Web API
Lab Book

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning HTML version x.x. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for HTML

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 6.0 or higher
- MS-Access/Connectivity to Oracle database
- Apache Tomcat Version 5.0.

Please ensure that the following is done:

- A text editor like Notepad or Eclipse is installed.
- JDK 1.4 is installed. (This path is henceforth referred as <java_install_dir>)
- Apache Tomcat is installed but not started (Pls. Refer below on how to install Tomcat).

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory html_assgn. For each lab exercise create a directory as lab <lab number>.
- Download all files required to complete assignments from: http://pace.patni.com/TechRS/download.asp?course=Internet_HTML
- You may also look up the on-line help provided in the MSDN library.

Learning More (Bibliography if applicable)

- HTML Source Book by Ian S. Graham
- HTML: Complete Concepts and Techniques by Gary B. Shelly
- HTML: The Definitive Guide by Chuck Musciano
- Dynamic HTML: The Definitive Reference by Danny Goodman
- HTML: The Complete Reference by Thomas A. Powell

Problem Statement/ Case Study (If applicable)

Give the case study used for this lab book here. If applicable.

Lab 1.Web API Basics

Goals	Understand the process of creating and consuming a Web API Learn to create a Asp.Net Web API Learn to consume Asp.Net Web API in .Net Client Application
Time	60 minutes

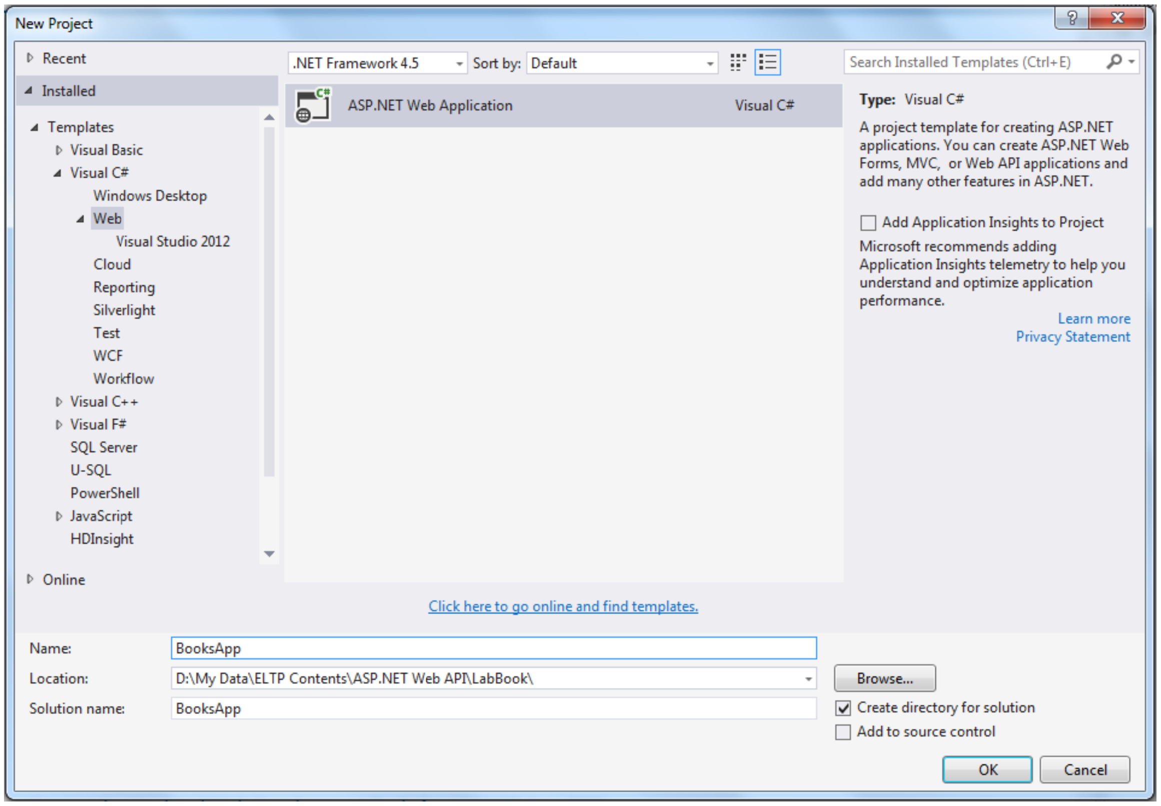
1) Creating a Asp.Net Web API

Create a Asp.Net Web API Which will allow you to retrieve details of Books

Solution:-

Open Visual Studio 2013 and Create the a new Web Api Project following the following sequence

File →New → Project→ Installed Templates →Select Web.
In the list project template select ASP.NET Web Application name the project as BooksApp and Click OK



Type: Visual C#

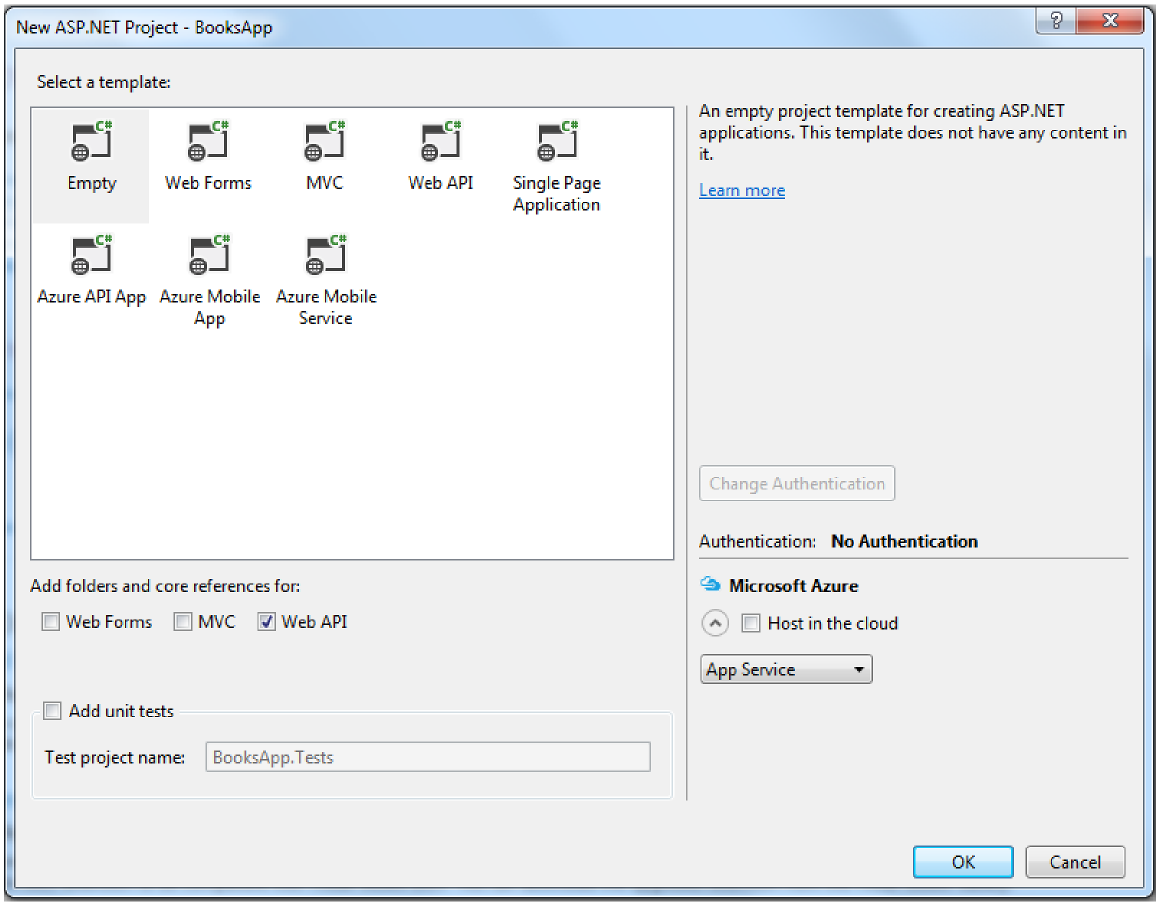
A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

☐ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Name: BooksApp
Location: D:\My Data\ELTP Contents\ASP.NET Web API\LabBook\
Solution name: BooksApp
☑ Create directory for solution
☐ Add to source control

In the Asp.Net Project Dialog ,Select the Empty template. Under the “Add Folders and core reference” check Web API and click OK



New ASP.NET Project - BooksApp

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Learn more

Authentication: **No Authentication**

Add folders and core references for:

☐ Web Forms ☐ MVC ☑ Web API

☐ Add unit tests

Test project name: BooksApp.Tests

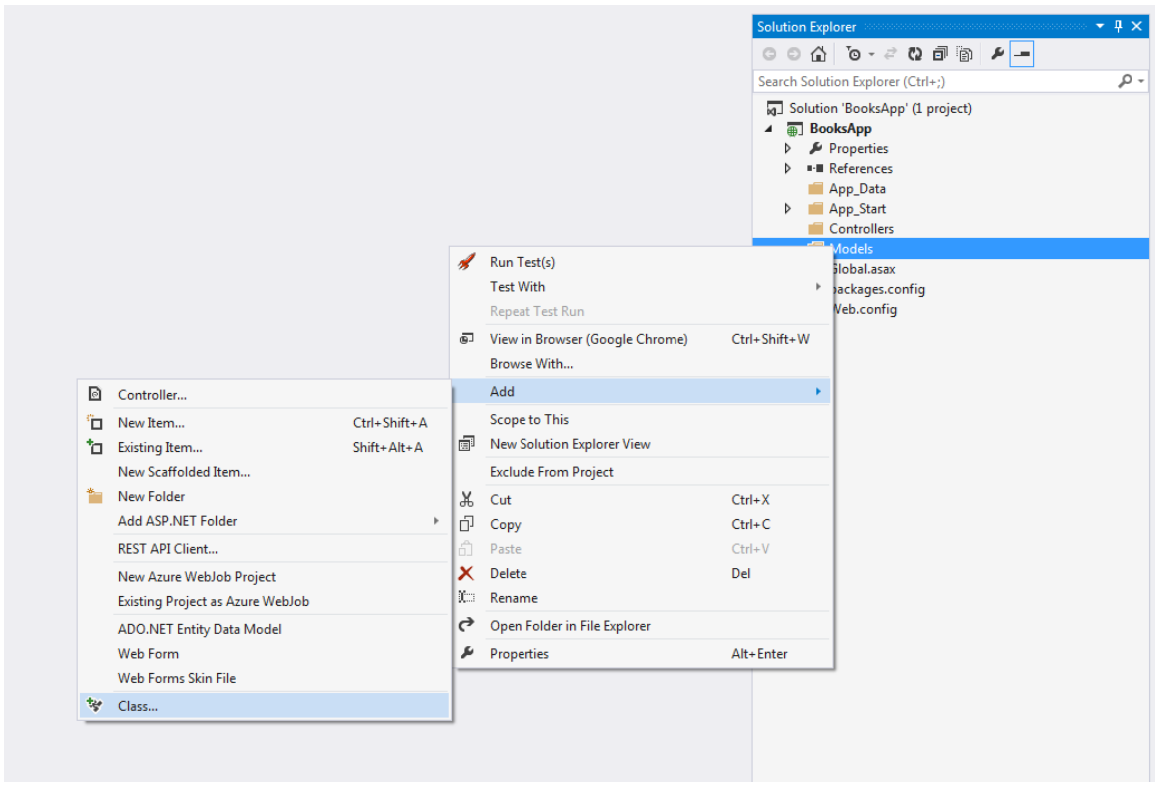
Microsoft Azure

☐ Host in the cloud

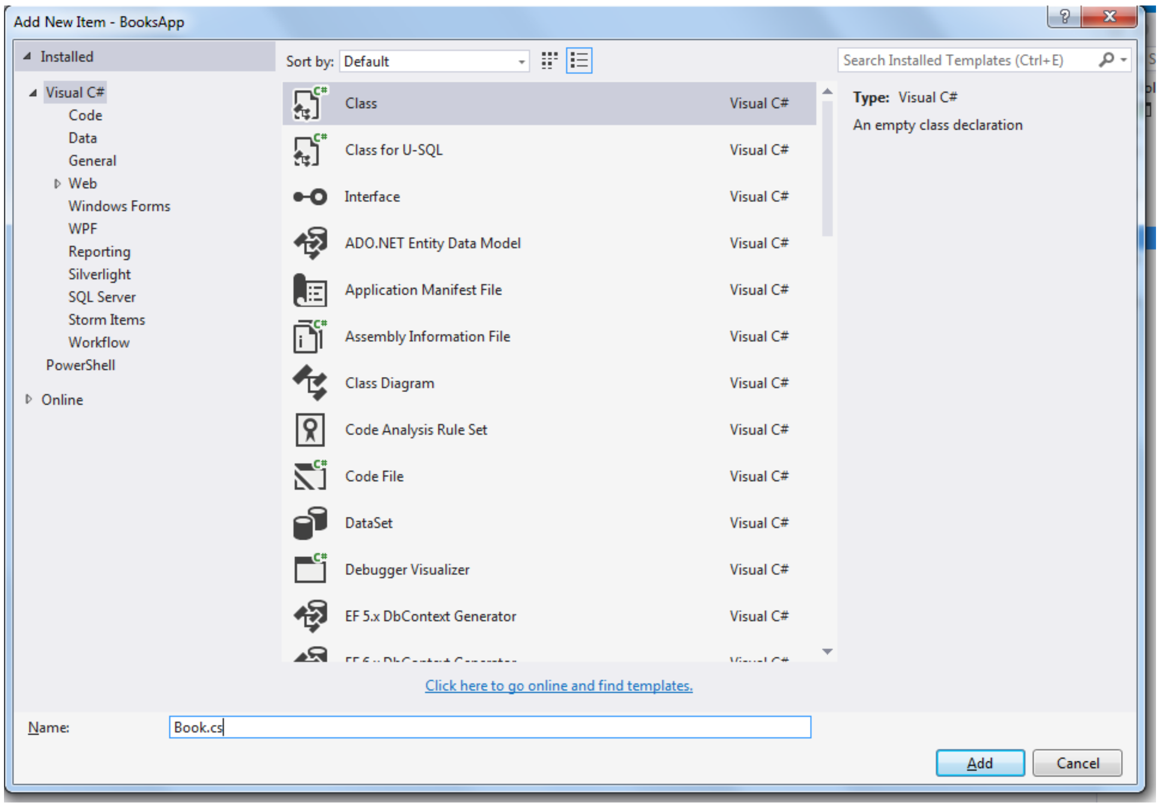
App Service

Now After creating the project we have to add a Model to the project for that we have to follow following steps

In the Solution Explorer select Model folder and Right Click on the Model Folder select Add → Class



In the Add New Dialog Box Name the class as Book


Add New Item - BooksApp

Installed
- Visual C#
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Silverlight
  - SQL Server
  - Storm Items
  - Workflow
- PowerShell

Online

Sort by: Default

Search Installed Templates (Ctrl+E)

Class	Visual C#
Class for U-SQL	Visual C#
Interface	Visual C#
ADO.NET Entity Data Model	Visual C#
Application Manifest File	Visual C#
Assembly Information File	Visual C#
Class Diagram	Visual C#
Code Analysis Rule Set	Visual C#
Code File	Visual C#
DataSet	Visual C#
Debugger Visualizer	Visual C#
EF 5.x DbContext Generator	Visual C#

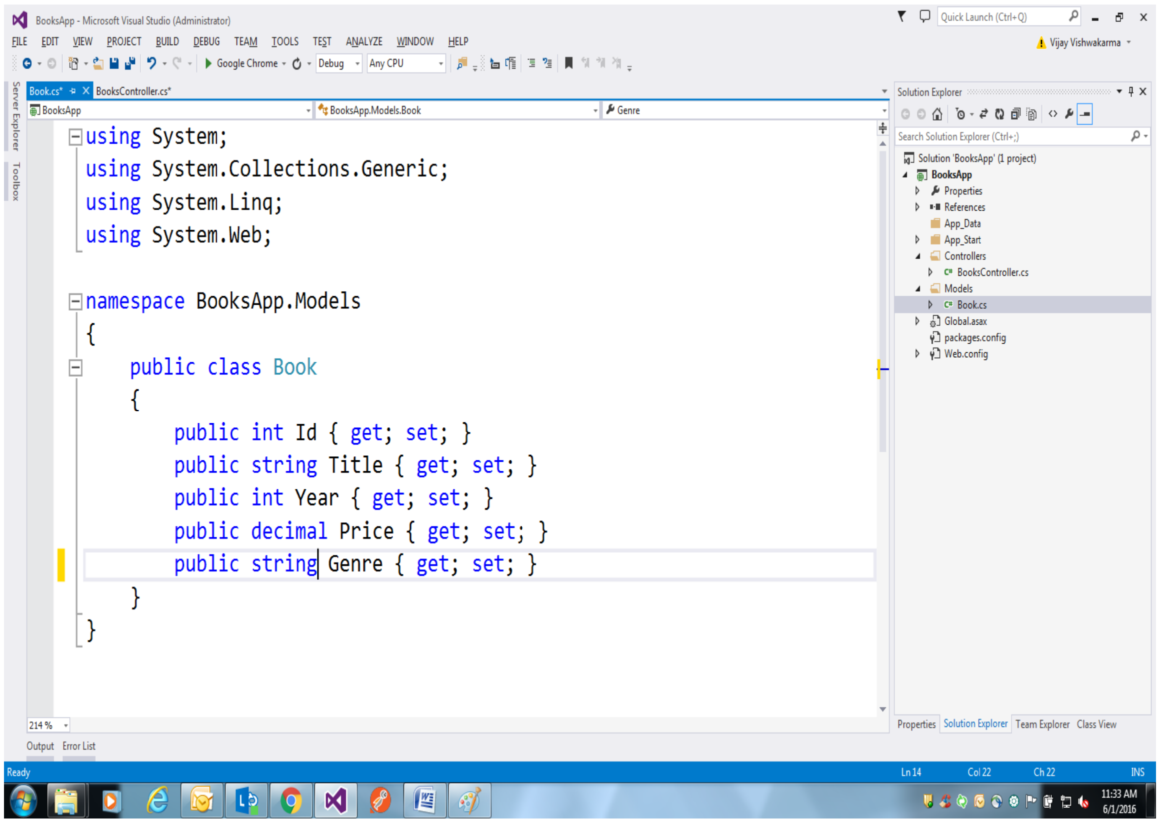
Type: Visual C#

An empty class declaration

Click here to go online and find templates.

Name: Book.cs

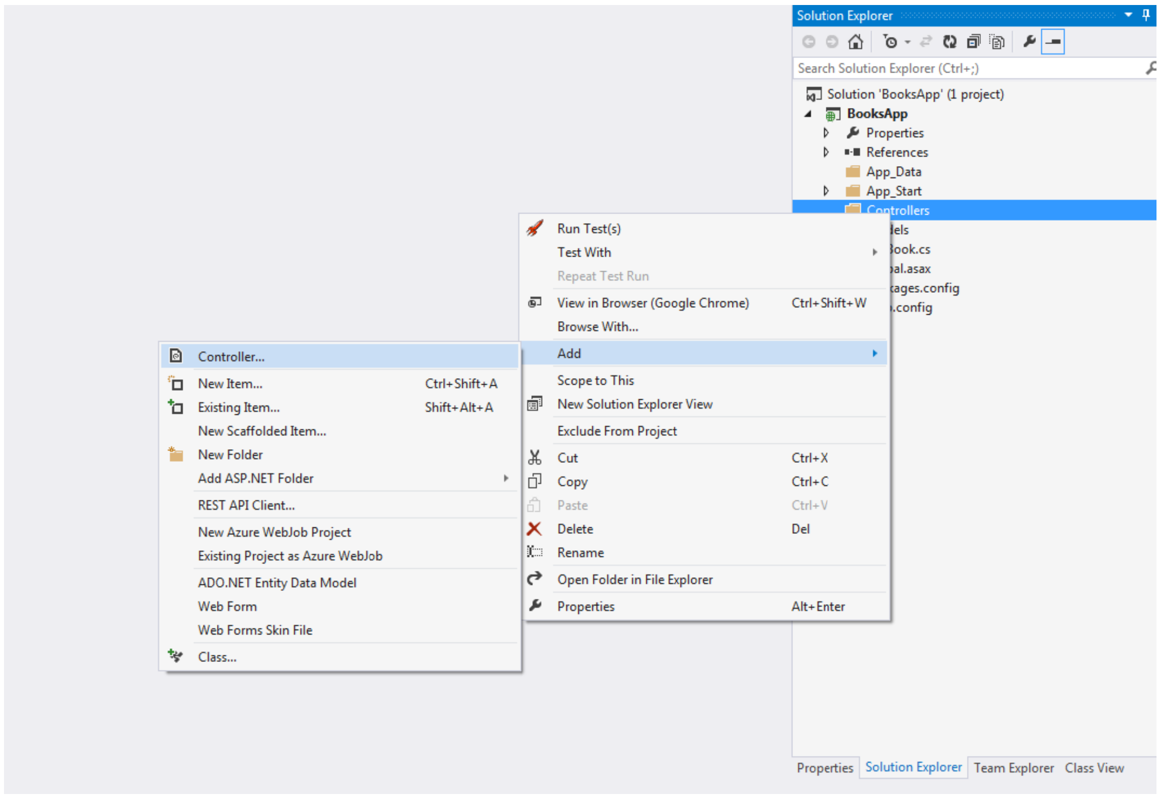
Now add the following code in the Book.cs



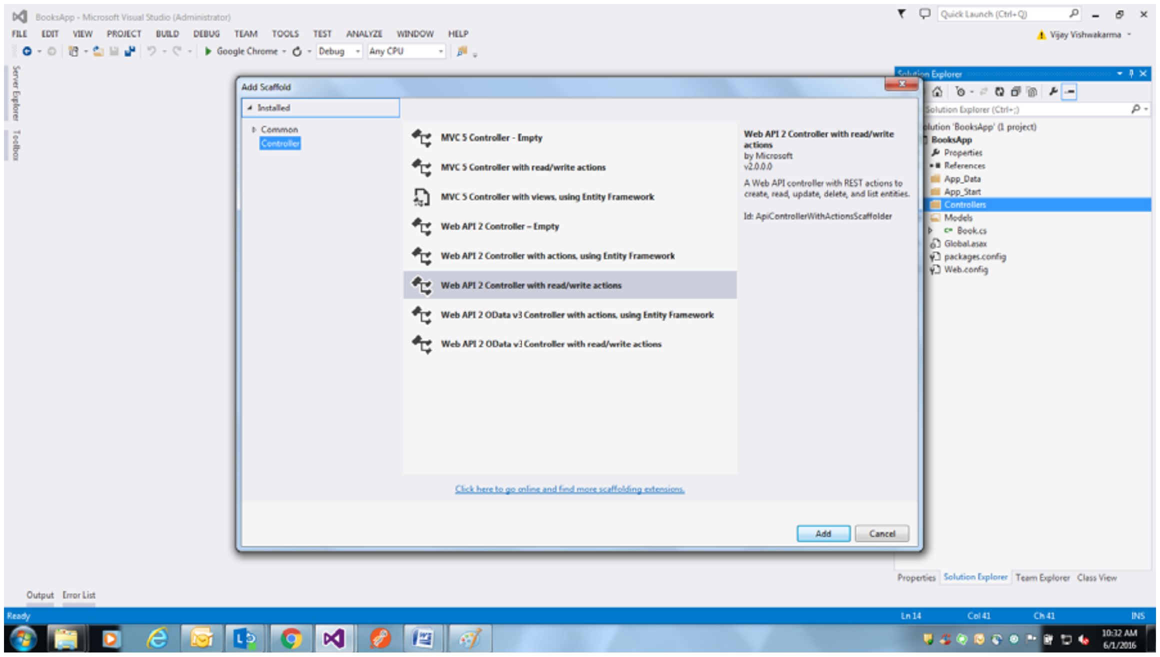
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace BooksApp.Models
{
    public class Book
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int Year { get; set; }
        public decimal Price { get; set; }
        public string Genre { get; set; }
    }
}
```

Now we have add Web API Controller to the project which contain all the actions

To Add the Controller to the project select and right click on the Controller folder Add→Controller



In the Add Scaffold select Web API 2 Controller – Empty and click on ADD


Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData v3 Controller with actions, using Entity Framework
Web API 2 OData v3 Controller with read/write actions
Click here to go online and find more scaffolding extensions.
Add
Cancel
Solution Explorer
BooksApp
Properties
References
App_Data
App_Start
Controllers
Models
Book.cs
Global.asax
packages.config
Web.config

Web API 2 Controller with read/write actions
by Microsoft
v2.0.0.0
A Web API controller with REST actions to create, read, update, delete, and list entities.
Id: ApiControllerWithActionsScaffolder

There are other options which are as follows

Web API 2 Controller –Empty:-Adds a empty Controller with no Read/Write Action

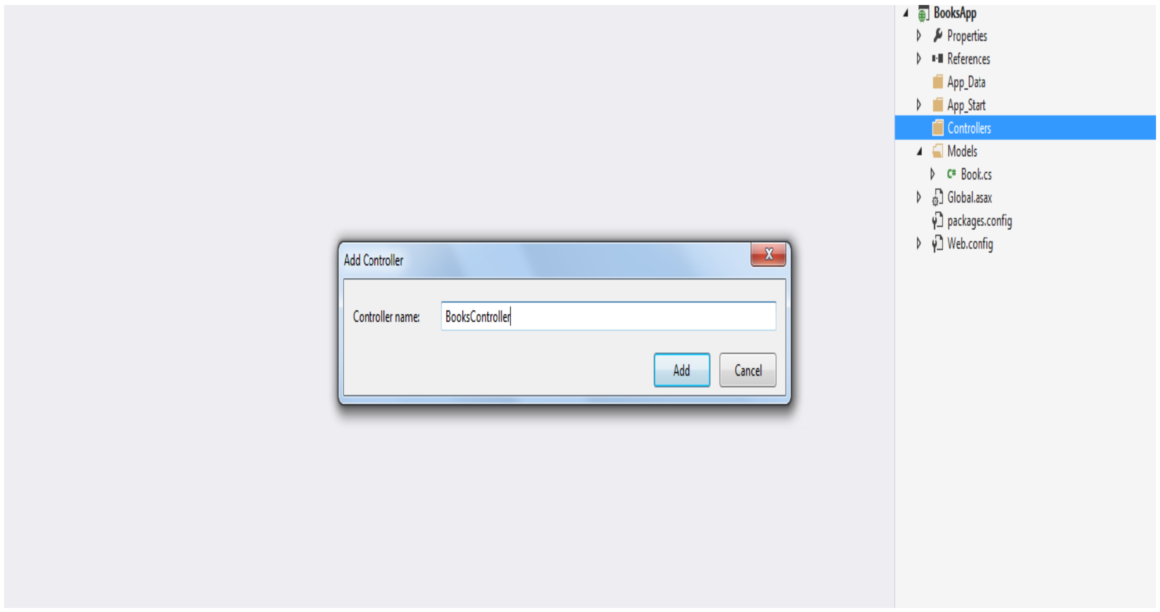
Web API 2 Controller with action using Entity Framework: - Adds a Controller with Read/Write Action based on the Entity data Model specified

Web API 2 Controller with read/write action :- Add a Controller with implementation of Read/Write Action which can be modified as per requirement

Every Controller class is inherited from ApiController Class

ApiController Class Defines properties and methods for API Controller. It is available in System.Web.Http Namespace

After clicking on Add it will ask for naming the Controller Name it as BooksController

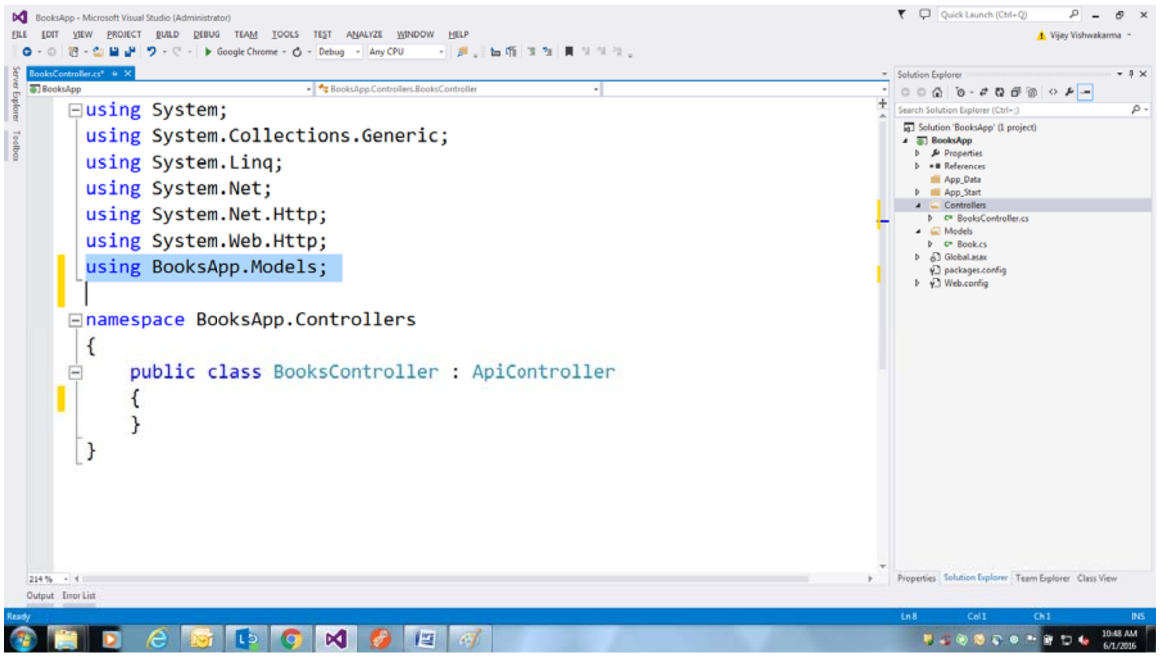


Controller name: BooksController

Now in the books controller class Add the following Namespace

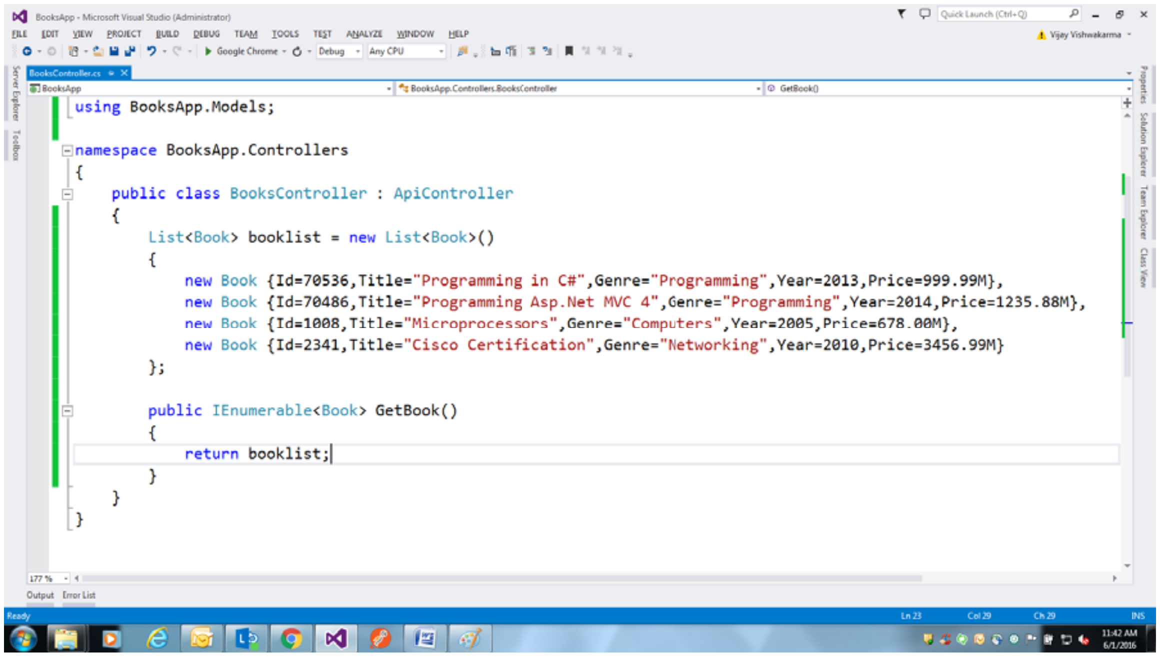
```
using BooksApp.Models;
```

This namespace will allow you a have access to Book Model class



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using BooksApp.Models;

namespace BooksApp.Controllers
{
    public class BooksController : ApiController
    {
    }
}
```

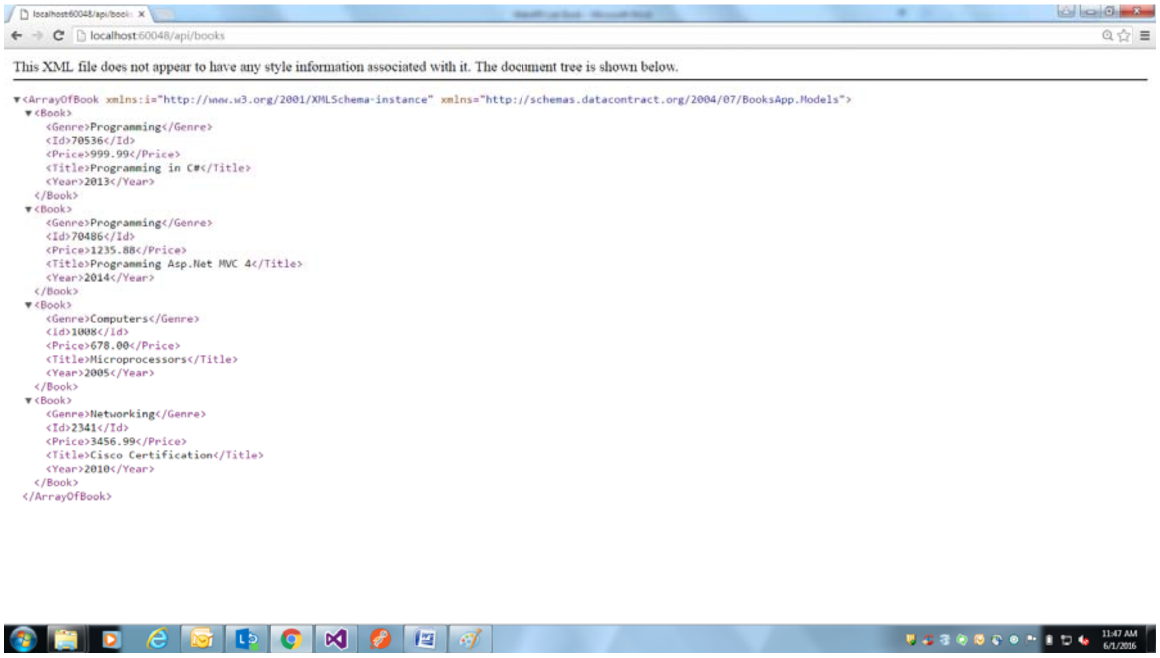
Now we will add a List collection which we used as repository of Books in our application in Book.cs File and add Action Method to return list of Books



```
using BooksApp.Models;

namespace BooksApp.Controllers
{
    public class BooksController : ApiController
    {
        List<Book> booklist = new List<Book>()
        {
            new Book {Id=70536,Title="Programming in C#",Genre="Programming",Year=2013,Price=999.99M},
            new Book {Id=70486,Title="Programming Asp.Net MVC 4",Genre="Programming",Year=2014,Price=1235.88M},
            new Book {Id=1008,Title="Microprocessors",Genre="Computers",Year=2005,Price=678.00M},
            new Book {Id=2341,Title="Cisco Certification",Genre="Networking",Year=2010,Price=3456.99M}
        };

        public IEnumerable<Book> GetBook()
        {
            return booklist;
        }
    }
}
```

Now Press F5 or click on the run button in Visual Studio to run the application.
And change the url in the browser to http://localhost:60048/api/books

You will the following output

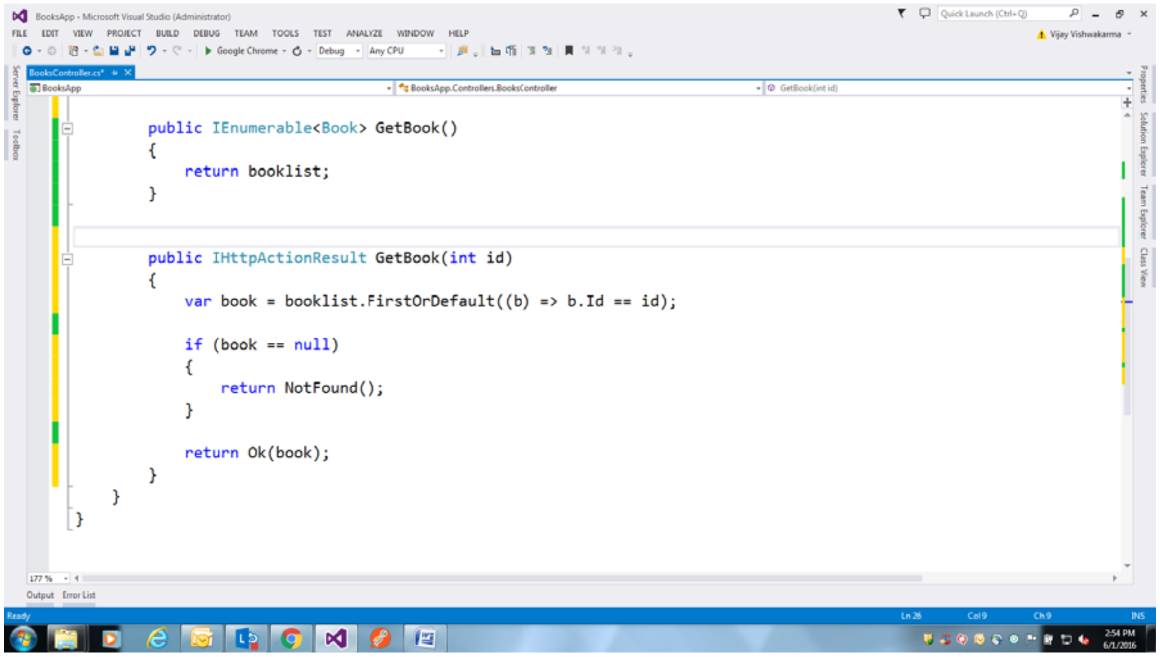


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ArrayOfBook xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/BooksApp.Models">
  ▼<Book>
      <Genre>Programming</Genre>
      <Id>70536</Id>
      <Price>999.99</Price>
      <Title>Programming in C#</Title>
      <Year>2013</Year>
    </Book>
  ▼<Book>
      <Genre>Programming</Genre>
      <Id>70486</Id>
      <Price>1235.88</Price>
      <Title>Programming Asp.Net MVC 4</Title>
      <Year>2014</Year>
    </Book>
  ▼<Book>
      <Genre>Computers</Genre>
      <Id>1008</Id>
      <Price>678.00</Price>
      <Title>Microprocessors</Title>
      <Year>2005</Year>
    </Book>
  ▼<Book>
      <Genre>Networking</Genre>
      <Id>2341</Id>
      <Price>3456.99</Price>
      <Title>Cisco Certification</Title>
      <Year>2010</Year>
    </Book>
  </ArrayOfBook>
```

If you are using Internet Explorer you will be prompted for saving or viewing the JSON file for the same.

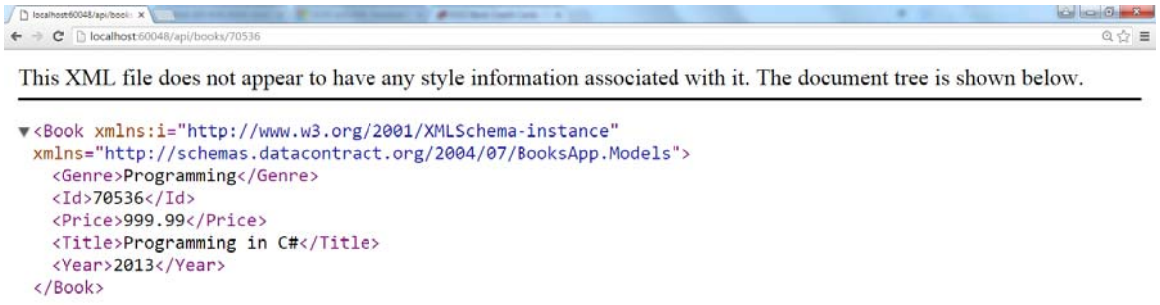
Now we will add one more Action Method for searching a book based on Bookid



```
        public IEnumerable<Book> GetBook()
        {
            return booklist;
        }

        public IHttpActionResult GetBook(int id)
        {
            var book = booklist.FirstOrDefault((b) => b.Id == id);

            if (book == null)
            {
                return NotFound();
            }

            return Ok(book);
        }
    }
}
```

Now press F5 or click on the run button in Visual Studio to run this project .
Type the following in the browser http://localhost:60048/api/books/id
eg: http://localhost:60048/api/books/70536

This is give the details of the book whose id is supplied


localhost:60048/api/books/70536

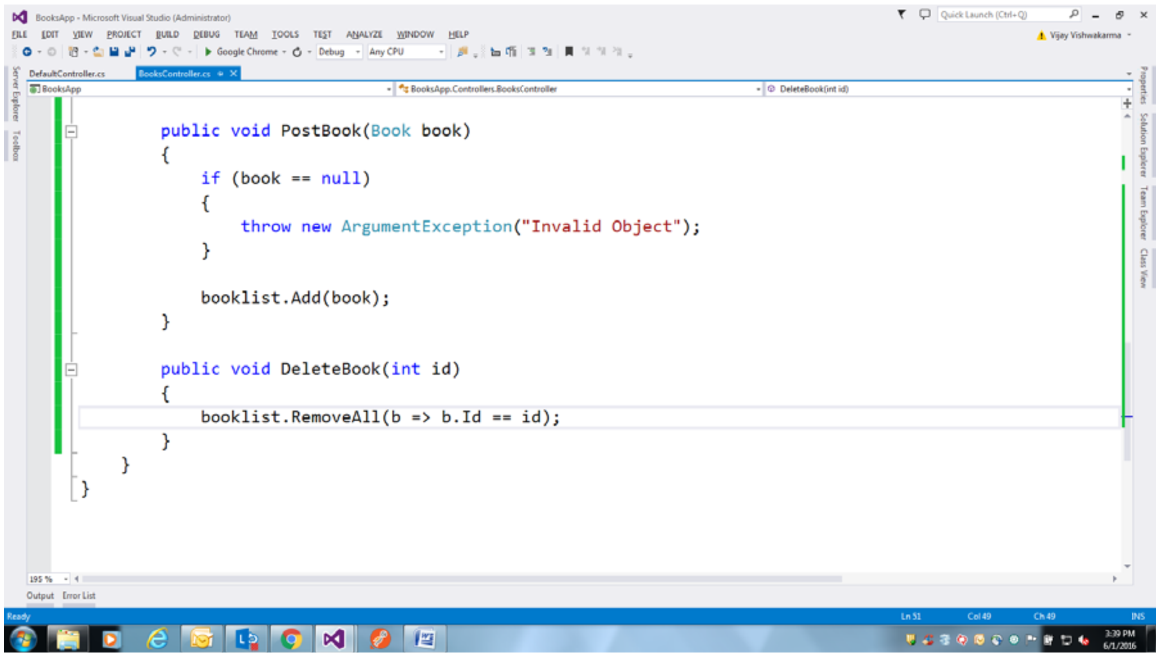
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Book xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://schemas.datacontract.org/2004/07/BooksApp.Models">
   <Genre>Programming</Genre>
   <Id>70536</Id>
   <Price>999.99</Price>
   <Title>Programming in C#</Title>
   <Year>2013</Year>
 </Book>
```

2) Consuming Web API in a Asp.Net Application using JQuery

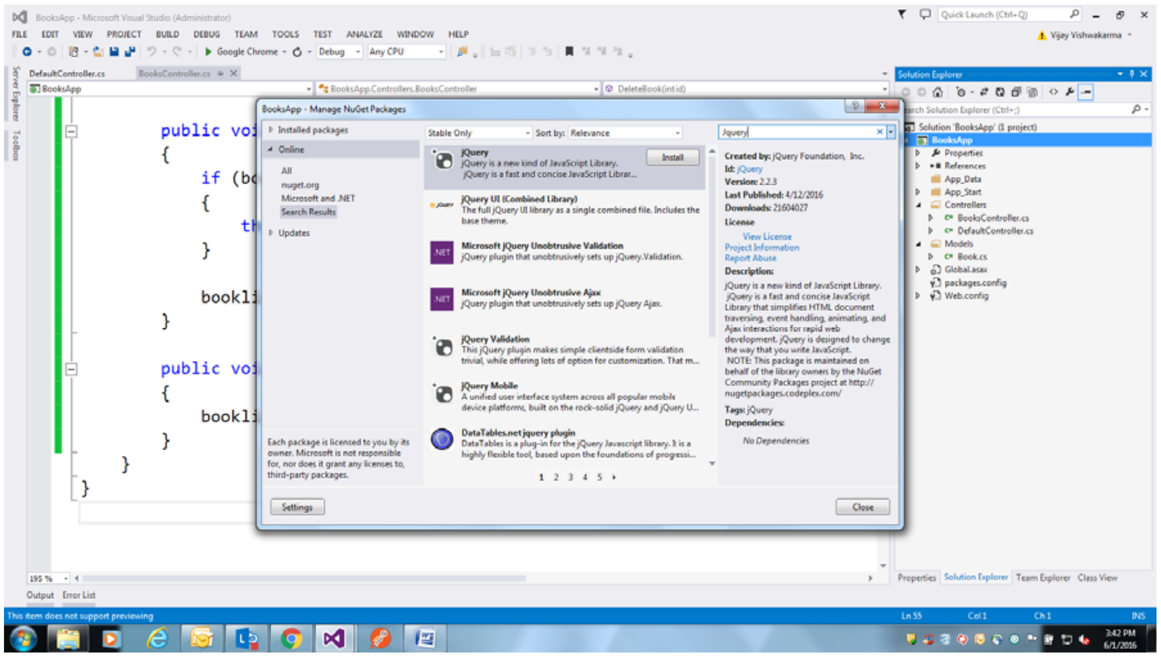
Now we have to consume Web API and perform operation like adding a Book or searching of Book.

For that we will add two Action Method which will allow us to add a new Book Record or search one.



```
        public void PostBook(Book book)
        {
            if (book == null)
            {
                throw new ArgumentException("Invalid Object");
            }

            booklist.Add(book);
        }

        public void DeleteBook(int id)
        {
            booklist.RemoveAll(b => b.Id == id);
        }
    }
}
```

Now we will add Jquery Reference to the Project using Nuget Pacakge Manager



Stable Only | Sort by: Relevance | Jquery

jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Librar...

jQuery UI (Combined Library)
The full jQuery UI library as a single combined file. Includes the base theme.

Microsoft jQuery Unobtrusive Validation
jQuery plugin that unobtrusively sets up jQuery.Validation.

Microsoft jQuery Unobtrusive Ajax
jQuery plugin that unobtrusively sets up jQuery Ajax.

jQuery Validation
This jQuery plugin makes simple clientside form validation trivial, while offering lots of option for customization. That m...

jQuery Mobile
A unified user interface system across all popular mobile device platforms, built on the rock-solid jQuery and jQuery U...

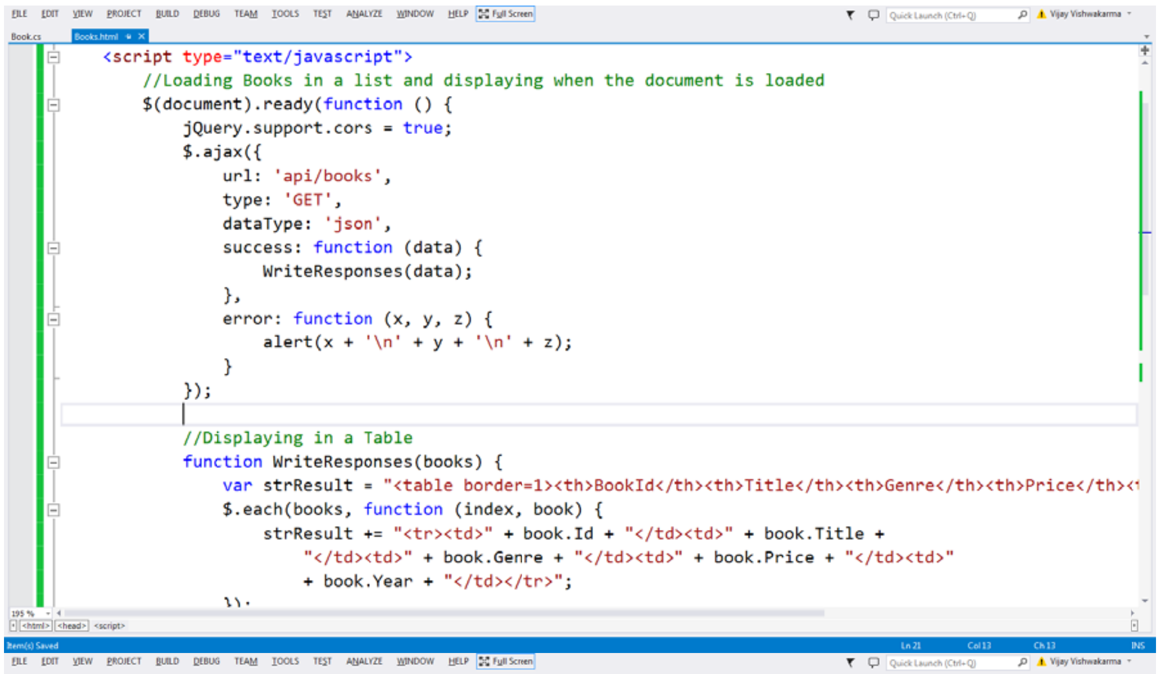
DataTables.net jquery plugin
DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressi...

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

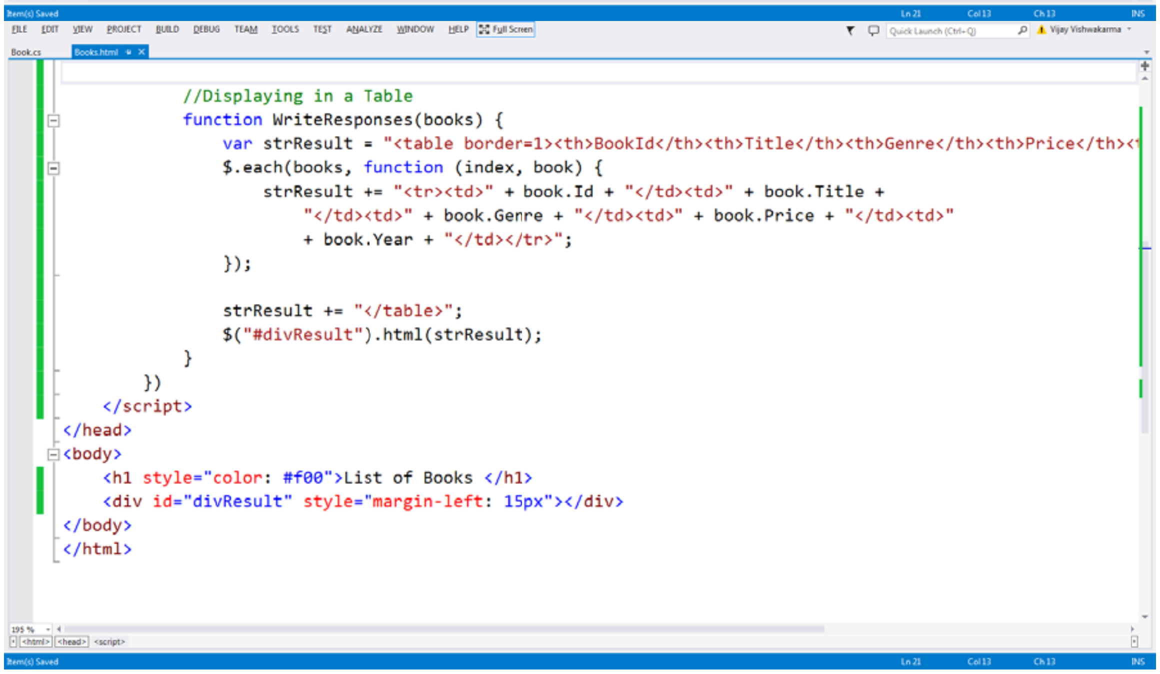
Created by: jQuery Foundation, Inc.
Id: jQuery
Version: 2.2.3
Last Published: 4/12/2016
Downloads: 21604027
License
View License
Project Information
Report Abuse
Description:
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.
NOTE: This package is maintained on behalf of the library owners by the NuGet Community Packages project at http://nugetpackages.codeplex.com/
Tags: jQuery
Dependencies:
No Dependencies

Click on Install which will add jquery reference to the project

Now add a new HTML file to Project name as Books.html and add jquery references to page

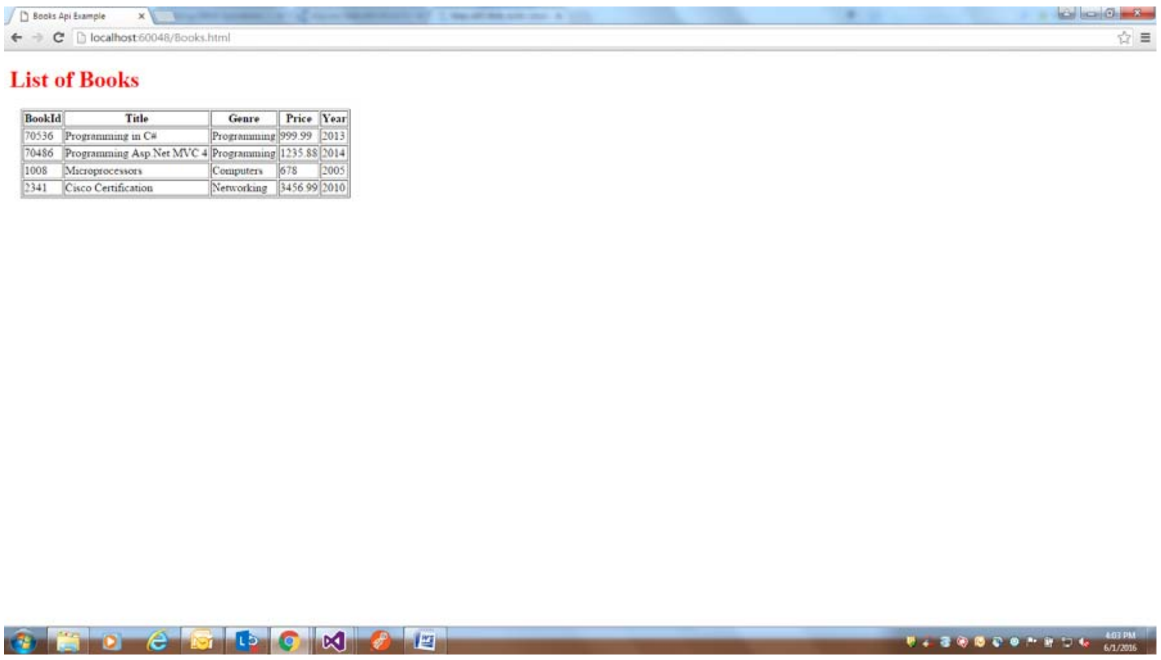


```
    <script type="text/javascript">
        //Loading Books in a list and displaying when the document is loaded
        $(document).ready(function () {
            jQuery.support.cors = true;
            $.ajax({
                url: 'api/books',
                type: 'GET',
                dataType: 'json',
                success: function (data) {
                    WriteResponses(data);
                },
                error: function (x, y, z) {
                    alert(x + '\n' + y + '\n' + z);
                }
            });

            //Displaying in a Table
            function WriteResponses(books) {
                var strResult = "<table border=1><th>BookId</th><th>Title</th><th>Genre</th><th>Price</th><
                $.each(books, function (index, book) {
                    strResult += "<tr><td>" + book.Id + "</td><td>" + book.Title +
                        "</td><td>" + book.Genre + "</td><td>" + book.Price + "</td><td>"
                        + book.Year + "</td></tr>";
```



```
            //Displaying in a Table
            function WriteResponses(books) {
                var strResult = "<table border=1><th>BookId</th><th>Title</th><th>Genre</th><th>Price</th><
                $.each(books, function (index, book) {
                    strResult += "<tr><td>" + book.Id + "</td><td>" + book.Title +
                        "</td><td>" + book.Genre + "</td><td>" + book.Price + "</td><td>"
                        + book.Year + "</td></tr>";
                });

                strResult += "</table>";
                $("#divResult").html(strResult);
            }
        })
    </script>
</head>
<body>
    <h1 style="color: #f00">List of Books </h1>
    <div id="divResult" style="margin-left: 15px"></div>
</body>
</html>
```

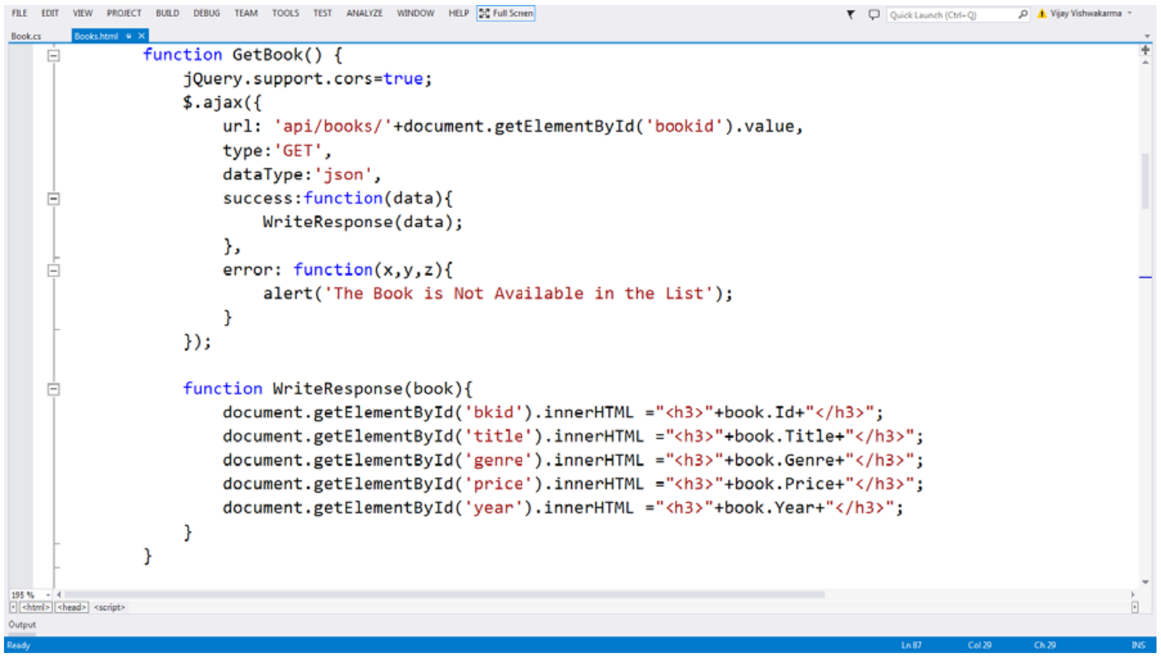
Now run the project and you will the following output



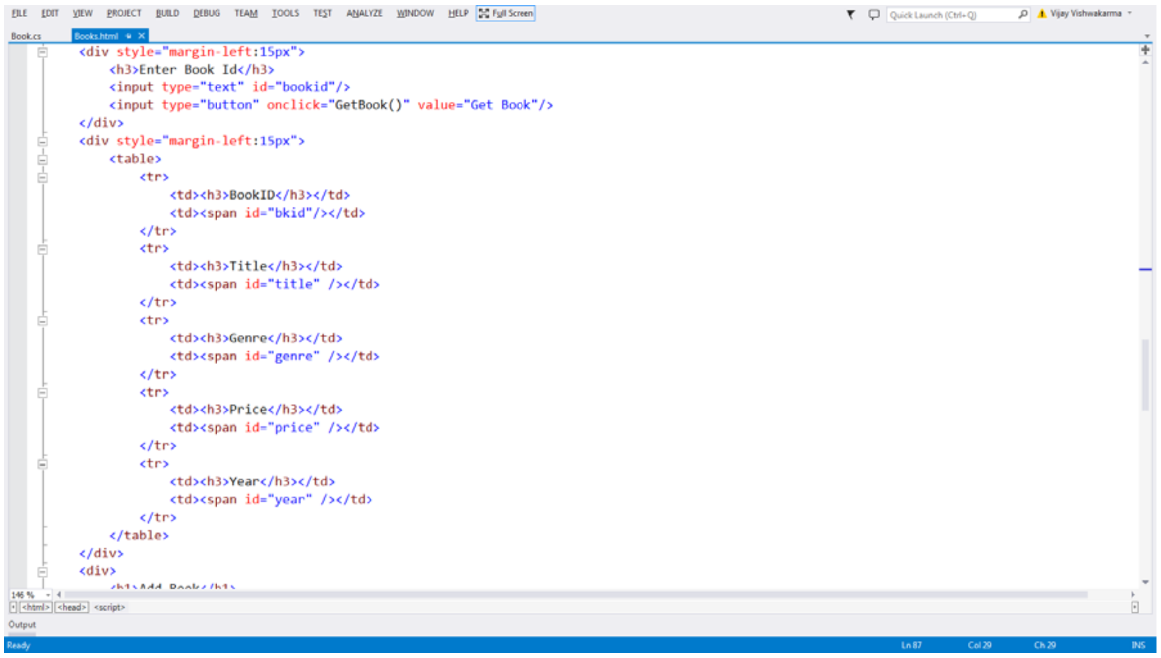
List of Books

BookId	Title	Genre	Price	Year
70536	Programming in C#	Programming	999.99	2013
70486	Programming Asp.Net MVC 4	Programming	1235.88	2014
1008	Microprocessors	Computers	678	2005
2341	Cisco Certification	Networking	3456.99	2010

For Searching the Book add the following code to the Book.html File

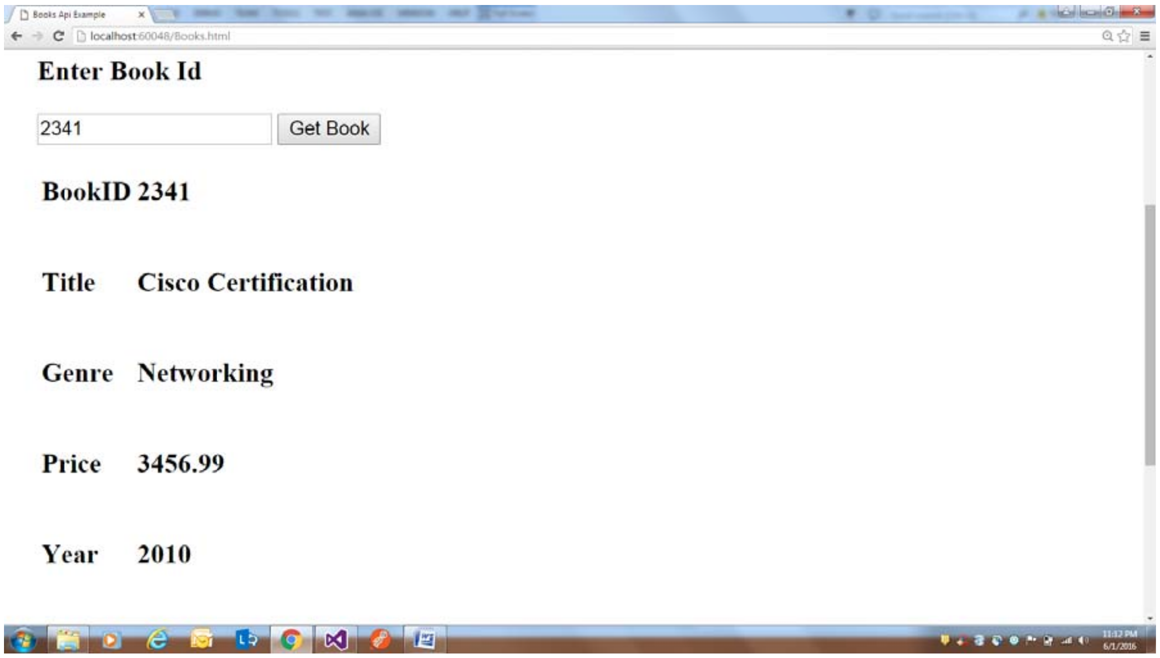


```
        function GetBook() {
            jQuery.support.cors=true;
            $.ajax({
                url: 'api/books/'+document.getElementById('bookid').value,
                type:'GET',
                dataType:'json',
                success:function(data){
                    WriteResponse(data);
                },
                error: function(x,y,z){
                    alert('The Book is Not Available in the List');
                }
            });

            function WriteResponse(book){
                document.getElementById('bkid').innerHTML ="<h3>"+book.Id+"</h3>";
                document.getElementById('title').innerHTML ="<h3>"+book.Title+"</h3>";
                document.getElementById('genre').innerHTML ="<h3>"+book.Genre+"</h3>";
                document.getElementById('price').innerHTML ="<h3>"+book.Price+"</h3>";
                document.getElementById('year').innerHTML ="<h3>"+book.Year+"</h3>";
            }
        }
```



```
    <div style="margin-left:15px">
        <h3>Enter Book Id</h3>
        <input type="text" id="bookid"/>
        <input type="button" onclick="GetBook()" value="Get Book"/>
    </div>
    <div style="margin-left:15px">
        <table>
            <tr>
                <td><h3>BookID</h3></td>
                <td><span id="bkid"/></td>
            </tr>
            <tr>
                <td><h3>Title</h3></td>
                <td><span id="title" /></td>
            </tr>
            <tr>
                <td><h3>Genre</h3></td>
                <td><span id="genre" /></td>
            </tr>
            <tr>
                <td><h3>Price</h3></td>
                <td><span id="price" /></td>
            </tr>
            <tr>
                <td><h3>Year</h3></td>
                <td><span id="year" /></td>
            </tr>
        </table>
    </div>
    <div>
        <h1>Add Book</h1>
```

Now press F5 or click run button in Visual Studio You will the following output



Enter Book Id

2341

BookID 2341

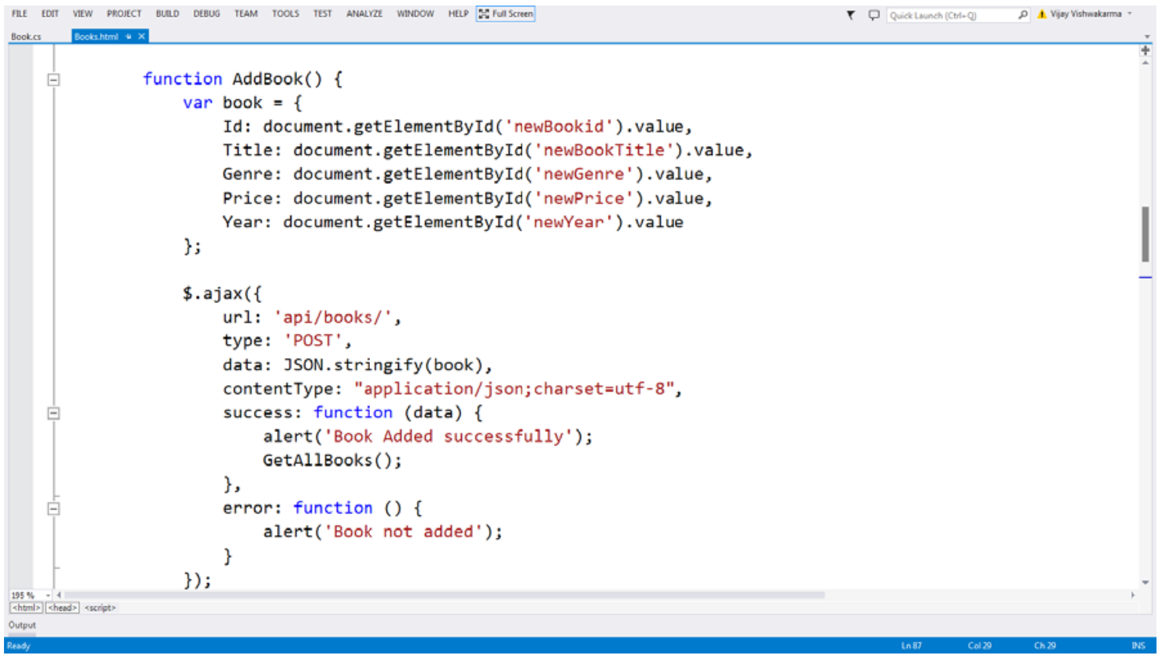
Title Cisco Certification

Genre Networking

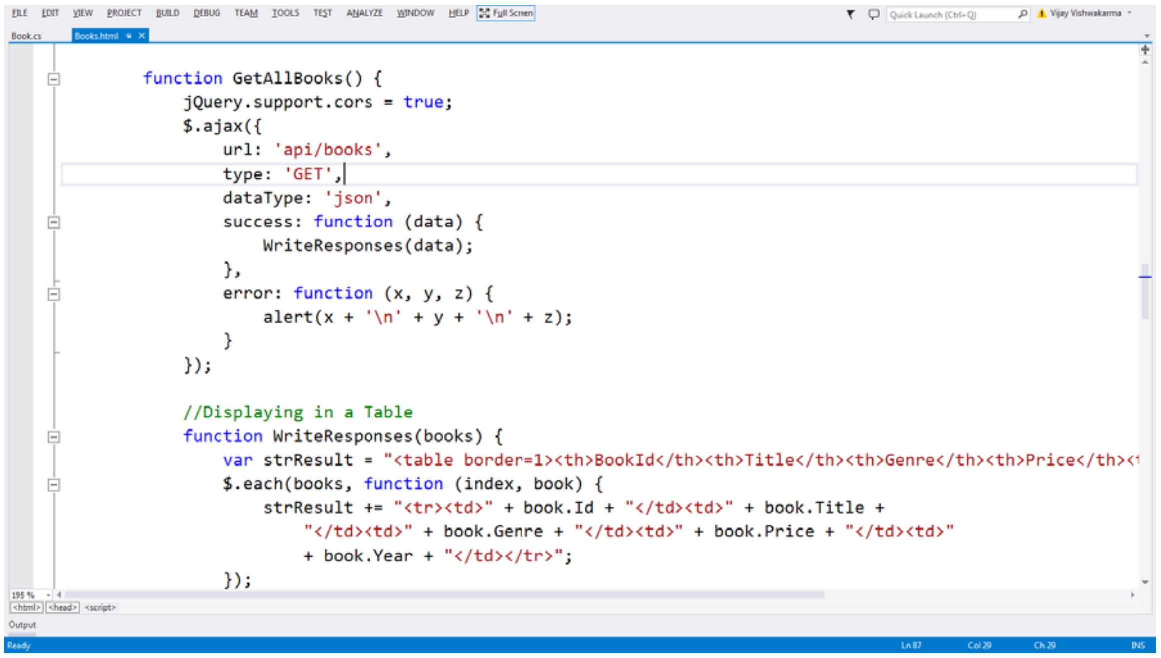
Price 3456.99

Year 2010

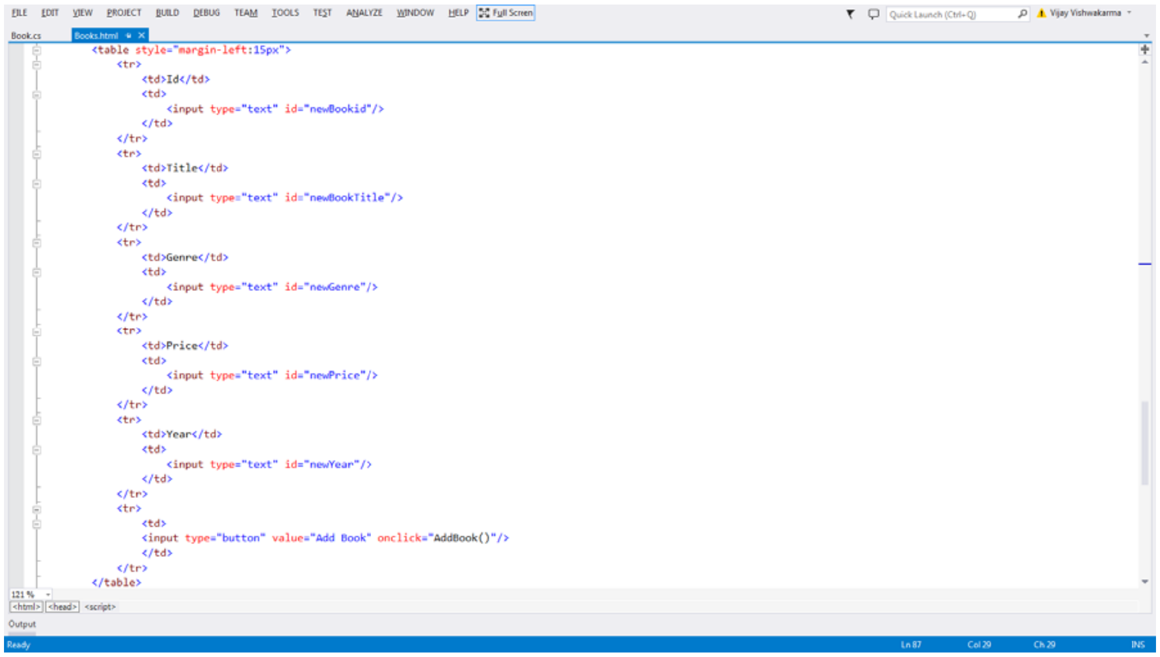
For adding a new book add the following code



```
        function AddBook() {
            var book = {
                Id: document.getElementById('newBookid').value,
                Title: document.getElementById('newBookTitle').value,
                Genre: document.getElementById('newGenre').value,
                Price: document.getElementById('newPrice').value,
                Year: document.getElementById('newYear').value
            };

            $.ajax({
                url: 'api/books/',
                type: 'POST',
                data: JSON.stringify(book),
                contentType: "application/json;charset=utf-8",
                success: function (data) {
                    alert('Book Added successfully');
                    GetAllBooks();
                },
                error: function () {
                    alert('Book not added');
                }
            });
```

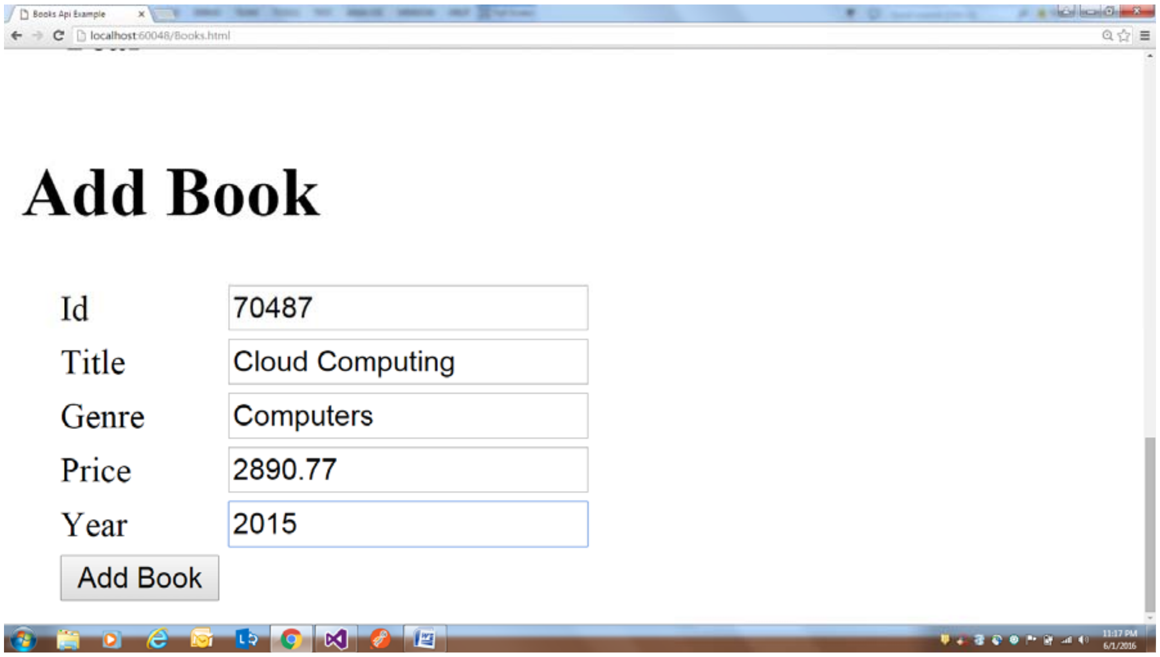


```
        function GetAllBooks() {
            jQuery.support.cors = true;
            $.ajax({
                url: 'api/books',
                type: 'GET',
                dataType: 'json',
                success: function (data) {
                    WriteResponses(data);
                },
                error: function (x, y, z) {
                    alert(x + '\n' + y + '\n' + z);
                }
            });

            //Displaying in a Table
            function WriteResponses(books) {
                var strResult = "<table border=1><th>BookId</th><th>Title</th><th>Genre</th><th>Price</th><
                $.each(books, function (index, book) {
                    strResult += "<tr><td>" + book.Id + "</td><td>" + book.Title +
                        "</td><td>" + book.Genre + "</td><td>" + book.Price + "</td><td>"
                        + book.Year + "</td></tr>";
                });
```



```
<table style="margin-left:15px">
    <tr>
        <td>Id</td>
        <td>
            <input type="text" id="newBookid"/>
        </td>
    </tr>
    <tr>
        <td>Title</td>
        <td>
            <input type="text" id="newBookTitle"/>
        </td>
    </tr>
    <tr>
        <td>Genre</td>
        <td>
            <input type="text" id="newGenre"/>
        </td>
    </tr>
    <tr>
        <td>Price</td>
        <td>
            <input type="text" id="newPrice"/>
        </td>
    </tr>
    <tr>
        <td>Year</td>
        <td>
            <input type="text" id="newYear"/>
        </td>
    </tr>
    <tr>
        <td>
        <input type="button" value="Add Book" onclick="AddBook()"/>
        </td>
    </tr>
</table>
```

Now press F5 or click on run button in Visual studio for running the application



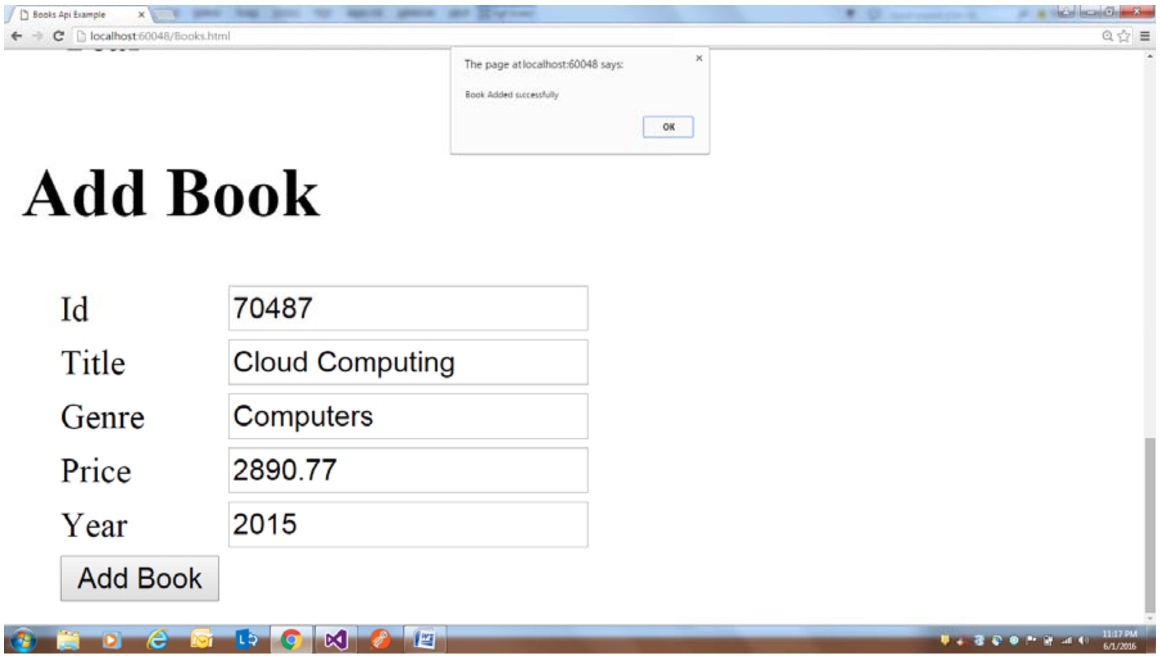
Books Api Example

localhost:60048/Books.html

Add Book

Id	70487
Title	Cloud Computing
Genre	Computers
Price	2890.77
Year	2015

Add Book



Books Api Example

localhost:60048/Books.html

The page at localhost:60048 says:

Book Added successfully

OK

Add Book

Id	70487
Title	Cloud Computing
Genre	Computers
Price	2890.77
Year	2015

Add Book

To do Assignment:-

Create a Web API application to maintain the details of the Employee in an organization. Employee detail will include the following fields

Employee:-

EmployeeID
FirstName
LastName
DOB
Email
Grade
Contact

Web API will perform following action

i) **Returns the detail of all the employee**
ii) **Search Employee based on ID**
iii) **Search Employee based on Grade**
iv) **Add New Employee Details**
v) **Delete Employee Details**

Create a Client Web Application to implement all the action specified in web api