.ProductID == updateProductID select
c).FirstOrDefault();
          if (myItem != null)
          {
            myItem.Quantity = quantity;
            _db.SaveChanges();
          }
        }
        catch (Exception exp)
        {
          throw new Exception("ERROR: Unable to Update Cart Item - " +
exp.Message.ToString(), exp);
        }
      }
    }

    public void EmptyCart()
    {
      ShoppingCartId = GetCartId();
      var cartItems = _db.ShoppingCartItems.Where(
          c => c.CartId == ShoppingCartId);
      foreach (var cartItem in cartItems)
      {
        _db.ShoppingCartItems.Remove(cartItem);
      }
      // Save changes.
      _db.SaveChanges();
    }

    public int GetCount()
    {
      ShoppingCartId = GetCartId();

      // Get the count of each item in the cart and sum them up
      int? count = (from cartItems in _db.ShoppingCartItems
                    where cartItems.CartId == ShoppingCartId
                    select (int?)cartItems.Quantity).Sum();
      // Return 0 if all entries are null
      return count ?? 0;
    }

    public struct ShoppingCartUpdates
    {
      public int ProductId;
      public int PurchaseQuantity;
      public bool RemoveItem;
    }
  }
}
```

12. Adding a Shopping Cart Counter

```
<ul class="nav navbar-nav">
      <li><a runat="server" href="~/">Home</a></li>
      <li><a runat="server" href="~/About">About</a></li>
      <li><a runat="server" href="~/Contact">Contact</a></li>
      <li><a runat="server" href="~/ProductList">Products</a></li>
      <li><a runat="server" href="~/ShoppingCart"
ID="cartCount">&nbsp;</a></li>
  </ul>

using System;
using System.Collections.Generic;
using System.Security.Claims;
using System.Security.Principal;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Linq;
using WingtipToys.Models;
using WingtipToys.Logic;

namespace WingtipToys
{
    public partial class SiteMaster : MasterPage
    {
        private const string AntiXsrfTokenKey = "__AntiXsrfToken";
        private const string AntiXsrfUserNameKey = "__AntiXsrfUserName";
        private string _antiXsrfTokenValue;
        protected void Page_Init(object sender, EventArgs e)
        {
            // The code below helps to protect against XSRF attacks
            var requestCookie = Request.Cookies[AntiXsrfTokenKey];
            Guid requestCookieGuidValue;
            if (requestCookie != null && Guid.TryParse(requestCookie.Value,
out requestCookieGuidValue))
            {
                // Use the Anti-XSRF token from the cookie
                _antiXsrfTokenValue = requestCookie.Value;
                Page.ViewStateUserKey = _antiXsrfTokenValue;
            }
            else
            {
                // Generate a new Anti-XSRF token and save to the cookie
                _antiXsrfTokenValue = Guid.NewGuid().ToString("N");
                Page.ViewStateUserKey = _antiXsrfTokenValue;

                var responseCookie = new HttpCookie(AntiXsrfTokenKey)
                {
                    HttpOnly = true,
                    Value = _antiXsrfTokenValue
                };
                if (FormsAuthentication.RequireSSL &&
Request.IsSecureConnection)
                {
                    responseCookie.Secure = true;
                }
                Response.Cookies.Set(responseCookie);
            }

            Page.PreLoad += master_Page_PreLoad;
        }

        protected void master_Page_PreLoad(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                // Set Anti-XSRF token
                ViewState[AntiXsrfTokenKey] = Page.ViewStateUserKey;
                ViewState[AntiXsrfUserNameKey] = Context.User.Identity.Name
?? String.Empty;
            }
            else
            {
                // Validate the Anti-XSRF token
                if ((string)ViewState[AntiXsrfTokenKey] != _antiXsrfTokenValue
                    || (string)ViewState[AntiXsrfUserNameKey] !=
(Context.User.Identity.Name ?? String.Empty))
                {
                    throw new InvalidOperationException("Validation of Anti-
XSRF token failed.");
                }
            }
        }

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Page_PreRender(object sender, EventArgs e)
        {
          using (ShoppingCartActions usersShoppingCart = new
ShoppingCartActions())
          {
            string cartStr = string.Format("Cart ({0})",
usersShoppingCart.GetCount());
            cartCount.InnerText = cartStr;
          }
        }

        public IQueryable<Category> GetCategories()
        {
          var _db = new WingtipToys.Models.ProductContext();
          IQueryable<Category> query = _db.Categories;
          return query;
        }

        protected void Unnamed_LoggingOut(object sender,
LoginCancelEventArgs e)
        {
            Context.GetOwinContext().Authentication.SignOut();
        }
    }
}
```