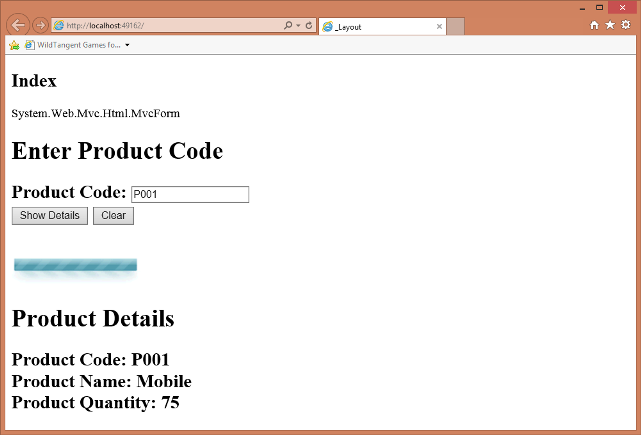
Read up on AJAX.Beginform



## Introduction

This tip explains how we can implement AJAX in ASP.NET MVC. Here, I have attempted to explain the concept of partial page updates using a partial view to display product details asynchronously. For simplicity, I have stored details of three products statically in a list. The application searches the product code entered by a user and displays the corresponding product details.

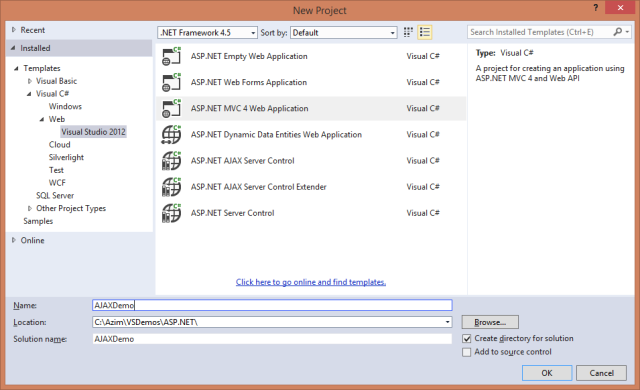
## Background

Traditional web applications rely on synchronous method calls to process data. The entire web page is posted to the server each time data is submitted through the page, causing performance delays.

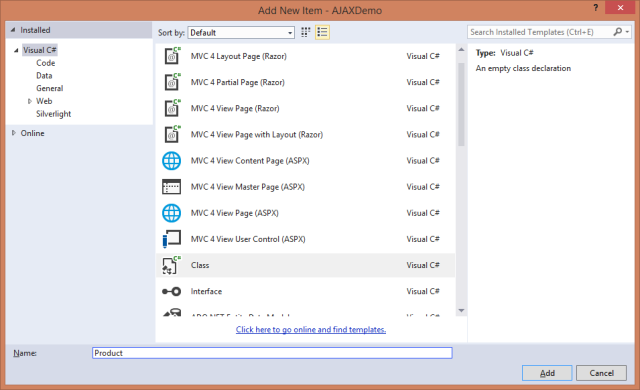
This problem can be overcome by using AJAX. AJAX allows web applications to call methods asynchronously. Instead of posting the entire page to the web server, only the required data is posted. This improves the overall performance of the web applications.

## Using the Code

Begin by creating a new ASP.NET MVC4 project as follows:



We will use a model class to display data in a partial view. For this, we have to add a Product class in the Modelsfolder as follows by right clicking on the Modelsfolder and selecting the Add Class option:



Following is the code of the Product class:

Hide   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace AJAXDemo.Models

{

public class Product

{

public string ProdCode

{

get;

set;

}

public string ProdName

{

get;

set;

}

public int ProdQty

{

get;

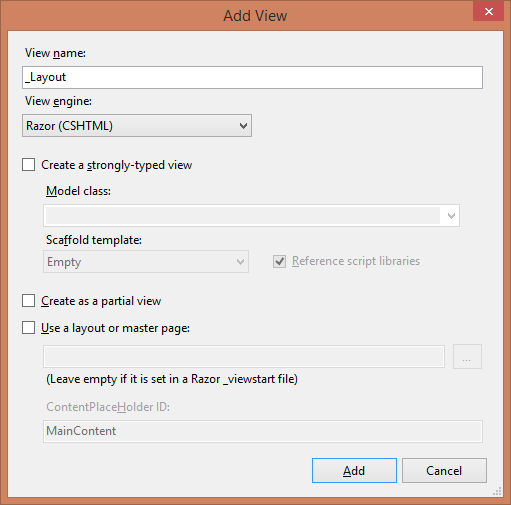
set;

}

}

}

Then, add new layout in the Shared folder under the Viewsfolder as follows:



Following is the code of the \_Layout.cshtmlfile:

Hide   Copy Code

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>\_Layout</title>

@RenderSection("scripts")

</head>

<body>

<div>

@RenderBody()

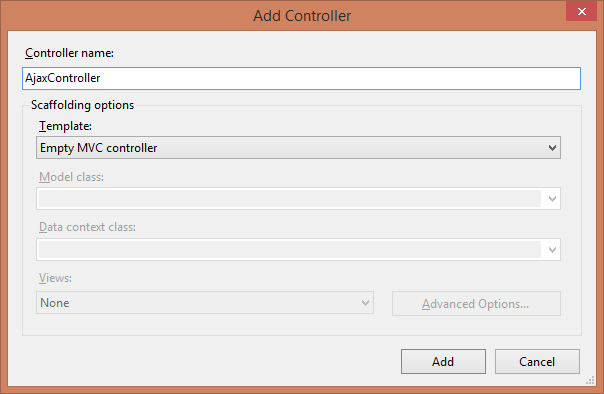
</div>

</body>

</html>

This layout will be used by the Index view. In this layout, the RenderSection() method defines a placeholder for the scripts section and the RenderBody() method defines a placeholder for the main content of the Index view.

Add a new controller as follows by right clicking on the Controllersfolder:



Following is the code of the Index() method:

Hide   Copy Code

public ActionResult Index()

{

Product p1 = new Product { ProdCode = "P001", ProdName = "Mobile", ProdQty = 75 };

Product p2 = new Product { ProdCode = "P002", ProdName = "Laptop", ProdQty = 125 };

Product p3 = new Product { ProdCode = "P003", ProdName = "Netbook", ProdQty = 100 };

prodList.Add(p1);

prodList.Add(p2);

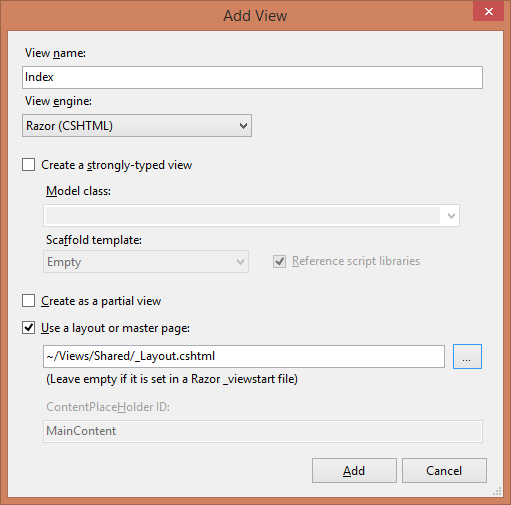
prodList.Add(p3);

return View();

}

This code creates three product objects and adds them to a static list called prodList which is declared at the class level and it renders the view associated with the Index view.

Add the view corresponding to the above action method as follows by right clicking on the Index() method in the controller:  
(**Note**: Remember to select the layout page while adding the Index view.)



Following is the code of the Index.cshtml view:

Hide   Copy Code

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@section scripts

{

<script type="text/javascript" src="~/Scripts/jquery-1.7.1.js"></script>

<script type="text/javascript" src="~/Scripts/jquery.unobtrusive-ajax.js"></script>

}

<h2>Index</h2>

@Ajax.BeginForm("ShowDetails","Ajax",

new AjaxOptions

{ HttpMethod = "POST", UpdateTargetId = "div1",

InsertionMode = InsertionMode.Replace, LoadingElementId = "waitimage" })

<h1>Enter Product Code</h1>

<h2>

Product Code: @Html.TextBox("txtCode")<br />

<input type="submit" value="Show Details" />

<input type="reset" value="Clear"/>

</h2>

@{Html.EndForm();}

<br />

<img id="waitimage" src="~/Images/loading42.gif" style="display:none" />

<br />

<div id="div1">

</div>

The above view references the jquery.unobtrusive-ajax.jsscript to implement the AJAX functionality. It also uses a div with the id "div1" to display the requested product details. It uses the Ajax.BeginForm() helper method to accept product code and submit it asynchronously. The first parameter of the Ajax.BeginForm() method specifies the name of the action method to be invoked to process the input data, the second parameter specifies the name of the controller and the third parameter specifies the AjaxOptions to be used. In the AjaxOptions, the HttpMethod property specifies the HTTP method to be used, the UpdateTargetId property specifies the ID of the div element where the partial view will be displayed, the InsertionMode property specifies the mode of insertion of the partial view, and the LoadingElemetId property specifies the ID of the image to be displayed while waiting for the asynchronous call to be completed and the view to be rendered.

The InsertionMode property can have one of the following values:

* InsertionMode.InsertAfter - The new details are added after the existing details
* InsertionMode.InsertBefore - The new details are added before the existing details
* InsertionMode.Replace - The new details replace the existing details

In order to process the input and search the product code entered by the user, we add a method in the controller which returns a partial view as follows:

Hide   Copy Code

public PartialViewResult ShowDetails()

{

System.Threading.Thread.Sleep(3000);

string code = Request.Form["txtCode"];

Product prod = new Product();

foreach(Product p in prodList)

{

if (p.ProdCode == code)

{

prod = p;

break;

}

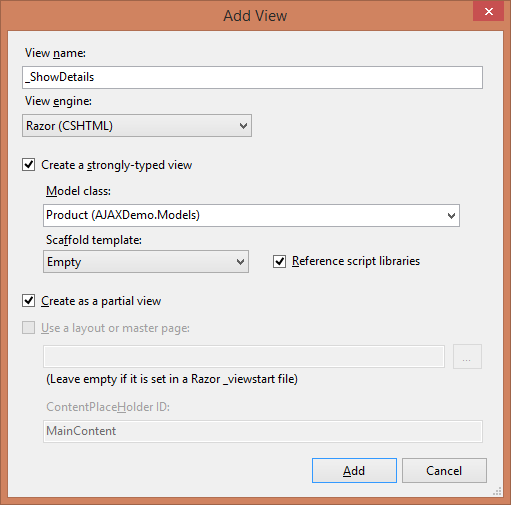
}

return PartialView("\_ShowDetails", prod);

}

The above code searches the entered product code and returns the product corresponding to it using a partial view. It uses the Thread.Sleep() method to simulate delay in fetching the product details. This allows you to view the page loading animation while waiting for the asynchronous method call to complete.

In order to display the product details, we have to add a strongly typed partial view as follows by right clicking on the method and selecting the Add View option:



Following is the code of the \_ShowDetails view:

Hide   Copy Code

@model AJAXDemo.Models.Product

@if(Model.ProdCode==null)

{

<h1>Invalid Product Code</h1>

}

else

{

<h1>Product Details</h1>

<h2>

Product Code: @Model.ProdCode<br />

Product Name: @Model.ProdName<br />

Product Quantity: @Model.ProdQty<br />

</h2>

}

The above code displays the product details if the product code is found and the message "Invalid Product Code" if it is not found.

Before executing the application, we have to specify the routing information in the RouteConfig.csfile in the App\_Startfolder as follows:

Hide   Copy Code

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Ajax", action = "Index", id = UrlParameter.Optional }

);

Also, we need to enable unobtrusive JavaScript by adding the "UnobtrusiveJavaScriptEnabled" key under the <configuration> element in the Web.configfile as follows:

Hide   Copy Code

<appSettings>

<add key="webpages:Version" value="2.0.0.0" />

<add key="webpages:Enabled" value="false" />

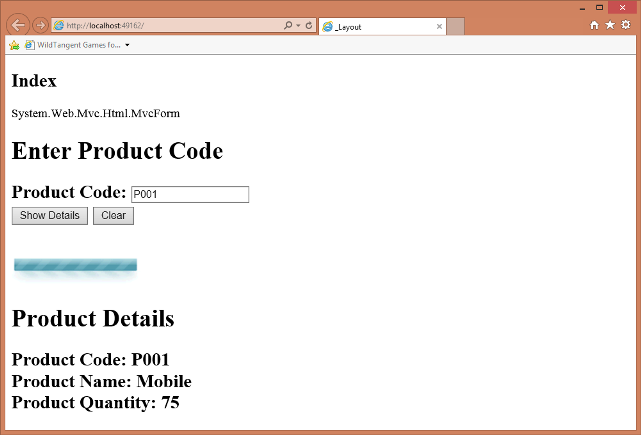
<add key="PreserveLoginUrl" value="true" />

<add key="ClientValidationEnabled" value="true" />

<add key="UnobtrusiveJavaScriptEnabled" value="true" />

</appSettings>

On execution, the application produces the following output:



Entering a product code and clicking the submit button displays the corresponding product details asynchronously.

## Points of Interest

AJAX is a very important feature of web applications and it provides significant performance advantage over traditional web applications.

I have developed the application using Microsoft Visual Studio Express 2013 for Web.

Exploring key features/capabilities of ASP.NET and getting answers for **ASP.NET Interview Questions** in this Web Development Tutorial series, we have reached this Part-5 with a set of questions on ASP.NET and AJAX (Asynchronous JavaScript and XML).What we have covered so far can be summarized as follows:

So, let’s keep exploring further ASP.NET technology concepts through Interview Questions Series.

*For a comprehensive list of****ASP.NET MVC Interview Questions****,*[*follow here*](http://www.webdevelopmenthelp.net/2013/09/top-asp-net-mvc-interview-questions.html)*.*

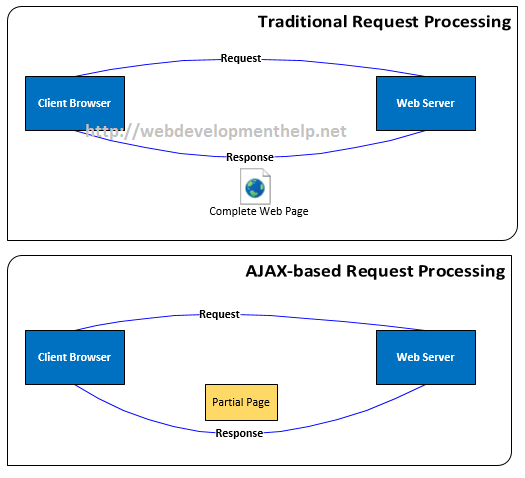
**ASP.NET AJAX Interview Questions List**

1. [Define AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q1)
2. [Please elaborate XMLHttpRequest Object further?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q2)
3. [How to send a request to server using XMLHttpRequest Object?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q3)
4. [What is ASP.NET AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q4)
5. [Difference between Synchronous and Asynchronous Postback?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q5)
6. [What are the basic controls in ASP.NET AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q6)
7. [What is a ScriptManager in ASP.NET AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q7)
8. [ScriptManager Vs ScriptManagerProxy?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q8)
9. [What is the role of UpdatePanel in ASP.NET AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q9)
10. [What are the limitations of AJAX?](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#Q10)

**Define AJAX?**

AJAX stands for “Asynchronous JavaScript and XML”. It’s basically a technique for creating Rich Internet Applications (RIA) that are faster as well as more interactive, using a combination of commonly used techniques as HTML/XHTML, CSS, Document Object Model (DOM), JavaScript, XML/XSLT and XMLHttpRequest object.

**XMLHttpRequest** object is the key basis of AJAX and makes it possible to communicate with the web server asynchronously and exchange data. As compared to a traditional request which returns a complete web page,  partial web page is returned as response to an AJAX request.



[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**::::: Free Practical Guide to ASP.NET MVC Web API (**[**Web Edition**](http://www.webdevelopmenthelp.net/2014/02/practical-guide-asp-net-web-api.html)**|**[**PDF Download**](https://docs.google.com/file/d/0BzG_9iK33sJSS3EzWVNHRl9wRDA/edit)**) :::::**

**Please elaborate XMLHttpRequest Object further?**

XMLHttpRequest is the core object in AJAX technology regardless of any implementation. XMLHttpRequest object is used to exchange data with a server seamlessly. Basically JavaScript uses this Object to exchange XML as well as text data between client and server. An AJAX implementation uses this object and communicate with server but it doesn’t require the complete page to be refreshed.

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**How to send a request to server using XMLHttpRequest Object?**

We can send a request to server using HTTP GET and POST methods as follows:

***//Simple GET Request***

*var xmlHttp = new XMLHttpRequest();*

*xmlHttp.open(“GET”, “TestFile.txt”, true);*

*xmlHttp.send();*

***//Simple POST Request***

*var xmlHttp = new XMLHttpRequest();*

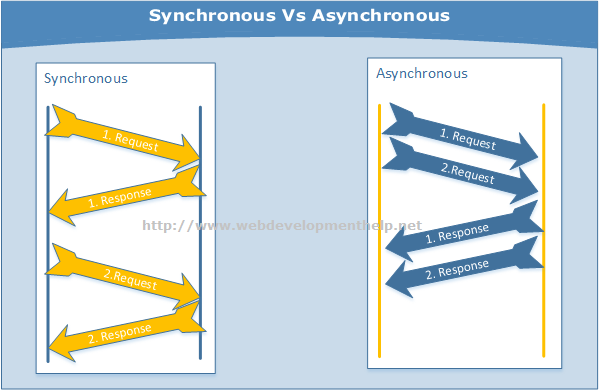
*xmlHttp.open(“POST”, “TestFile.txt”, true);*

*xmlHttp.send();*

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**Difference between Synchronous and Asynchronous Postback?**

In Synchronous postback, complete web page is sent to server and in return rendering the output (i.e. complete page), whereas in case of Asynchronous postback, partial page goes to the server and renders only partial (required) part of the page.

Normally a method making Synchronous call always waits for response to do next task (i.e. might be another call). On the other hand, for Asynchronous, no waiting required as illustrated by below figure:

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**What are the basic controls in ASP.NET AJAX?**

Following controls can be considered as core AJAX controls in ASP.NET.

* ScriptManager
* ScriptManagerProxy
* UpdatePanel
* UpdateProgress
* Timer

Later more controls are added to ASP.NET AJAX library e.g. Script Loader, Client Data Context, Client Data Access, jQuery Integration etc. For complete reference, look for [AJAX Control Toolkit](https://ajaxcontroltoolkit.codeplex.com/).

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**What is a ScriptManager in ASP.NET AJAX?**

In order to use AJAX functionality on a web page, we add a ScriptManager control to the page in most of the scenarios, because ScriptManager control register AJAX library scripts to that particular web page. We can have only one ScriptManager per page.

*<asp:ScriptManager ID=”ScriptManager1″ runat=”server”></asp:ScriptManager>*

ScriptManager basically manages all ASP.NET AJAX resources of a web page, creates proxies for asynchronous web service call and also manages partial page updates… etc.

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**ScriptManager Vs ScriptManagerProxy?**

As we understand that we can have only one **ScriptManager** control on a page but we can have multiple **ScriptManagerProxy** controls.  
Consider a scenario that we have ScriptManager in our MasterPage that is available for all content pages. Now, we wanted to register a web service in a particular page. So, we will not add another ScriptManager to that page instead we will add ScriptManagerProxy to it in order to avoid error.  
[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**What is the role of UpdatePanel in ASP.NET AJAX?**

UpdatePanel is the control that facilitate the partial page rendering functionality in an ASP.NET application. As discussed earlier that using ASP.NET AJAX, we can communicate with a web server asynchronously and update a part of a page without a complete page postback. In order to apply partial page update/rendering, we can add one or more UpdatePanel controls to our ASP.NET Page as follows:

*<asp:ScriptManager ID=”ScriptManager1″ runat=”server”></asp:ScriptManager>*

*<asp:UpdatePanel ID=”UpdatePanel1″ runat=”server”>*

*<ContentTemplate>*

*<asp:Label ID=”lblPanel” runat=”server” Text=”Update Panel Added.”></asp:Label><br />*

*<asp:Button ID=”btnTestButton”*  
*runat=”server”*  
*OnClick=”btnTestButton\_Click”*  
*Text=”Test Button” />*

*</ContentTemplate>*

*</asp:UpdatePanel>*

[Back to top](http://www.webdevelopmenthelp.net/2014/04/asp-net-interview-questions-part5.html#top)

**What are the limitations of AJAX?**

* AJAX on an application will not work if JavaScript is disabled.
* In some scenarios, it exposes vulnerability.
* It will always be difficult to bookmark application state.
* Application behavior may be slow in some scenarios, because of different loading time of controls on a single page.