Using Web API with ASP.NET Web Forms

By Mike Wasson | April 3, 2012 Like 15 Tweet 4

Although ASP.NET Web API is packaged with ASP.NET MVC, it is easy to add Web API to a traditional ASP.NET Web Forms application. This tutorial walks you through the steps.

Overview

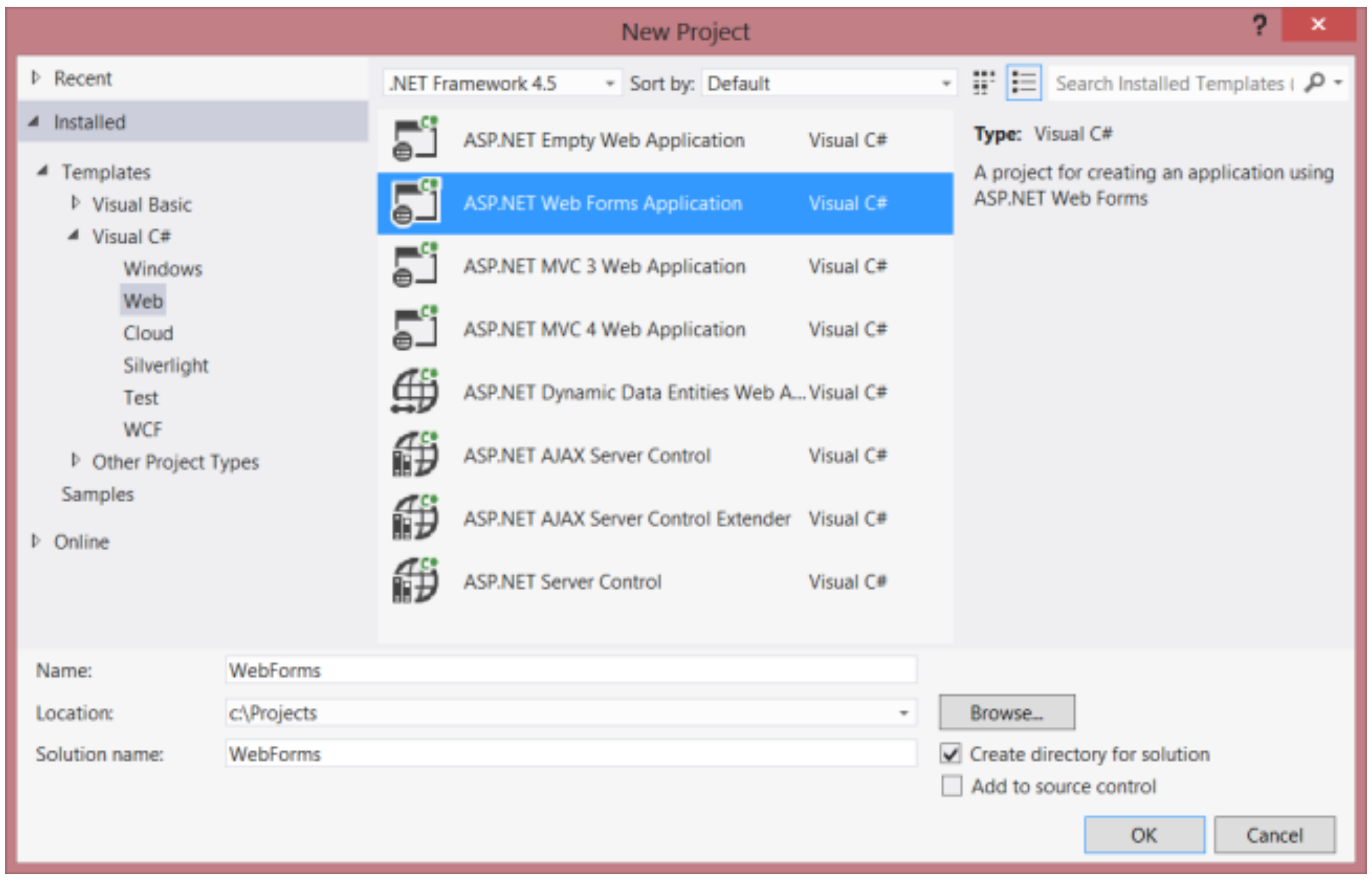
To use Web API in a Web Forms application, there are two main steps:

- Add a Web API controller that derives from the **ApiController** class.
- Add a route table to the **Application_Start** method.

Create a Web Forms Project

Start Visual Studio and select **New Project** from the **Start** page. Or, from the **File** menu, select **New** and then **Project**.

In the **Templates** pane, select **Installed Templates** and expand the **Visual C#** node. Under **Visual C#**, select **Web**. In the list of project templates, select **ASP.NET Web Forms Application**. Enter a name for the project and click **OK**.


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Cloud
Silverlight
Test
WCF
Other Project Types
Samples
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates
ASP.NET Empty Web Application Visual C#
ASP.NET Web Forms Application Visual C#
ASP.NET MVC 3 Web Application Visual C#
ASP.NET MVC 4 Web Application Visual C#
ASP.NET Dynamic Data Entities Web A... Visual C#
ASP.NET AJAX Server Control Visual C#
ASP.NET AJAX Server Control Extender Visual C#
ASP.NET Server Control Visual C#
Browse...
Add to source control
OK
Cancel

Type: Visual C#
A project for creating an application using ASP.NET Web Forms

Name: WebForms
Location: c:\Projects
Solution name: WebForms
☑ Create directory for solution

Create the Model and Controller

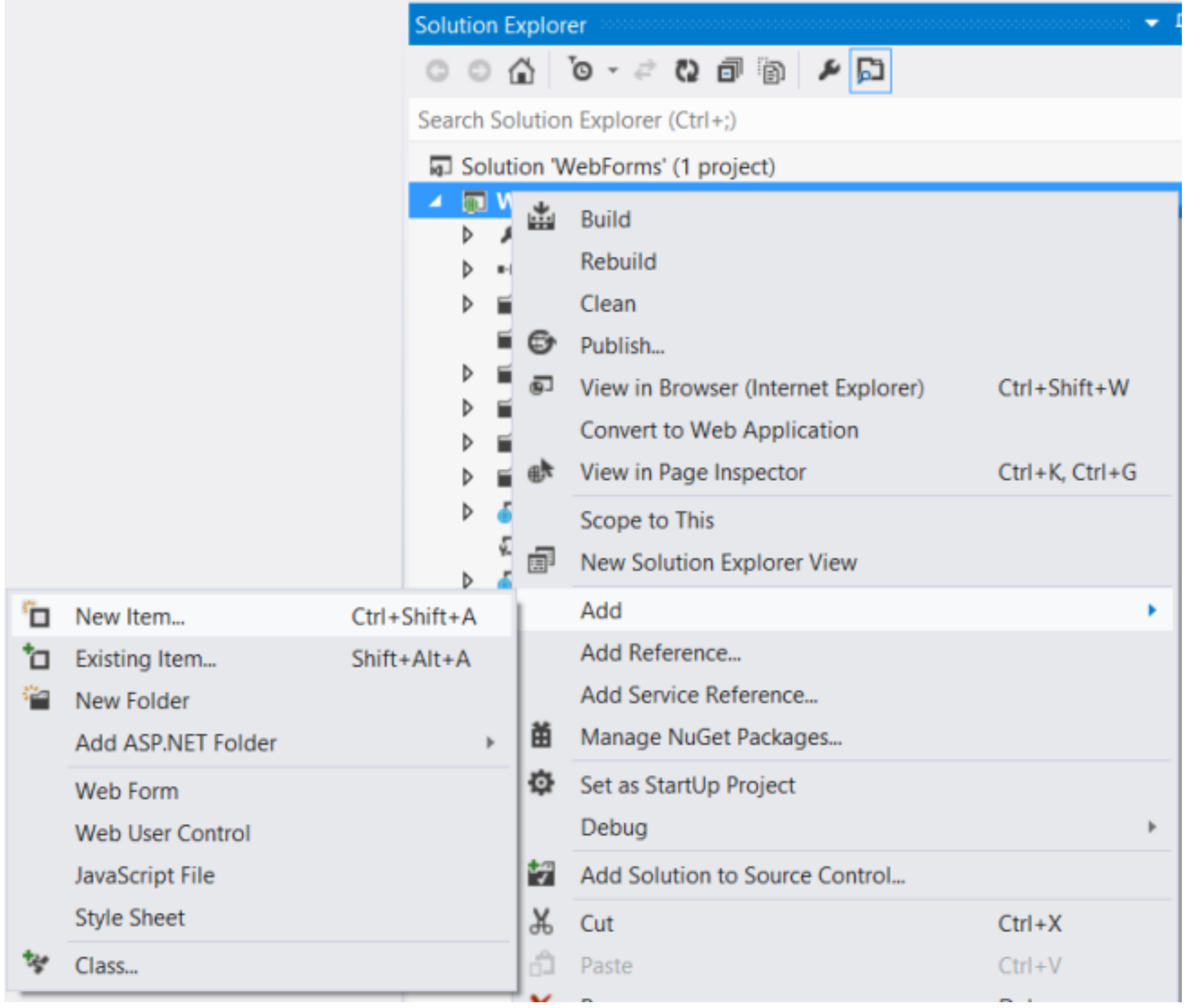
This tutorial uses the same model and controller classes as the Getting Started tutorial.

First, add a model class. In **Solution Explorer**, right-click the project and select **Add Class**. Name the class Product, and add the following implementation:

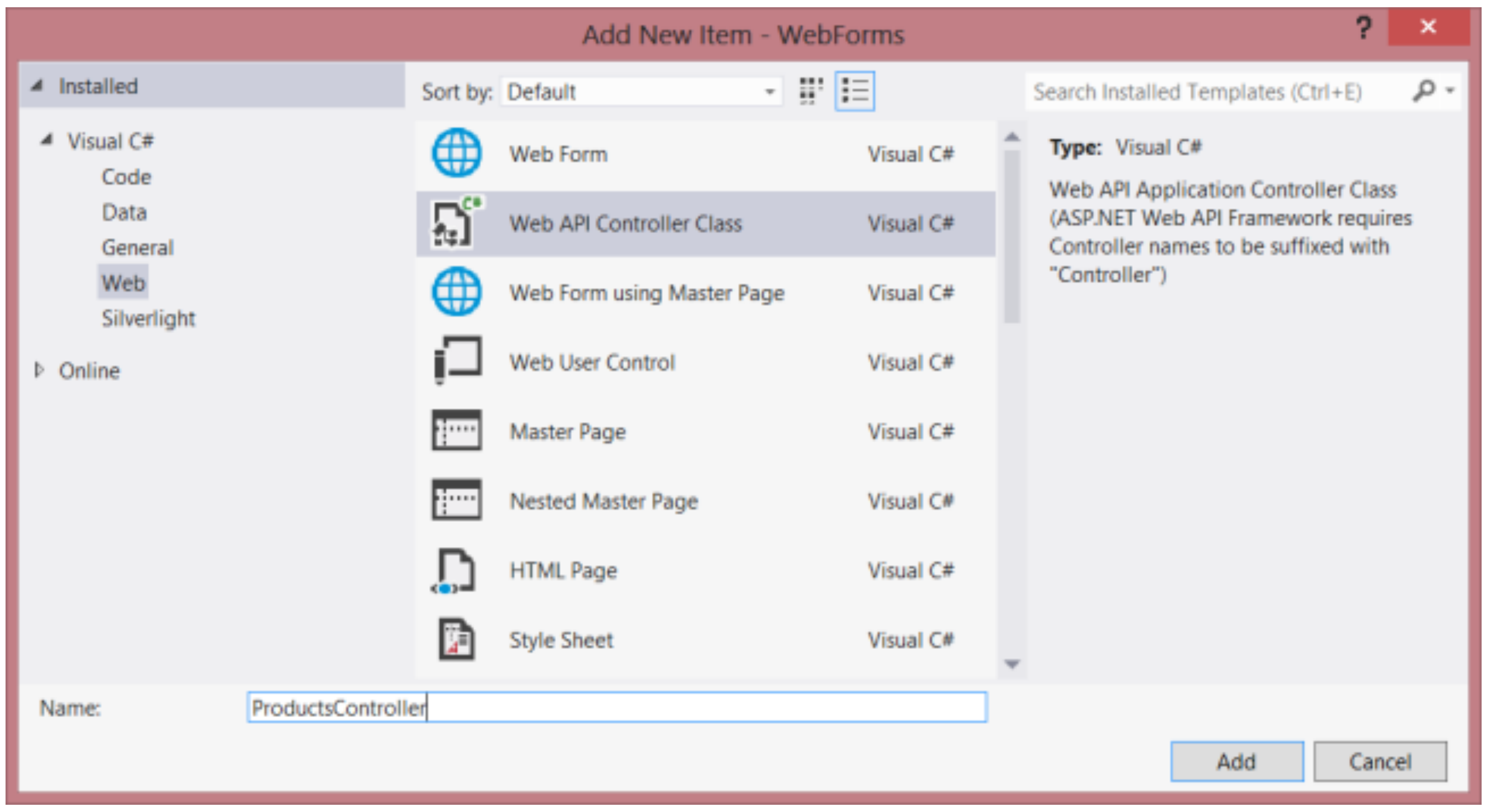
```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public string Category { get; set; }
}
```

Next, add a Web API controller to the project., A *controller* is the object that handles HTTP requests for Web API.

In **Solution Explorer**, right-click the project. Select **Add New Item**.



Under **Installed Templates**, expand **Visual C#** and select **Web**. Then, from the list of templates, select **Web API Controller Class**. Name the controller "ProductsController" and click **Add**.



Type: Visual C#

Web API Application Controller Class (ASP.NET Web API Framework requires Controller names to be suffixed with "Controller")

Name: ProductsController

The **Add New Item** wizard will create a file named ProductsController.cs. Delete the methods that the wizard included and add the following methods:

```
namespace WebForms
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Net.Http;
    using System.Web.Http;

    public class ProductsController : ApiController
    {

        Product[] products = new Product[]
        {
            new Product { Id = 1, Name = "Tomato Soup", Category = "Groceries", Price = 1 },
            new Product { Id = 2, Name = "Yo-yo", Category = "Toys", Price = 3.75M },
            new Product { Id = 3, Name = "Hammer", Category = "Hardware", Price = 16.99M }
        };

        public IEnumerable<Product> GetAllProducts()
        {
            return products;
        }

        public Product GetProductById(int id)
        {
            var product = products.FirstOrDefault((p) => p.Id == id);
            if (product == null)
            {
                throw new HttpResponseException(HttpStatusCode.NotFound);
            }
            return product;
        }

        public IEnumerable<Product> GetProductsByCategory(string category)
        {
            return products.Where(
                (p) => string.Equals(p.Category, category,
                    StringComparison.OrdinalIgnoreCase));
        }
    }
}
```

For more information about the code in this controller, see the Getting Started tutorial.

Add Routing Information

Next, we'll add a URI route so that URIs of the form "/api/products/" are routed to the controller.

In **Solution Explorer**, double-click Global.asax to open the code-behind file Global.asax.cs. Add the following **using** statement.

```
using System.Web.Http;
```

Then add the following code to the **Application_Start** method:

```
RouteTable.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = System.Web.Http.RouteParameter.Optional }
    );
```

For more information about routing tables, see Routing in ASP.NET Web API.

Add Client-Side AJAX

That's all you need to create a web API that clients can access. Now let's add an HTML page that uses jQuery to call the API.

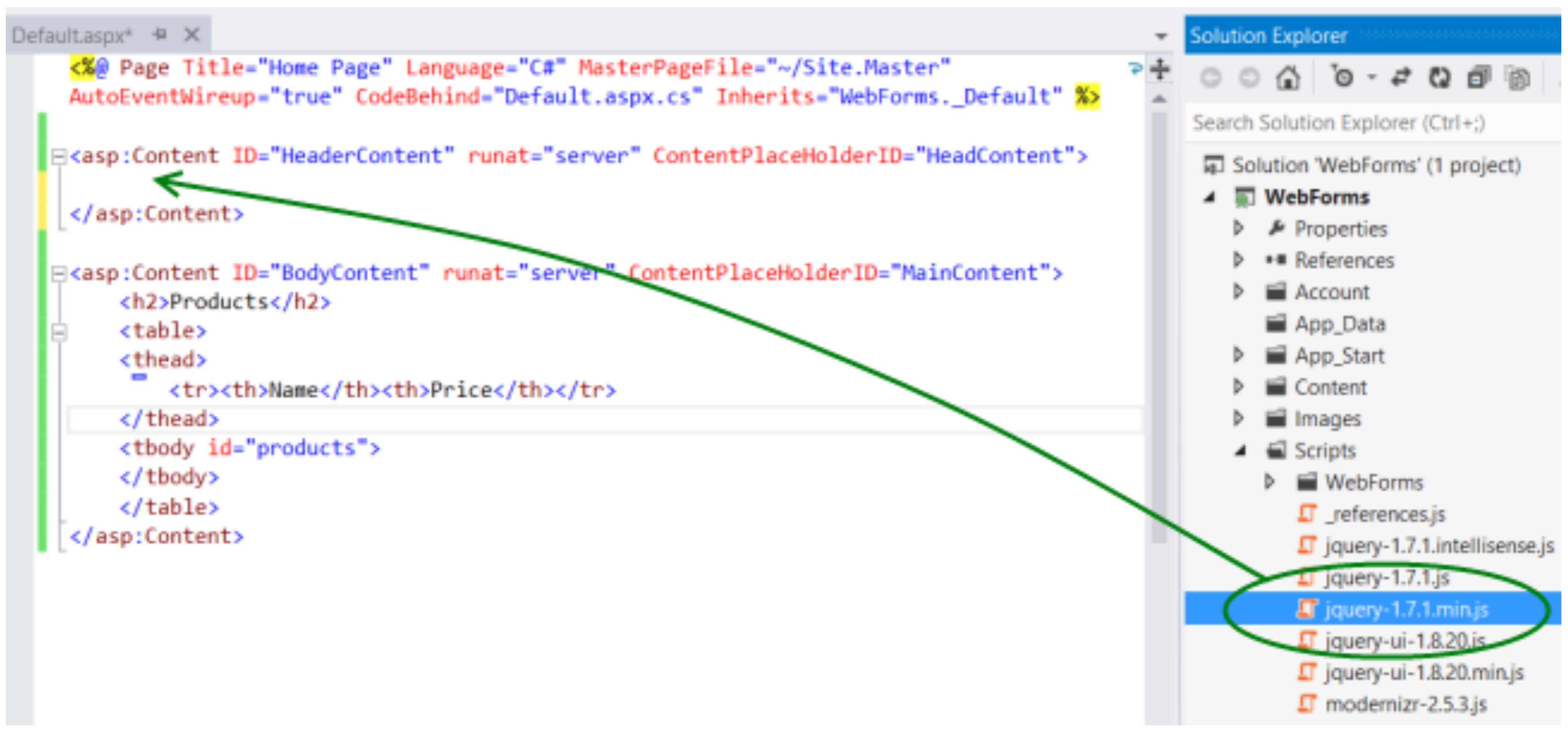
Open the file Default.aspx. Replace the boilerplate text that is in the main content section, as shown:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebForms._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>

<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>Products</h2>
    <table>
    <thead>
        <tr><th>Name</th><th>Price</th></tr>
    </thead>
    <tbody id="products">
    </tbody>
    </table>
</asp:Content>
```

Next, add a reference to the jQuery source file in the `HeaderContent` section:

```
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
    <script src="Scripts/jquery-1.7.1.min.js" type="text/javascript"></script>
</asp:Content>
```

Note: You can easily add the script reference by dragging and dropping the file from **Solution Explorer** into the code editor window.



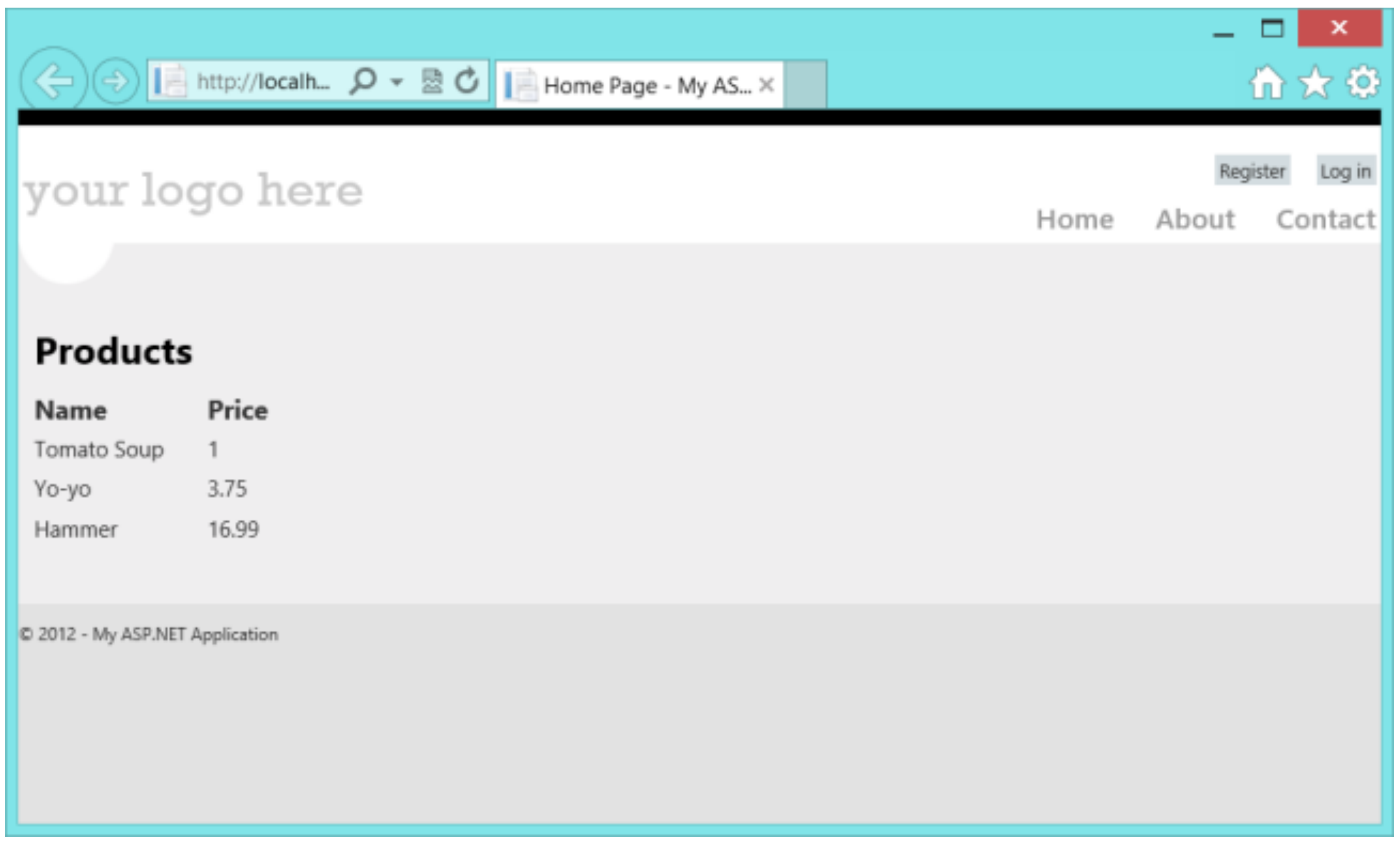
```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebForms._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">

</asp:Content>

<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>Products</h2>
    <table>
    <thead>
        <tr><th>Name</th><th>Price</th></tr>
    </thead>
    <tbody id="products">
    </tbody>
    </table>
</asp:Content>
```

Below the jQuery script tag, add the following script block:

```
<script type="text/javascript">
    function getProducts() {
        $.getJSON("api/products",
            function (data) {
                $('#products').empty(); // Clear the table body.

                // Loop through the list of products.
                $.each(data, function (key, val) {
                    // Add a table row for the product.
                    var row = '<td>' + val.Name + '</td><td>' + val.Price + '</td>';
                    $('<tr/>', { text: row })  // Append the name.
                        .appendTo($('#products'));
                });
            });
        }

        $(document).ready(getProducts);
</script>
```

When the document loads, this script makes an AJAX request to "api/products". The request returns a list of products in JSON format. The script adds the product information to the HTML table.

When you run the application, it should look like this:



Products

Name	Price
Tomato Soup	1
Yo-yo	3.75
Hammer	16.99

© 2012 - My ASP.NET Application

Next Steps

You can make your web API available over the Internet by deploying it to a hosting provider. Microsoft offers free web hosting for up to 10 web sites in a free Windows Azure trial account. For information about how to deploy a Visual Studio web project to a Windows Azure Web Site, see Deploying an ASP.NET Web Application to a Windows Azure Web Site.

Like 15 Tweet 4

Author Information

Mike Wasson – Mike Wasson is a programmer-writer at Microsoft.

Comments (13)

You must be logged in to leave a comment.

Show Comments

Telerik Sitefinity
Develop a Full Mobile Strategy on a CMS Platform!
Responsive Design | Mobile Apps | Mobile Sites
Download Trial