Microsoft

OFFICIAL MICROSOFT LEARNING PRODUCT

20486B

Developing ASP.NET MVC 4 Web Applications

Companion Content

Product Number: 20486B

Part Number : X18-52163

Released: 05/2013

MICROSOFT LICENSE TERMS
OFFICIAL MICROSOFT LEARNING PRODUCTS
MICROSOFT OFFICIAL COURSE Pre-Release and Final Release Versions

---

These license terms are an agreement between Microsoft Corporation and you. Please read them. They apply to the Licensed Content named above, which includes the media on which you received it, if any. These license terms also apply to any updates, supplements, internet based services and support services for the Licensed Content, unless other terms accompany those items. If so, those terms apply.

BY DOWNLOADING OR USING THE LICENSED CONTENT, YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THEM, DO NOT DOWNLOAD OR USE THE LICENSED CONTENT.

If you comply with these license terms, you have the rights below.

---

1. **DEFINITIONS.**

   a. "Authorized Learning Center" means a Microsoft Learning Competency Member, Microsoft IT Academy Program Member, or such other entity as Microsoft may designate from time to time.

   b. "Authorized Training Session" means the Microsoft-authorized instructor-led training class using only MOC Courses that are conducted by a MCT at or through an Authorized Learning Center.

   c. "Classroom Device" means one (1) dedicated, secure computer that you own or control that meets or exceeds the hardware level specified for the particular MOC Course located at your training facilities or primary business location.

   d. "End User" means an individual who is (i) duly enrolled for an Authorized Training Session or Private Training Session, (ii) an employee of a MPN Member, or (iii) a Microsoft full-time employee.

   e. "Licensed Content" means the MOC Course and any other content accompanying this agreement. Licensed Content may include (i) Trainer Content, (ii) software, and (iii) associated media.

   f. "Microsoft Certified Trainer" or "MCT" means an individual who is (i) engaged to teach a training session to End Users on behalf of an Authorized Learning Center or MPN Member, (ii) currently certified as a Microsoft Certified Trainer under the Microsoft Certification Program, and (iii) holds a Microsoft Certification in the technology that is the subject of the training session.

   g. "Microsoft IT Academy Member" means a current, active member of the Microsoft IT Academy Program.

   h. "Microsoft Learning Competency Member" means a Microsoft Partner Network Program Member in good standing that currently holds the Learning Competency status.

   i. "Microsoft Official Course" or "MOC Course" means the Official Microsoft Learning Product instructor-led courseware that educates IT professionals or developers on Microsoft technologies.

j. "Microsoft Partner Network Member" or "MPN Member" means a silver or gold-level Microsoft Partner Network program member in good standing.

k. "Personal Device" means one (1) device, workstation or other digital electronic device that you personally own or control that meets or exceeds the hardware level specified for the particular MOC Course.

l. "Private Training Session" means the instructor-led training classes provided by MPN Members for corporate customers to teach a predefined learning objective. These classes are not advertised or promoted to the general public and class attendance is restricted to individuals employed by or contracted by the corporate customer.

m. "Trainer Content" means the trainer version of the MOC Course and additional content designated solely for trainers to use to teach a training session using a MOC Course. Trainer Content may include Microsoft PowerPoint presentations, instructor notes, lab setup guide, demonstration guides, beta feedback form and trainer preparation guide for the MOC Course. To clarify, Trainer Content does not include virtual hard disks or virtual machines.

2. **INSTALLATION AND USE RIGHTS**. The Licensed Content is licensed not sold. The Licensed Content is licensed on a one copy per user basis, such that you must acquire a license for each individual that accesses or uses the Licensed Content.

2.1 Below are four separate sets of installation and use rights. Only one set of rights apply to you.

a. **If you are a Authorized Learning Center:**

i. If the Licensed Content is in digital format for each license you acquire you may either:

1. install one (1) copy of the Licensed Content in the form provided to you on a dedicated, secure server located on your premises where the Authorized Training Session is held for access and use by one (1) End User attending the Authorized Training Session, or by one (1) MCT teaching the Authorized Training Session, **or**
2. install one (1) copy of the Licensed Content in the form provided to you on one (1) Classroom Device for access and use by one (1) End User attending the Authorized Training Session, or by one (1) MCT teaching the Authorized Training Session.

ii. You agree that:

1. you will acquire a license for each End User and MCT that accesses the Licensed Content,
2. each End User and MCT will be presented with a copy of this agreement and each individual will agree that their use of the Licensed Content will be subject to these license terms prior to their accessing the Licensed Content. Each individual will be required to denote their acceptance of the EULA in a manner that is enforceable under local law prior to their accessing the Licensed Content,
3. for all Authorized Training Sessions, you will only use qualified MCTs who hold the applicable competency to teach the particular MOC Course that is the subject of the training session,
4. you will not alter or remove any copyright or other protective notices contained in the Licensed Content,

5. you will remove and irretrievably delete all Licensed Content from all Classroom Devices and servers at the end of the Authorized Training Session,
6. you will only provide access to the Licensed Content to End Users and MCTs,
7. you will only provide access to the Trainer Content to MCTs, and
8. any Licensed Content installed for use during a training session will be done in accordance with the applicable classroom set-up guide.

b. **If you are a MPN Member.**

i. If the Licensed Content is in digital format for each license you acquire you may either:

1. install one (1) copy of the Licensed Content in the form provided to you on (A) one (1) Classroom Device, or (B) one (1) dedicated, secure server located at your premises where the training session is held for use by one (1) of your employees attending a training session provided by you, or by one (1) MCT that is teaching the training session, **or**
2. install one (1) copy of the Licensed Content in the form provided to you on one (1) Classroom Device for use by one (1) End User attending a Private Training Session, or one (1) MCT that is teaching the Private Training Session.

ii. You agree that:

1. you will acquire a license for each End User and MCT that accesses the Licensed Content,
2. each End User and MCT will be presented with a copy of this agreement and each individual will agree that their use of the Licensed Content will be subject to these license terms prior to their accessing the Licensed Content. Each individual will be required to denote their acceptance of the EULA in a manner that is enforceable under local law prior to their accessing the Licensed Content,
3. for all training sessions, you will only use qualified MCTs who hold the applicable competency to teach the particular MOC Course that is the subject of the training session,
4. you will not alter or remove any copyright or other protective notices contained in the Licensed Content,
5. you will remove and irretrievably delete all Licensed Content from all Classroom Devices and servers at the end of each training session,
6. you will only provide access to the Licensed Content to End Users and MCTs,
7. you will only provide access to the Trainer Content to MCTs, and
8. any Licensed Content installed for use during a training session will be done in accordance with the applicable classroom set-up guide.

c. **If you are an End User:**

You may use the Licensed Content solely for your personal training use. If the Licensed Content is in digital format, for each license you acquire you may (i) install one (1) copy of the Licensed Content in the form provided to you on one (1) Personal Device and install another copy on another Personal Device as a backup copy, which may be used only to reinstall the Licensed Content; or (ii) print one (1) copy of the Licensed Content. You may not install or use a copy of the Licensed Content on a device you do not own or control.

d. **If you are a MCT.**

i. For each license you acquire, you may use the Licensed Content solely to prepare and deliver an Authorized Training Session or Private Training Session. For each license you acquire, you may install and use one (1) copy of the Licensed Content in the form provided to you on one (1) Personal Device and install one (1) additional copy on another Personal Device as a backup copy, which may be used only to reinstall the Licensed Content. You may not install or use a copy of the Licensed Content on a device you do not own or control.

ii. **Use of Instructional Components in Trainer Content.** You may customize, in accordance with the most recent version of the MCT Agreement, those portions of the Trainer Content that are logically associated with instruction of a training session. If you elect to exercise the foregoing rights, you agree: (a) that any of these customizations will only be used for providing a training session, (b) any customizations will comply with the terms and conditions for Modified Training Sessions and Supplemental Materials in the most recent version of the MCT agreement and with this agreement. For clarity, any use of "*customize"* refers only to changing the order of slides and content, and/or not using all the slides or content, it does not mean changing or modifying any slide or content.

2.2 **Separation of Components.** The Licensed Content components are licensed as a single unit and you may not separate the components and install them on different devices.

2.3 **Reproduction/Redistribution Licensed Content**. Except as expressly provided in the applicable installation and use rights above, you may not reproduce or distribute the Licensed Content or any portion thereof (including any permitted modifications) to any third parties without the express written permission of Microsoft.

2.4 **Third Party Programs**. The Licensed Content may contain third party programs or services. These license terms will apply to your use of those third party programs or services, unless other terms accompany those programs and services.

2.5 **Additional Terms**. Some Licensed Content may contain components with additional terms, conditions, and licenses regarding its use. Any non-conflicting terms in those conditions and licenses also apply to that respective component and supplements the terms described in this Agreement.

3. **PRE-RELEASE VERSIONS.** If the Licensed Content is a pre-release ("**beta**") version, in addition to the other provisions in this agreement, then these terms also apply:

a. **Pre-Release Licensed Content.** This Licensed Content is a pre-release version. It may not contain the same information and/or work the way a final version of the Licensed Content will. We may change it for the final version. We also may not release a final version. Microsoft is under no obligation to provide you with any further content, including the final release version of the Licensed Content.

b. **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, either directly or through its third party designee, you give to Microsoft without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software, Microsoft product, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software, technologies, or products to third parties because we include your feedback in them. These rights

survive this agreement.

c. **Term**. If you are an Authorized Training Center, MCT or MPN, you agree to cease using all copies of the beta version of the Licensed Content upon (i) the date which Microsoft informs you is the end date for using the beta version, or (ii) sixty (60) days after the commercial release of the Licensed Content, whichever is earliest ("**beta term**"). Upon expiration or termination of the beta term, you will irretrievably delete and destroy all copies of same in the possession or under your control.

4. **INTERNET-BASED SERVICES**. Microsoft may provide Internet-based services with the Licensed Content, which may change or be canceled at any time.

a. **Consent for Internet-Based Services.** The Licensed Content may connect to computer systems over an Internet-based wireless network. In some cases, you will not receive a separate notice when they connect. Using the Licensed Content operates as your consent to the transmission of standard device information (including but not limited to technical information about your device, system and application software, and peripherals) for internet-based services.

b. **Misuse of Internet-based Services.** You may not use any Internet-based service in any way that could harm it or impair anyone else's use of it. You may not use the service to try to gain unauthorized access to any service, data, account or network by any means.

5. **SCOPE OF LICENSE**. The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allows you to use it in certain ways. Except as expressly permitted in this agreement, you may not:
   - install more copies of the Licensed Content on devices than the number of licenses you acquired;
   - allow more individuals to access the Licensed Content than the number of licenses you acquired;
   - publicly display, or make the Licensed Content available for others to access or use;
   - install, sell, publish, transmit, encumber, pledge, lend, copy, adapt, link to, post, rent, lease or lend, make available or distribute the Licensed Content to any third party, except as expressly permitted by this Agreement.
   - reverse engineer, decompile, remove or otherwise thwart any protections or disassemble the Licensed Content except and only to the extent that applicable law expressly permits, despite this limitation;
   - access or use any Licensed Content for which you are not providing a training session to End Users using the Licensed Content;
   - access or use any Licensed Content that you have not been authorized by Microsoft to access and use; or
   - transfer the Licensed Content, in whole or in part, or assign this agreement to any third party.

6. **RESERVATION OF RIGHTS AND OWNERSHIP**. Microsoft reserves all rights not expressly granted to you in this agreement. The Licensed Content is protected by copyright and other intellectual property laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Licensed Content. You may not remove or obscure any copyright, trademark or patent notices that appear on the Licensed Content or any components thereof, as delivered to you.

7. **EXPORT RESTRICTIONS**. The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, End Users and end use. For additional information, see www.microsoft.com/exporting.

8. **LIMITATIONS ON SALE, RENTAL, ETC. AND CERTAIN ASSIGNMENTS**. You may not sell, rent, lease, lend or sublicense the Licensed Content or any portion thereof, or transfer or assign this agreement.

9. **SUPPORT SERVICES**. Because the Licensed Content is "as is", we may not provide support services for it.

10. **TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of this agreement. Upon any termination of this agreement, you agree to immediately stop all use of and to irretrievable delete and destroy all copies of the Licensed Content in your possession or under your control.

11. **LINKS TO THIRD PARTY SITES**. You may link to third party sites through the use of the Licensed Content. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.

12. **ENTIRE AGREEMENT.** This agreement, and the terms for supplements, updates and support services are the entire agreement for the Licensed Content.

13. **APPLICABLE LAW.**
    a. United States. If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

    b. Outside the United States. If you acquired the Licensed Content in any other country, the laws of that country apply.

14. **LEGAL EFFECT**. This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

15. **DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS," "WITH ALL FAULTS," AND "AS AVAILABLE." YOU BEAR THE RISK OF USING IT. MICROSOFT CORPORATION AND ITS RESPECTIVE AFFILIATES GIVE NO EXPRESS WARRANTIES, GUARANTEES, OR CONDITIONS UNDER OR IN RELATION TO THE LICENSED CONTENT. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT CORPORATION AND ITS RESPECTIVE AFFILIATES EXCLUDE ANY IMPLIED WARRANTIES OR CONDITIONS, INCLUDING THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

16. **LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. TO THE EXTENT NOT PROHIBITED BY LAW, YOU CAN RECOVER FROM MICROSOFT CORPORATION AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO USD$5.00. YOU AGREE NOT TO SEEK TO RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES FROM MICROSOFT CORPORATION AND ITS RESPECTIVE SUPPLIERS.**

This limitation applies to

- anything related to the Licensed Content, services made available through the Licensed Content, or content (including code) on third party Internet sites or third-party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection dues consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 $ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.
Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence , aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers ; et
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Revised December 2011

Module 01

Exploring ASP.NET MVC 4

Contents:

Lesson 3

Introduction to ASP.NET MVC 4

Contents:

Demonstration

Demonstration: How to Explore an MVC Application

Demonstration Steps

1. In the Solution Explorer pane of the **PhotoSharingSample – Microsoft Visual Studio** window, expand **PhotoSharingSample**, and then note that the PhotoSharingSample application does not have the default.htm, the default.aspx, or the default.cshtml files to act as a home page.
2. In the Solution Explorer pane, under PhotoSharingSample, expand **Controllers**, and then click **HomeController.cs**.
3. In the HomeController.cs code window, locate the following code.

```
Public ActionResult Index()
{
   return View();
}
```

Note: This code block represents an action that will return a view called Index.

4. In the Solution Explorer pane, expand **Views**, and then expand **Photo**.
5. In the Solution Explorer pane, under Photo, click **Index.cshtml**.
6. In the Index.cshtml code window, locate the following code.

```
<h2>@ViewBag.Title</h2>
<p>
 @Html.ActionLink("Create New", "Create")
</p>
```

Note: This code block represents the View that renders the home page.

7. On the toolbar of the **PhotoSharingSample – Microsoft Visual Studio** window, click **Internet Explorer**.
8. In the **http://localhost:**<*yourportnumber*>**/** window, note that the default home page is displayed.
9. On the taskbar, click the **Microsoft Visual Studio** icon.
10. In the **PhotoSharingSample – Microsoft Visual Studio** window, in the Solution Explorer pane, expand **App_Start**, and then click **RouteConfig.cs**.
11. In the RouteConfig.cs code window, locate the following code.

```
routes.MapRoute(
   name: "Default",
   url: "{controller}/{action}/{id}",
   )
```

Note: This code block represents the default route that forwards requests to the specified controller.

12. On the taskbar, click the **Internet Explorer** icon.

13. In the Address bar of the Windows Internet Explorer window, type the URL **http://localhost:**<*yourportnumber*>**/home/index**, and then click the **Go to** button.

Note: The browser window displays the Home page of the **http://localhost:**<*yourportnumber*>**/home/index** web application.

14. On the taskbar, click the **Microsoft Visual Studio** icon.
15. In the **PhotoSharingSample – Microsoft Visual Studio** window, in the Solution Explorer pane, expand **Models**, and then click **Photo.cs**.
16. In the Photo.cs code window, locate the following code.

```
 [Required]
public string Title { get; set;}
```

Note: This code block represents the **Title** property for a photo stored in the application.

17. In the Solution Explorer pane, under Controllers, click **PhotoController.cs**.
18. In the PhotoController.cs code window, locate the following code.

```
public class PhotoController : Controller
```

Note: This code block represents that the **PhotoController** class inherits the System.Web.MVC.Controller base class.

19. In thePhotoController.cs code window, locate the following code.

```
public ActionResult Details
    (int id = 0)
{
  Photo photo = db.Photos.Find(id);
  if (photo == null)
  {
    return HttpNotFound();
  }
  return View("Details", photo);
}
```

Note: This code block represents the **Details** action of the Photo Controller.

20. In the Solution Explorer pane, expand **Views**, expand **Photo**, and then click **Details.cshtml**.
21. In the Details.cshtml code window, locate the following code.

```
<h2>"@Model.Title"</h2>
```

Note: The Razor view engine runs this code and renders the Photo Title property that you viewed in the model.

22. On the taskbar, click the **Internet Explorer** icon.
23. In the Address bar of the Windows Internet Explorer window, type **http://localhost:**<*yourportnumber*>**/photo/details/2**, and then click the **Go to** button.

 **Note:** The photo with ID 2 is displayed in the browser window. Note that the title of the photo is rendered at the top.

24. In the Windows Internet Explorer window, click the **Close** button.
25. In the **PhotoSharingSample (Running) – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Best Practice

Use Web Pages when you have simple requirements or have developers with little experience of ASP.NET.

Best Practice

Use Web Forms when you want to create a user interface by dragging controls from a toolbox onto each webpage or when your developers have experience of Web Forms or Windows Forms.

Best Practice

Use MVC when you want the most precise control over HTML and URLs, when you want to cleanly separate business logic, user interface code, and input logic, or when you want to perform Test Driven Development.

Review Question(s)

Question: Which of the shared features of ASP.NET can you use in Web Pages, Web Forms, and MVC applications to increase the speed with which frequently-requested pages are returned to the browser?

Answer: Caching.

Question: You want to create a simple website that shares dates and venues for games for your sports club members. The sports club has no budget to buy software. Which development environment should you use to create the site?

Answer: WebMatrix

Real-world Issues and Scenarios

You have written a web application for a client that sells hats. Visitors to the site will be able to register, redeem offer vouchers, and purchase hats. You expect site traffic to be steady through most of the year, but to peak just before Christmas. Should you recommend IIS or Windows Azure for hosting the site?

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
You add a new view to an MVC application, but when you try to access the page, you receive an HTTP 404 error.	In MVC, views cannot function without controller actions. To use the new view, you must add a controller action that returns that view.

Lab Review Questions and Answers

Lab: Exploring ASP.NET MVC 4

Question and Answers

Question: Which of the three programming models has the simplest method of applying a single layout across multiple pages?

Answer: Opinions will vary.

Question: Which of the three programming models has the simplest method of building a user interface?

Answer: Opinions will vary.

Question: Which of the application programming models will you recommend for the photo sharing application: Web Pages, Web Forms, or MVC?

Answer: Opinions may vary.

Module 02

Designing ASP.NET MVC 4 Web Applications

Contents:

Module Review and Takeaways

Best Practice

In Agile Development and Extreme Programming projects, developers discuss with users and stakeholders throughout development to ensure that their code will meet changing requirements. Even if you are not formally using these methodologies, it is good practice to regularly communicate with users.

Best Practice

When you design an ASP.NET MVC web application, start with the model, and then plan controllers, actions, and views. The controllers, actions, and views that you create each depend on the model.

Review Question(s)

Question: You want to support both English and Spanish in your web application. You have both Spanish-speaking and English-speaking developers and want to ensure that views remain readable as easily as possible. Should you use multiple view files or multiple resource files to globalize your site?

Answer: Multiple view files.

Real-world Issues and Scenarios

You should bear in mind that when you select a project methodology, few projects follow a neat plan in real situations. Of the methodologies described in this module, agile development and extreme programming are the most flexible and respond when plans change in the middle of development. However, even with these methodologies, changing circumstances result in wasted development time and your project budget should include a contingency to cope with such changes.

Furthermore, when working with agile development and extreme programming projects, project managers must take care to avoid project creep or scope-creep. This occurs when people add new requirements when development takes place. Project creep results in projects that are over-budget and late.

Tools

Microsoft Office Visio: You can use Visio to create all types of UML software design diagrams, including Domain Model diagrams and LDMs. You can also use it to create wireframes.

Visual Studio 2012: You can create class diagrams such as LDMs in Visual Studio 2012.

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
When you create a very detailed project plan, much of your work is wasted when requirements change late in the project.	Use Agile Development and Extreme Programming methodologies. Such methodologies, which are based on the real-world assumption that requirements will change frequently, are proven to prevent the time wastage that often occurs when complete designs are altered during the development phase.

Lab Review Questions and Answers

Lab: Designing ASP.NET MVC 4 Web Applications

Question and Answers

Question: What model classes should be created for the photo sharing application based on the initial investigation?

Answer: Answers may vary. The initial investigation implies that the following model classes, or a similar set, should be created:

- Photo
- Comment
- User

Question: What controllers should be created for the photo sharing application based on the initial investigation?

Answer: Answers may vary. Students may design different controllers for the application. However, there is usually one controller for each model class. Bearing in mind that the controller name is conventionally the model class name with "Controller" appended, the following controllers may be appropriate.

The initial investigation implies that the following controllers, or a similar set, should be created:

- PhotoController
- CommentController
- UserController

Question: What views should be created for the photo sharing application?

Answer: Answers may vary.

Module 03

Developing ASP.NET MVC 4 Models

Contents:

Lesson 1

Creating MVC Models

Contents:

Demonstration

Demonstration: How to Add a Model

Demonstration Steps

1. On the **File** menu of the **Start Page - Microsoft Visual Studio** window, point to **New,** and then click **Project.**
2. In the navigation pane of the **New Project** dialog box, expand **Installed**, expand **Templates**, and then expand **Visual C#**.
3. Under Visual C#, click **Web**, and then, in the result pane, click **ASP.NET MVC 4 Web Application**.
4. In the **Name** box of the **New Project** dialog box, type **OperasWebSites**.
5. In the **New Project** dialog box, click **Browse**.
6. In the **Location** text box, navigate to **Allfiles (D):\Democode\Mod03**, and then click **Select Folder**.
7. In the **New Project** dialog box, click **OK**.
8. In the **Select a Template** list of the **New ASP.NET MVC 4 Project** dialog box, click **Empty**, and then click **OK**.
9. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, right-click **Models**, point to **Add**, and then click **Class**.
10. In the **Name** box of the **Add New Item - OperasWebSites** dialog box, type **Opera.cs**, and then click **Add**.
11. In the Opera class of the Opera.cs code window, type the following code.

```
public int OperaID { get; set; }
public string Title { get; set; }
public int Year { get; set; }
public string Composer { get; set; }
```

12. Place the mouse cursor at the end of the **OperaID** property code, press Enter, and then type the following code.

```
[Required]
[StringLength(200)]
```

13. In the Required data annotation, right-click **Required**, point to **Resolve**, and then click **using System.ComponentModel.DataAnnotations**.
14. Place the mouse cursor at the end of the **Year** property, press Enter, and then type the following code.

```
[Required]
```

15. Place the mouse cursor at the end of the Opera class, press Enter, and then type the following code.

```
public class CheckValidYear : ValidationAttribute
{
}
```

16. In the CheckValidYear class, type the following code.

```
public override bool IsValid(object value)
{
   int year = (int)value;
```

```
   if (year < 1598)
   {
      return false;
   }
   else
   {
      return true;
   }
}
```

17. In the CheckValidYear class, type the following code.

```
public CheckValidYear()
{
   ErrorMessage = "The earliest opera is Daphne, 1598, by Corsi, Peri, and
Rinuccini";
}
```

18. In the Opera class, place the mouse cursor at the end of the **Title** property code, press Enter, and then type the following code.

```
[CheckValidYear]
```

19. On the **Build** menu of the **OperasWebSites - Microsoft Visual Studio** window, click **Build Solution**, and then note that the application is being built.

20. In the **OperasWebSites - Microsoft Visual Studio** window, click the **Close** button.

Lesson 2

Working with Data

Contents:

Demonstration

Demonstration: How to Use Entity Framework Code

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, click **web.config**.
2. In the web.config code window, place the mouse cursor at the end of the **</appsettings>** tag, press Enter, and then type the following code.

```
<connectionStrings>
  <add name="OperasDB"
   connectionString=
   "Data Source=(LocalDB)\v11.0;" +
"AttachDbFilename=" +
   "|DataDirectory|\Operas.mdf;" +
   "Integrated Security=True"
   providerName=
    "System.Data.SqlClient" />
</connectionStrings>
```

3. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, right-click **References**, and then click **Manage NuGet Packages**.
4. In the **OperasWebSite - Manage NuGet Packages** window, click **Online**, click **EntityFramework**, and then click **Install**.
5. On the **License Acceptance** page, click **I Accept**.
6. In the **OperasWebSite - Manage NuGet Packages** window, click **Close**.
7. In the **Microsoft Visual Studio** dialog box, click **Yes to All**.
8. In the Solution Explorer pane, right-click **Models**, point to **Add**, and then click **Class**.
9. In the **Name** box of the **Add New Item - OperasWebSite** dialog box, type **OperasDB**, and then click **Add**.
10. In the OperasDB.cs code window, locate the following code.

```
using System.Web;
```

11. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

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

12. In the OperasDB.cs code window, locate the following code.

```
public class OperaDB
```

13. Append the following code to the existing line of code.

```
: DbContext
```

14. In the **OperaDB** class, type the following code.

```
public DbSet<Opera> Operas
   { get; set; }
```

15. In the Solution Explorer pane, right-click **Models**, point to **Add**, and then click **Class**.

16. In the **Name** box of the **Add New Item - OperasWebSite** dialog box, type **OperasInitializer**, and then click **Add**.
17. In the OperasInitializer.cs code window, place the mouse cursor at the end of the System.web namespace code, press Enter, and then type the following code.

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

18. In the OperasInitializer.cs code window, locate the following code.

```
public class OperasInitializer
```

19. Append the following code to the existing line of code.

```
: DropCreateDatabaseAlways
   <OperasDB>
```

20. In the **OperasInitializer** class code block, type the following code, press Spacebar, and then click, **Seed(OperasDB context)**.

```
override
```

21. In the **Seed** method, place the mouse cursor after the call to base.Seed, press Enter, and then type the following code.

```
var operas = new List<Opera>
{
   new Opera {
      Title = "Cosi Fan Tutte",
      Year = 1790,
      Composer = "Mozart"
   }
};
operas.ForEach(s =>
   context.Operas.Add(s));
context.SaveChanges();
```

22. On the **Build** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Build Solution**, and then note that the application is built successfully.
23. In the Solution Explorer pane, right-click **Controllers**, click **Add**, and then click **Controller**.
24. In the **Controller Name** box, type **OperaController**.
25. In the **Template** box, click **MVC controller with read/write actions and views, using Entity Framework**.
26. In the **Model Class** box, click **Opera (OperasWebSite.Models)**.
27. In the **Data context class** box, click **OpersDB (OperasWebSite.Models)**, and then click **Add**.
28. In the Solution Explorer pane, in the **Views/Operas** folder, double-click **Create.cshtml**.
29. In the Create.cshtml code window, locate and delete the following code.

```
@section Scripts {
   @Script.Render("~/bundles/jqueryval")
}
```

30. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual** Studio window, click **Start Debugging**

Note: The Internet Explorer window is displayed with an error message. The error message is expected because the home page view has not been added

31. In the Address bar of the Internet Explorer window, append the existing URL with **opera/index** and then click the **Go to** button.
32. On the Index page, click **Create New**.
33. In the **Title** box of the result page, type **Carmen**, and then, in the **Year** box, type **1475**.
34. In the **Composer** box, type **Bizet**, and then click **Create**.

Note: An error message is displayed by the custom validator.

35. In the **Year** box, type **1875**, and then click **Create**.
36. In the Internet Explorer window, click the **Close** button.
37. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Best Practice

If you have a pre-existing database for a web application, use Entity Framework in the database-first workflow to import and create your model and its classes.

Best Practice

If you want to create a new database for a web application and prefer to draw your model in a Visual Studio designer, use Entity Framework in the model-first workflow to create your model and its classes.

Best Practice

If you want to create a new database for a web application and prefer to write code that describes your model, use Entity Framework in the code-first workflow to create your model and its classes.

Best Practice

If you want to separate business logic from data access logic, create separate model classes and repository classes.

Review Question(s)

Question: At the end of the first iteration of your project, you have a website that displays photos that users upload. However, during development, the database is empty and users must upload several photos to the site so they can test the functionality. Your manager wants you find some way to populate the database whenever it is deployed to the test server. How can you do this?

Answer: Create an Entity Framework Initializer class.

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
The website cannot connect to or create a database.	The connection string contains many parameters and is a common source of errors. Incorrect values in the string may prevent the website from locating the database server or authenticating properly. For full information about building connection strings in ADO.NET, see http://go.microsoft.com/fwlink/?LinkID=293684&clcid=0x409

Lab Review Questions and Answers

Lab: Developing ASP.NET MVC 4 Models

Question and Answers

Question: You are building a site that collects information from customers for their accounts. You want to ensure that customers enter a valid email address in the **Email** property. How would you do this?

> **Answer:** You can use a Regular Expression validation data annotation. This requires a regular expression that matches only valid email addresses. Alternatively, you can create a custom validation data annotation.

Question: You have been asked to create an intranet site that publishes a customer database, created by the sales department, to all employees within your company. How would you create the model with Entity Framework?

> **Answer:** Use Entity Framework in the database-first workflow.

Module 04

Developing ASP.NET MVC 4 Controllers

Contents:

Lesson 1

Writing Controllers and Actions

Contents:

Demonstration

Demonstration: How to Create a Controller

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, right-click **Controllers**, point to **Add**, and then click **Controller**.
2. In the **Controller Name** box of the **Add Controller** dialog box, type **OperaController**.
3. In the **Template** box, click **Empty MVC controller**, and then click **Add**.
4. In the OperaController.cs code window, locate the following code.

```
using System.Web.MVC;
```

5. Place the mouse cursor at the end of the System.Web.MVC namespace, press Enter, and then type the following code.

```
using System.Data.Entity;
using OperasWebSite.Models;
```

6. In the **OperaController** class code block, press Enter, type the following code, and then press Enter.

```
private OperasDB contextDB =
   new OperasDB();
```

7. In the **Index** action code block, select the following code.

```
   return View();
```

8. Replace the selected code with the following code.

```
return View("Index",
   contextDB.Operas.ToList());
```

9. Place the mouse cursor at the end of the **Index** action code block, press Enter, and then type the following code.

```
public ActionResult Details (int id)
{
}
```

10. In the **Details** action code block, type the following code.

```
Opera opera =
   contextDB.Operas.Find(id);
if (opera != null)
{
   return View("Details", opera);
}
else
{
   return HttpNotFound();
}
```

11. Place the mouse cursor at the end of the **Details** action code block, press Enter twice, and then type the following code.

```
public ActionResult Create ()
{
```

```
}
```

12. In the **Create** action code block, type the following code.

```
Opera newOpera = new Opera();
return View("Create", newOpera);
```

13. Place the mouse cursor at the end of the **Create** action code block, press Enter twice, and then type the following code.

```
[HttpPost]
public ActionResult Create
   (Opera newOpera)
{
}
```

14. Place the mouse cursor in the **Create** action code block with the HTTP verb **POST**, and then type the following code.

```
if (ModelState.IsValid)
{
   contextDB.Operas.Add(newOpera);
   contextDB.SaveChanges();
return
      RedirectToAction("Index");
}
else
{
   return View("Create", newOpera);
}
```

15. On the **FILE** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Save Controllers\OperaControllers.cs**.
16. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Note: The message, "Save changes to the following items?" is displayed.

17. In the Microsoft Visual Studio dialog box, note that the message, "Save changes to the following items?" is displayed, and then click Yes.

Module Review and Takeaways

Best Practice

Unless you have a good reason not to, keep to the convention that each controller should be named to match the corresponding model class, with "Controller" appended. For example, for the Photo model class, the controller should be called PhotoController. By maintaining this convention, you create a logically named set of model and controller classes, and can use the default controller factory.

Best Practice

Take great care if you choose to override the built-in AuthorizeAttribute filter because it implements permissions and security for the web application. If you carelessly modify the security infrastructure of the MVC framework, you may introduce vulnerabilities in your application. Instead, use the built-in filter wherever possible.

Review Question(s)

Question: You want to ensure that the **CreatedDate** property of a new **Photo** object is set to today's date when the object is created. Should this value be set in the **Photo** model class constructor method or in the **PhotoController Create** action method?

Answer: The **CreatedDate** property should be set in the **PhotoController Create** action method.

Lab Review Questions and Answers

Lab: Developing ASP.NET MVC 4 Controllers

Question and Answers

Question: What will happen if you click the Edit or Delete links in the Index view in the Lab?

Answer: If you click the Edit or Delete links in the Index view in the lab, a 404 error message will appear.

Question: Why did you use the **ActionName** annotation for the **DeleteConfirmed** action in the **PhotoController** class?

Answer: Two methods in the same .NET Framework class cannot have the same name and signature. When you have an action called **Delete**, you cannot have the method with the same name because the **Delete** action for the GET verb already exists and has the same ID parameter. To resolve this issue, use a different name to the method, which in this case is **DeleteConfirmed**, but use the **ActionName** annotation to ensure it runs when the **Delete** action is called in an HTTP POST.

Question: In the lab, you added two actions with the name, **Create**. Why is it possible to add these actions without using the ActionName annotation?

Answer: The two **Create** methods (one for HTTP GET and one for HTTP POST) have different signatures because the method parameters are different.

Module 05

Developing ASP.NET MVC 4 Views

Contents:

Lesson 2

Using HTML Helpers

Contents:

Demonstration

Demonstration: How to Use HTML Helpers

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, expand **Controllers**, and then click **OperaController.cs**.
2. On the **BUILD** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Build Solution**.
3. In the **OperaController.cs** code window, locate the following code, right-click the code, and then click **Add View**.

```
public ActionResult Create ()
```

4. In the **View Name** box of the **Add View** dialog box, ensure that the name displayed is **Create**.
5. In the **Add View** dialog box, ensure that the **Create a strongly-typed view** check box is selected.
6. In the **Model class** box, ensure that the value is **Opera (OperasWebSite.Models)**. If not, in the **Model class** box, click **Opera (OperasWebSite.Models)**.
7. In the **Scaffold template** box, ensure that the value is **Empty**.
8. In the **Add View** dialog box, ensure that the **Use a layout or master page** check box is not selected, and then click **Add**.
9. In the **DIV** element of the **Create.cshtml** code window, type the following code.

```
<h2>Add an Opera</h2>
```

10. Place the mouse cursor at the end of the **</h2>** tag, press Enter twice, and then type the following code.

```
@using (Html.BeginForm(
   "Create", "Opera",
   FormMethod.Post))
{
}
```

11. In the **using** code block, type the following code.

```
<p>
@Html.LabelFor(model =>
      model.Title):
@Html.EditorFor(model =>
      model.Title)
   @Html.ValidationMessageFor(
      model => model.Title)
</p>
```

12. Place the mouse cursor at the end of the **</p>** tag corresponding to the **model.Title** property, press Enter twice, and then type the following code.

```
<p>
@Html.LabelFor(model =>
      model.Year):
@Html.EditorFor(model =>
      model.Year)
   @Html.ValidationMessageFor(
      model => model.Year)
</p>
```

13. Place the mouse cursor at the end of the **</p>** tag corresponding to the **model.Year** property, press Enter twice, and then type the following code.

```
<p>
   @Html.LabelFor(model =>
      model.Composer):
   @Html.EditorFor(model =>
      model.Composer)
   @Html.ValidationMessageFor(
      model => model.Composer)
</p>
```

14. Place the mouse cursor at the end of the **</p>** tag corresponding to the **model.Composer** property, press Enter twice, and then type the following code.

```
<input type="submit"
   value="Create" />
```

15. Place the mouse cursor at the end of the **<input>** tag, press Enter, and then type the following code.

```
@Html.ActionLink("Back to List", "Index")
```

16. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.

Note: The Operas I Have Seen page is displayed.

17. On the Operas I Have Seen page, click **operas I've seen**.

Note: On the Index page, the list of Operas is displayed.

18. On the Index page, click **Create New**.

Note: The Add an Opera page is displayed.

19. In the **Year** box of the Add an Opera page, type **1597**, and then click **Create**.

Note: Messages corresponding to the **Title**, **Year**, and **Composer** boxes are displayed. The web application mandates you to enter values in all the boxes. Alerts are also displayed for any inappropriate entries, with relevant messages.

20. In the **Title** box of the Add an Opera page, type **Rigoletto**.
21. In the **Year** box of the Add an Opera page, type **1851**.
22. In the **Composer** box of the Add an Opera page, type **Verdi**, and then click **Create**.

Note: The Opera is created with the mentioned values.

23. In the Windows Internet Explorer window, click the **Close** button.

24. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Best Practice

Use Razor comments, declared with the **@* *@** delimiters, throughout your Razor views to help explain the view code to other developers in your team.

Best Practice

Only use **@:** and **<text>** tags when Razor misinterprets content as code. Razor has sophisticated logic for distinguishing content from code, so this is rarely necessary.

Best Practice

Use strongly-typed views wherever possible because Visual Studio helps you to write correct code by displaying IntelliSense feedback.

Review Question(s)

Question: You want to display the name of the **Comment.Subject** property in an MVC view that renders an Edit form for comments. You want the label Edit Subject to appear to the left of a text box so that a user can edit the **Subject** value. Which HTML helper should you use to render the field name?

Answer: Use the **Html.LabelFor()** helper

Question: If your Razor generates errors by wrongly interpreting content as server-side code, this indicates that you have explicitly declared the content by using the **@* *@** delimiters.

( ) True

( ) False

Answer:

( ) True

(√) False

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
When a controller tries to access a partial view, an exception is thrown.	Place partial views in the /Views/Shared folder if they need to be used by various controllers.

Lab Review Questions and Answers

Lab: Developing ASP.NET MVC 4 Views

Question and Answers

Question: How can you improve the accessibility of the HTML that your photo views render?

Answer: Ensure all **<img>** tags include an information **alt** tag.

Question: In the lab, how did you ensure that the **Create** view for **Photo** model objects could upload photo files when the user clicked the **Create** button?

Answer: By passing the **enctype = "multipart/form-data"** parameter as an HTML attribute to the **Html.BeginForm()** helper.

Module 06

Testing and Debugging ASP.NET MVC 4 Web Applications

Contents:

Lesson 1

Unit Testing MVC Components

Contents:

Demonstration

Demonstration: How to Run Unit Tests

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, right-click **Solution 'OperasWebSite' (1 project)**, point to **Add**, and then click **New Project**.
2. In the navigation pane of the **Add New Project** dialog box, under Installed, expand **Visual C#**, and then click **Test**.
3. In the result pane of the **Add New Project** dialog box, click **Unit Test Project**, in the **Name** box, type **OperasWebSiteTests**, and then click **OK**.
4. In the Solution Explorer pane, under OperasWebSiteTests, right-click **References**, and then click **Add Reference**.
5. In the navigation pane of the **Reference Manager - OperasWebSiteTests** dialog box, click **Solution**.
6. In the **Name** column of the result pane, click **OperasWebSite**, select the check box corresponding to OperasWebSite, and then click **OK**.
7. In the Solution Explorer pane, under OperasWebSiteTests, right-click **References**, and then click **Add Reference**.
8. In the navigation pane of the **Reference Manager - OperasWebSiteTests** dialog box, click **Assemblies**, and then click **Extensions**.
9. In the **Name** column of the result pane, click **System.Web.Mvc** with version number **4.0.0.0**, select the corresponding check box, and then click **OK**.
10. In the Solution Explorer pane, under OperasWebSiteTests, right-click **UnitTest1.cs**, and then click **Rename**.
11. In the Solution Explorer pane, replace **UnitTest1** with **HomeControllerTests.cs**, and then press Enter.
12. In the **Microsoft Visual Studio** dialog box, click **Yes**.
13. In the HomeControllerTests.cs code window, locate the following code.

```
public void TestMethod1()
```

14. Replace the code with the following code.

```
public void Test_Index_Return_View()
```

15. Place the mouse cursor at the end of the Microsoft.VisualStudio.TestTools.UnitTesting namespace, press Enter, and then type the following code.

```
using System.Web.Mvc;
using OperasWebSite.Controllers;
using OperasWebSite.Models;
```

16. In the **Test_Index_Return_View** code block, press Enter, and then type the following code.

```
HomeController controller =
   new HomeController();
var result = controller.Index()
   as ViewResult;
Assert.AreEqual("WrongName",
   result.ViewName);
```

Note: This test is created to show the students a failing test.

17. On the **TEST** menu of the **OperasWebSite - Microsoft Visual Studio** window, point to **Run**, and then click **All Tests**.
18. In the Failed Tests (1) section of the Test Explorer pane, note that **Test_Index_Return_View** is listed.
19. In the Test Explorer pane, click **Test_Index_Return_View**.
20. At the lower part of the Test Explorer pane, drag the separator upward, and then view the test results.
21. In the Test Explorer pane, click the **Close** button.
22. In the Solution Explorer pane, under OperasWebSiteTests, click **HomeControllerTests.cs**.
23. In the HomeControllerTests.cs code window, locate the following code.

```
Assert.AreEqual("WrongName",
   result.ViewName);
```

24. Replace the code with the following code.

```
Assert.AreEqual("Index",
   result.ViewName);.
```

25. On the **TEST** menu of the **OperasWebSite - Microsoft Visual Studio** window, point to **Run**, and then click **All Tests**.
26. In the Passed Tests (1) section of the Test Explorer pane, note that **Test_Index_Return_View** is listed.
27. In the Test Explorer pane, click **Test_Index_Return_View**, and then, in the lower part of the Test Explorer pane, view the test results.
28. In the Test Explorer pane, click the **Close** button.
29. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Best Practice

If you are using TDD or Extreme Programming, define each test before you write the code that implements a requirement. Use the test as a full specification that your code must satisfy. This requires a full understanding of the design.

Best Practice

Investigate and choose a mocking framework to help you create test double objects for use in unit tests. Though it may take time to select the best framework and to learn how to code mock objects, your time investment will be worth it over the life of the project.

Best Practice

Do not be tempted to skip unit tests when under time pressure. Doing so can introduce bugs and errors into your system and result in more time being spent debugging.

Review Question(s)

Question: You want to ensure that the PhotoController object passes a single Photo object to the Display view, when a user calls the **Search** action for an existing photo title. What unit tests should you create to check this functionality?

> **Answer:** A unit test should assert only a single fact. Therefore, you should create a unit test that calls the **Search** action and asserts that the **ActionResult** has a **ViewName** of Display. Create a second unit test. In the Arrange phase, add multiple **Photo** objects to test double context, with different titles. Call the **Search** action, passing a title that exists, and assert that the right **Photo** is returned. Create a third unit test, but this time, call the **Search** action with a non-existent title. Assert that **null** is returned.

Tools

NinJect, StructureMap. These are Inversion of Control (IoC) containers, also known as Dependency Injection (DI) frameworks. They create non-test implementations of interfaces in your web application.

Moq, RhinoMocks, NSubstitute. These are mocking frameworks. They automate the creation of test doubles for unit tests.

IntelliTrace. This is a part of Visual Studio that displays application state at the point of an exception or break, and at earlier times.

Health Monitoring. This part of ASP.NET can store health events in a database, log, or other locations for later analysis.

ELMAH. This exception logging tool can store exceptions in database tables, XML files, and elsewhere, and enable administrators to view exception details on a webpage.

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
No information appears in the IntelliTrace window.	The IntelliTrace files store data during a debugging session and are destroyed when Visual Studio exits. If there is no information in the IntelliTrace window, it is often because you have not entered the debugging mode in the current Visual Studio session. If this is not

Common Issue	Troubleshooting Tip
	the problem, check that IntelliTrace is enabled.
A unit test takes a long time to run or returns an error connecting to a database.	When you design unit tests, you should test the smallest and simplest operation possible. For example, you should test a single method. Also create unit tests that do not rely on infrastructure such as databases and network connections. Use test doubles to create in-memory objects that simulate data objects such as Entity Framework context objects. Unit tests should test your code, and not the infrastructure.

Lab Review Questions and Answers

Lab: Testing and Debugging ASP.NET MVC 4 Web Applications

Question and Answers

Question: When you ran the tests for the first time in Exercise 1, why did **Test_Index_Return_View** pass, while **Test_GetImage_Return_Type** and **Test_PhotoGallery_Model_Type** failed?

> **Answer:** The **Test_GetImage_Return_Type** and the **Test_PhotoGallery_Model_Type** tests failed because the test project was not connected to the database. The **Index** action does not call the database, so the **Test_Index_Return_View** test passed.

Question: In Exercise 1, why did all the tests pass during the second run?

> **Answer:** All three tests used a mock repository to test the **PhotoController** actions. The mock repository does not connect to any database, but uses in-memory data. It can test the controller without making a database connection.

Module 07

Structuring ASP.NET MVC 4 Web Applications

Contents:

Lesson 2

Configuring Routes

Contents:

Demonstration

Demonstration: How to Add Routes

Demonstration Steps

1. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
2. On the Operas I Have Seen page, click the **operas I've seen** link.
3. On the Index page, click the **Details** link corresponding to **Cosi Fan Tutte**.
4. In the Address bar of the Windows Internet Explorer window, note that the URL is **http://localhost:**<*portnumber*>**/Opera/Details/1**.

Note: This URL indicates that the controller is **Opera**, the action is **Details**, and the ID is **1**.

5. In the Windows Internet Explorer window, click the **Close** button.
6. In the Solution Explorer pane, expand **OperasWebSite**, expand **Controllers**, and then click **OperaController.cs**.
7. In the OperaController.cs code window, place the mouse cursor at the end of the **Details** action code block, press Enter twice, and then type the following code.

```
public ActionResult DetailsByTitle(string title)
{
}
```

8. In the **DetailsByTitle** action code block, type the following code, and then press Enter.

```
Opera opera = (Opera)(from o in contextDB.Operas
   where o.Title == title
   select o).FirstOrDefault();
```

9. In the **DetailsByTitle** action code block, after the code that you just entered, type the following code.

```
if (opera == null)
{
   return HttpNotFound();
}
return View("Details", opera);
```

10. In the Solution Explorer pane, under OperasWebSite, expand **App_Start**, and then click **RouteConfig.cs**.
11. In the RouteConfig.cs code window, locate the following code.

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

12. Place the mouse cursor at the end of the call to the **IgnoreRoute()** method, press Enter twice, and then type the following code.

```
routes.MapRoute(
   name: "OperaTitleRoute",
   url: "opera/title/{title}",
   defaults: new { controller = "Opera", action = "DetailsByTitle" }
);
```

13. On the **FILE** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Save All**.
14. On the **DEBUG** menu of **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
15. On the Operas I Have Seen page, click the **operas I've seen** link.
16. In the Address bar of the Windows Internet Explorer window, append the existing URL with **/title/rigoletto**, and then click the **Go** button.

Note: The details of the **Rigoletto** opera are displayed.

17. In the Windows Internet Explorer window, click the **Close** button.
18. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Lesson 3

Creating a Navigation Structure

Contents:

Demonstration

Demonstration: How to Build Site Navigation

Demonstration Steps

1. On the **PROJECT** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Manage NuGet Packages**.
2. In the **OperasWebSite - Manage NuGet Packages** dialog box, click **Online**.
3. In the **Search Online (Ctrl+E)** box of the **OperasWebSite - Manage NuGet Packages** dialog box, type **mvcsitemapprovider**, and then click the **Search** button.
4. In the **OperasWebSite - Manage NuGet Packages** dialog box, click **Install** corresponding to **MvcSiteMapProvider**.
5. In the **OperasWebSite - Manage NuGet Packages** dialog box, ensure that the **MvcSiteMapProvider** package is installed, and then click **Close**.
6. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, expand **OperasWebSite**, collapse **App_Start**, collapse **Controllers**, and then collapse **Views**.
7. In the Solution Explorer pane, under Global.asax, click **Mvc.sitemap**.
8. In the Mvc.sitemap code window, locate the following code.

```
<mvcSiteMapNode title="Home" controller="Home" action="Index">
```

9. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

```
<mvcSiteMapNode title="All Operas" controller="Opera" action="Index" key="AllOperas"
/>
```

10. On the **BUILD** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Build Solution**.
11. In the Solution Explorer pane, expand **Views**, expand **Home**, and then click **Index.cshtml**.
12. In the Index.cshtml code window, place the mouse cursor after the ** ** tag, press Enter, and then type the following code.

```
Menu: @Html.MvcSiteMap().Menu(false, false, true)
```

13. Place the mouse cursor at the end of the site map menu code block, press Enter, and then type the following code.

```
Breadcrumb Trail: @Html.MvcSiteMap().SiteMapPath()
```

14. In the Solution Explorer pane, under Views, expand **Opera**, and then click **Index.cshtml**.
15. In the **Index.cshtml** code window, place the mouse cursor at the end of the **<body>** tag, press Enter, and then type the following code.

```
Menu: @Html.MvcSiteMap().Menu(false, false, true)
```

16. Place the mouse cursor at the end of the site map menu code block, press Enter, and then type the following code.

```
Breadcrumb Trail: @Html.MvcSiteMap().SiteMapPath()
```

17. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.

Note: On the Operas I Have Seen page, ensure that a menu is added.

18. On the Operas I Have Seen page, under Menu, click the **All Operas** link.
19. On the Index page, note that the Main Opera List is displayed.

Note: On the Index page, you can also view the menu.

20. In the Breadcrumb Trail section of the Index page, click the **Home** link.

Note: The Operas I Have Seen page is displayed.

21. On the Operas I Have Seen page, under Menu, click the **About** link.

Note: The About page of the web application is displayed.

22. In the Windows Internet Explorer window, click the **Close** button.
23. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Best Practice

The default route is logical. However, it requires knowledge of controllers and actions for users to understand URLs. You can consider creating custom routes that can be understood with information that users already have.

Best Practice

You can use breadcrumb trails and tree view navigation controls to present the current location of a user, in the context of the logical hierarchy of the web application.

Best Practice

You can use unit tests to check routes in the routing table, similar to the manner with which you use tests to check controllers, actions, and model classes. It is easy for a small mistake to completely change the way URLs are handled in your web application. This is because new routes, if poorly coded, can override other routes. Unit tests highlight such bugs as soon as they arise.

Review Question(s)

Question: You have implemented the MVC Site Map Provider in your web application and used it to build menus and breadcrumb trails with which users can navigate the logical hierarchy. MVC automatically takes routes from the MVC Site Map Provider so the same hierarchy is used in URLs.

( ) True

( ) False

Answer:

( ) True

(√) False

Question: You want to ensure that when the user specifies a relative URL in the form, "customer/3546", the request is forwarded to the **DisplayByID()** action in the **CustomerController**. You also want to ensure that when the user specifies a relative URL in the form, "customer/*fullname*", the request is forwarded to the **DisplayByName()** action in the **CustomerController**. What routes should you add?

Answer: routes.MapRoute(

```
  name: "CustomerIDRoute",
  url: "customer/{id}",
  defaults: new { controller = "Customer", action = "DisplayByID" },
  constraints: new { id = "[0-9]+" }
);
routes.MapRoute(
  name: "CustomerNameRoute",
  url: "customer/{name}",
  defaults: new { controller = "Customer", action = "DisplayByName" }
);
```

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
A route never takes effect.	You must ensure that you add routes to the routing table in the correct order. In general, the most specific routes should be added first and the least specific last. If you add a new route after the default route, which matches any URL, it never takes effect.

Lab Review Questions and Answers

Lab: Structuring ASP.NET MVC 4 Web Applications

Question and Answers

Question: In Exercise 1, when you ran the tests for the first time, why did Test_Default_Route_Controller_Only pass when Test_Photo_Route_With_PhotoID and Test_Photo_Title_Route fail?

Answer: The default route already existed when you ran the tests for the first time. No other routes existed.

Question: Why is the constraint necessary in the **PhotoRoute** route?

Answer: The URL for the **PhotoRoute** route is **/photo/{id}**. This matches any request with the path "/photo" and one more level specified. You have to match only URLs in which the second level is an integer.

Module 08

Applying Styles to ASP.NET MVC 4 Web Applications

Contents:

Lesson 2

Applying CSS Styles to an MVC Application

Contents:

Demonstration

Demonstration: How to Apply a Consistent Look and Feel

Demonstration Steps

1. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.

 **Note:** On the Operas I Have Seen page, note that the main heading, the menu list, and the breadcrumb control are displayed.

2. On the Operas I Have Seen page, click the **All Operas** link.

 **Note:** On the localhost page, the main heading, the menu list, and the breadcrumb controls are not displayed.

3. On the localhost page, click the **Details** link corresponding to any image.

 **Note:** On the localhost page, the details of the opera are displayed. The main heading, the menu list, and the breadcrumb controls are not displayed.

4. In the Windows Internet Explorer window, click the **Close** button.
5. In the Solution Explorer pane, expand **OperasWebSite**, and then expand **Views**.
6. In the Solution Explorer pane, under Views, right-click **Shared**, point to **Add**, and then click **View**.
7. In the **View name** box of the **Add View** dialog box, type **SiteTemplate**.
8. In the **View engine** box, ensure that the value is **Razor (CSHTML)**, and then ensure that the **Create a strongly-typed view** check box is cleared.
9. In the **Add View** dialog box, clear the **Use a layout or master page** check box, and then click **Add**.
10. In the **_SiteTemplate.cshtml** code window, locate the following code, select the code, and then press Delete.

```
@{
   Layout = null;
}
```

11. In the **_SiteTemplate.cshtml** code window, locate the following code.

```
<title>_SiteTemplate</title>
```

12. Replace the **TITLE** element with the following code.

```
<title>@ViewBag.Title</title>
```

13. In the Solution Explorer pane, under Views, expand **Home**, and then click **Index.cshtml**.
14. In the Index.cshtml code window, locate the following code, and then select the code.

```
<h1>Operas I Have Seen</h1>
<div class="topmenu">
```

```
   @Html.MvcSiteMap().Menu(false, true, true)
</div>
<div class="clear-floats" />
<div class="breadcrumb">
  Breadcrumb Trail: @Html.MvcSiteMap().SiteMapPath()
```

15. On the **EDIT** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Cut**.
16. In the Solution Explorer pane, under Shared, click **_SiteTemplate.cshtml**.
17. In the _SiteTemplate.cshtml code window, place the mouse cursor in the **DIV** element.
18. On the **EDIT** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Paste**
19. In the _SiteTemplate.cshtml code window, place the mouse cursor at the end of the code you just pasted, press Enter, and then type the following code.

```
<div>
   @RenderBody()
</div>
```

20. Place the mouse cursor after the **</title>** tag, press Enter, and then type the following code.

```
<link type="text/css" rel="stylesheet" href="~/content/OperasStyles.css" />
```

21. In the Solution Explorer pane, under Home, click **Index.cshtml**.
22. In the Razor code block of the Index.cshtml code window, locate the following code, select the code, and then press Delete.

```
Layout = null;
```

23. In the Razor code block, type the following code.

```
ViewBag.Title = "Operas I Have Seen";
```

24. In the Index.cshtml code window, locate the following code, select the code, and then press Delete.

```
<!DOCTYPE html>
<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>Operas I Have Seen</title>
</head>
<body>
   <div>
   </div>
```

25. In the Index.cshtml code window, locate the following code, select the code, and then press Delete.

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

26. In the Solution Explorer pane, right-click **Views**, point to **Add**, and then click **View**.
27. In the **View name** box of the **Add View** dialog box, type **_ViewStart**, and then, in the View engine box, ensure that the value is **Razor (CSHTML)**.
28. In the **Add View** dialog box, ensure that the **Create a strongly-typed view** and the **Use a layout or master page** check boxes are cleared, and then click **Add**.

29. In the _ViewStart.cshtml code window, locate the following code.

```
Layout = null;
```

30. Replace the code with the following code.

```
Layout = "~/Views/Shared/_SiteTemplate.cshtml";
```

31. In the _ViewStart.cshtml code window, locate the following code.

```
<!DOCTYPE html>
```

32. In the _ViewStart.cshtml code window, select from the code located to the end tag of the HTML element, and then press Delete
33. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
34. On the Operas I Have Seen page, note the main heading, the menu list, and the breadcrumb control.
35. On the Operas I Have Seen page, click the **All Operas** link, and then, on the Index of Operas page, note that the main heading, the menu list, and the breadcrumb controls are displayed.
36. On the Index of Operas page, click the **Details** link corresponding to any image, and then note that the main heading, the menu list, and the breadcrumb controls are displayed along with the opera details.
37. In the Windows Internet Explorer window, click the **Close** button.
38. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Real-world Issues and Scenarios

When you develop web applications, you need to create applications that work on different devices and browsers, such as iPhone, iPad, Windows Phone, Google Chrome, and Internet Explorer 10. In such cases, you can use the HTML5 elements and features in MVC 4, such as mobile-specific views, media queries, and jQuery Mobile library, to create applications that work well in various browsers and devices.

Review Question(s)

Question: You are building an application, which needs to work in different mobile devices, Windows Phone, Windows, and Mac. You want to reduce the effort for maintaining the code which is required for different devices and you want to ensure that it would work with new browsers. What should you do?

Answer: You could use HTML style with media query to create a single style.

Lab Review Questions and Answers

Lab: Applying Styles to MVC 4 Web Applications

Question and Answers

Question: When you first browsed the web application in Exercise 1, why was the menu and the breadcrumb trail visible on the home page, but not on the All Photos page or any other page?

> **Answer:** The menu and breadcrumb helpers were called in the **Views/Home/Index.cshtml** view, but not in the other views.

Question: When you first viewed the site as a mobile browser in Exercise 3, what are the problems you came across with the display of the site heading and menu?

> **Answer:** The site heading was wider than 480 pixels, so it was not visible without scrolling right. The menu items were fixed at a width of 200 pixels, so they were too wide for the page and displayed outside the top menu division. They also disturbed the flow of later elements.

Module 09

Building Responsive Pages in ASP.NET MVC 4 Web Applications

Contents:

Lesson 2

Implementing a Caching Strategy

Contents:

Demonstration

Demonstration: How to Configure Caching

Demonstration Steps

1. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
2. On the Operas I Have Seen page, click the **Tools** button, and then click **F12 developer tools**.
3. On the **Cache** menu of the developer window, click **Always refresh from server**.
4. On the **Network** tab of the developer window, click **Start Capturing**.
5. On the Operas I Have Seen page, click the **All Operas** link.
6. When the page is fully loaded, in the developer window, click **Stop Capturing**.
7. In the URL section of the developer window, click **http://localhost:**<*portnumber*>**/Opera**, and then click **Go to detailed view**.
8. On the **Timings** tab, click the **Request** entry.
9. In the **Duration** column, note the value displayed.
10. On the **Network** tab, click **Clear**, and then click **Start capturing**.
11. On the Operas I Have Seen page, click the **All Operas** link.
12. When the page is fully loaded, in the developer window, click **Stop capturing**.
13. In the URL section of the developer window, click **http://localhost:**<*portnumber*>**/Opera**, and then click **Go to detailed view**.
14. On the **Timings** tab, click the **Request** entry.
15. In the **Duration** column, note the value displayed.

Note: The time taken by the server to render the **/Opera** page and return the page to the browser is similar to the time taken by the server in the first instance. The page is not cached.

16. In the Windows Internet Explorer window, click the **Close** button.
17. In the Solution Explorer pane of the **OperasWebSite – Microsoft Visual Studio** window, under OperasWebSite, expand **Controllers**, and then click **OperaController.cs**.
18. In the OperaController.cs code window, locate the following code.

```
using System.Web.Mvc;
```

19. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

```
using System.Web.UI;
```

20. In the OperaController.cs code window, locate the following code.

```
public ActionResult Index()
```

21. Place the mouse cursor immediately before the located code, press Enter, and then type the following code.

```
[OutputCache(Duration=600, Location=OutputCacheLocation.Server, VaryByParam="none")]
```

22. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
23. On the **Cache** menu of the developer window, click **Always refresh from server**.
24. On the **Network** tab, click **Start capturing**.
25. On the Operas I Have Seen page, click the **All Operas** link.
26. When the page is fully loaded, in the developer window, click **Stop capturing**.
27. In the URL section of the developer window, click **http://localhost:**<*portnumber*>**/Opera**, and then click **Go to detailed view**.
28. On the **Timings** tab, click the **Request** entry.
29. In the **Duration** column, note the value displayed.
30. On the **Network** tab, click **Clear**, and then click **Start capturing**.
31. On the Operas I Have Seen page, click the **All Operas** link.
32. When the page is fully loaded, in the developer window, click **Stop capturing**.
33. In the URL section of the developer window, click **http://localhost:**<*portnumber*>**/Opera**, and then click **Go to detailed view**.
34. On the **Timings** tab, click the **Request** entry.
35. In the **Duration** column, note the value displayed.

 Note: Note that the time taken by the server to render the **/Opera** page and return the page to the browser is significantly less than the time taken by the server in the first instance.

36. On the **File** menu of the developer window, click **Exit**.
37. In the Windows Internet Explorer window, click the **Close** button.
38. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Real-world Issues and Scenarios

Web applications usually run multiple queries to retrieve information from a database and render content on the webpages. Users sometimes complain that webpages take longer to load. Therefore, developers implement caching in the web application, to reduce the need to load data from a database, every time a user places a request. Caching helps webpages load faster, thereby increasing the performance of the application.

Review Question(s)

Question: An application is refreshing the content every 10 seconds for the updated information from database. User complaints that this is impacting their work and has caused data loss. How would you propose to help resolve this issue?

> **Answer:** You can consider rewriting the code to use AJAX and partial update to allow automatic updation of the webpage information without reloading the webpage.

Lab Review Questions and Answers

Lab: Building Responsive Pages in ASP.NET MVC 4 Web Applications

Question and Answers

Question: In Exercise 2, why was the Request timing for /Photo not reduced for the first request when you configured the output cache for the index action?

> **Answer:** When you make the first request to a page after an application restart, there is no data in the output cache and the page is rendered afresh.

Question: In Exercise 2, when you configured the output cache for the GetImage() action, why was it necessary to set VaryByParam="id"?

> **Answer:** It was necessary to set VaryByParam="id" to configure the output cache for the GetImage() action because the GetImage() action renders a different image depending on the value of the id parameter.

Module 10

Using JavaScript and jQuery for Responsive MVC 4 Web Applications

Contents:

Lesson 1

Rendering and Executing JavaScript Code

Contents:

Demonstration

Demonstration: How to Use NuGet to Add a JavaScript Library

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, expand **OperasWebSite**.

Note: There is no folder named Scripts at the top level of the project.

2. In the Solution Explorer pane, expand **Content**.

Note: The Content folder has only one file named **OperaStyles.css**, and there are no sub-folders.

3. On the **PROJECT** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Manage NuGet Packages**.
4. In the navigation pane of the **OperasWebSite – Manage NuGet Packages** dialog box, ensure **Online** is selected, and then click **All**.
5. In the result pane of the **OperasWebSite – Manage NuGet Packages** dialog box, click **jQuery UI(Combined Library)**, click **Install**, and then, when the installation is complete, click **Close**.
6. In the Solution Explorer pane, expand **Scripts**.

Note: NuGet Package Manager has added five files for jquery and jqueryUI to the application. Note the version number for jquery and jqueryUI.

7. In the Solution Explorer pane, under Contents, expand **themes**, expand **base**, and then click **jquery-ui.css**.

Note: NuGet Package Manager has added style sheets to the Content folder. These styles are used to set the styles for jQueryUI widgets, and the most important of these style sheets is **jquery-ui.css**.

8. In the Solution Explorer pane, collapse **base**, expand **Views**, and then expand **Shared**.
9. In the Solution Explorer pane, under Shared, click **_SiteTemplate.cshtml**.
10. In the _SiteTemplate.cshtml code window, locate the following code.

```
</head>
```

11. Place the mouse cursor before the located code, type the following code, and then press Enter.

```
<script type="text/javascript" src="@Url.Content("~/Scripts/jquery-ui-
1.10.0.js")"></scripts>
```

Note: In the above code, note that the version number provided is 1.10.0. When you type the code, replace the version number 1.10.0 with the latest version number.

12. In the _SiteTemplate.cshtml code window, locate the following code.

```
<title>@ViewBag.Title</title>
```

13. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

```
<link type="text/css" rel="stylesheet"
href="@Url.Content("~/Content/themes/base/jquery-ui.css")" />
```

Note: You can now use jQueryUI calls on any views in the application.

14. On the **FILE** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Save All**.
15. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Lesson 2

Using jQuery and jQueryUI

Contents:

Demonstration

Demonstration: How to Add a jQueryUI Widget

Demonstration Steps

1. On the **DEBUG** menu of the **OperasWebsite – Microsoft Visual Studio** window, click **Start Debugging**.
2. On the Operas I Have Seen page, click the **All Operas** link.
3. In the Main Opera List section, click the **Details** link corresponding to **Cosi Fan Tutte**.
4. Under Reviews, note that there are three opera reviews displayed for **Cosi Fan Tutte**, simultaneously.
5. In the Windows Internet Explorer window, click the **Close** button.
6. In the Solution Explorer pane of the **OperasWebsite – Microsoft Visual Studio** window, under Shared, click **_ReviewsForOpera.cshtml**.
7. In the _ReviewsForOpera.cshtml code window, locate the following code.

```
<h3>Reviews</h3>
```

8. Place the mouse cursor immediately before the located code, type the following code, and then press Enter.

```
<script>
$(function() {
        $("#reviews-tool").accordion();
    });
</script>
```

9. On the **DEBUG** menu of the **OperasWebsite – Microsoft Visual Studio** window, click **Start Debugging**.
10. On the Operas I Have Seen page, click the **All Operas** link.
11. In the Main Opera List section, click the **Details** link corresponding to **Cosi Fan Tutte**.
12. Under **Reviews**, note that there are three expandable sections, and each section contains a review.

Note: You can expand each section and then read the review content.

13. In the Windows Internet Explorer window, click the **Close** button.
14. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Review Question(s)

Question: You are building an application that needs to update various parts of the page every 10 seconds. Your team is proposing to use IFRAME. But you want to reduce the number of pages to be created. What type of technology should you propose to achieve this?

Answer: You can consider using jQuery to select the element from the HTML code and update it using asynchronous functions.

Lab Review Questions and Answers

Lab: Using JavaScript and jQuery for Responsive MVC 4 Web Applications

Question and Answers

Question: What is the use of adding the two links to the **_MainLayout.cshtml** file in Task 1 of Exercise 2?

Answer: The links added to the **_MainLayout.cshtml** file serve the following purposes:

- The **<script>** tag linked the view to the jQueryUI JavaScript library.
- The **<link>** tag linked the view to the jQueryUIstylesheet.

Question: You added **<script>** tags to the **_MainTemplate.cshtml** file to enable jQueryUI. Is this the optimal location for this link?

Answer: No, it is not ideal to add **<script>** tags to the **_MainTemplate.cshtml** file to enable jQueryUI. It would be better to place this link in the **SlideShow.cshtml** view because jQueryUI is only used in that view.

Module 11

Controlling Access to ASP.NET MVC 4 Web Applications

Contents:

Lesson 1

Implementing Authentication and Authorization

Contents:

Demonstration

Demonstration: How to Authorize Access to Controller Actions

Demonstration Steps

1. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
2. On the Operas I Have Seen page, click **All Operas**.
3. On the Index of Operas page, click the **Create New** link.

 **Note:** The Create an Opera page is displayed without logging on to the application. This enables anonymous users to create new operas.

4. In the Windows Internet Explorer window, click the **Close** button.
5. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, expand **OperasWebSite**, expand **Controllers**, and then click **OperaController.cs**.
6. In the OperaController.cs code window, locate the following code.

```
public ActionResult Create()
```

7. Place the mouse cursor before the located code, type the following code, and then press Enter.

```
[Authorize]
```

8. In the OperaController.cs code window, locate the following code.

```
[HttpPost]
public ActionResult Create(Opera newOpera)
```

9. Place the mouse cursor before the located code, type the following code, and then press Enter.

```
 [Authorize]
```

10. On the **FILE** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Save All**.
11. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
12. On the Operas I Have Seen page, click **All Operas**.
13. On the Index of Operas page, click the **Create New** link.

 **Note:** The **Login** view is now displayed and this prevents anonymous users from creating new operas.

14. On the Index of Operas page, click the **Register** link.
15. In the **User name** box of the Register page, type **David Johnson**.
16. In the **Password** box, type **Pa$$w0rd**, in the **Confirm password** box, type **Pa$$w0rd**, and then click **Register**.
17. On the Operas I Have Seen page, click **All Operas**.

18. On the Index of Operas page, click the **Create New** link.

Note: The Add an Opera page is displayed because the request is authenticated.

19. In the Windows Internet Explorer window, click the **Close** button.
20. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Lesson 2

Assigning Roles and Membership

Contents:

Demonstration

Demonstration: How to Reset a Password

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite - Microsoft Visual Studio** window, under **Controllers**, click **AccountController.cs**.
2. In the AccountController.cs code window, locate the following comment.

```
//Add Reset Password Code Here
```

3. In the AccountController.cs code window, replace the located comment with the following code, and then press Enter.

```
try
{
}
catch (Exception)
{
}
```

4. In the **try** code block, type the following code.

```
changePasswordSucceeded = Membership.Provider.ChangePassword(User.Identity.Name,
model.OldPassword, model.NewPassword);
```

5. In the **catch** code block, type the following code.

```
changePasswordSucceeded = false;
```

6. On the **FILE** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Save All**.
7. On the **DEBUG** menu of the **OperasWebSite - Microsoft Visual Studio** window, click **Start Debugging**.
8. On the Operas I Have Seen page, click the **Log In** link.
9. On the Logon page, in the **User name** box, type **David Johnson**, in the **Password** box, type **Pa$$w0rd**, and then click **Log in**.
10. On the Operas I Have Seen page, click the **Reset Password** link.
11. On the ResetPassword page, in the **Current password** box, type **Pa$$w0rd**, and then in the **New password** box, type **Pa$$w0rd2**.
12. In the **Confirm new password** box, type **Pa$$w0rd2**, and then click **Change Password**.

Note: On the ResetPassword page, the message, "Your password has been changed." is displayed.

13. In the Windows Internet Explorer window, click the **Close** button.
14. In the OperasWebSite – Microsoft Visual Studio window, click the **Close** button.
15. If the Microsoft Visual Studio warning message appears, click **Yes**.

Module Review and Takeaways

Real-world Issues and Scenarios

When you create web applications, you may need to create custom providers because you do not want to use the schema provided by Microsoft. However, you can use **SimpleProviders** to remove the need to develop custom providers and reduce the effort required for building applications.

Review Question(s)

Question: What is the key difference between implementing authentication and authorization in ASP.NET 4.5 from implementing the same in the previous versions of ASP.NET?

> **Answer:** The key difference is the inclusion of universal providers and simple providers, which allow developers more flexibility in terms of the database schema used to store membership and role information for the application.

Lab Review Questions and Answers

Lab: Controlling Access to ASP.NET MVC 4 Web Applications

Question and Answers

Question: In Exercise 3, when you tried to add a photo before logging on to the application, why did ASP.NET display the **Login** view?

Answer: ASP.NET displayed the **Login** view because you tried to access an action that was restricted to authenticated users.

Question: How can you ensure that only Adventure Works employees are granted access to the **Delete** action of the **Photo** controller?

Answer: Configure an ASP.NET role provider such as the **DefaultRoleProvider** in the ASP.NET Universal Providers package. Use the **ASP.NET Configuration** site to create a role for Adventure Works employees. Use the same site to add employee accounts to the role. Add the **[Authorize(Roles = "Employees")]** annotation to the **Delete** action method.

Module 12

Building a Resilient ASP.NET MVC 4 Web Application

Contents:

Lesson 2

State Management

Contents:

Demonstration

Demonstration: How to Store and Retrieve State Information

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite – Microsoft Visual Studio** window, under **OperasWebSite**, expand **Controllers**, and then click **HomeController.cs**.
2. In the HomeController.cs code window, locate the following code.

```
public ActionResult About()
{
   return View();
}
```

3. Place the mouse cursor at the end of the located code, press Enter twice, and then type the following code.

```
public ContentResult GetBackground()
{
}
```

4. Place the mouse cursor within the **GetBackground** action code block, and then type the following code.

```
string style;
if (Session["BackgroundColor"] != null)
{
}
else
{
}
```

5. Place the mouse cursor within the **if** statement code block you just added, and then type the following code.

```
style = String.Format("background-color: {0};", Session["BackgroundColor"])
```

6. Place the mouse cursor within the **else** statement code block you just added, and then type the following code.

```
style = "background-color: #dc9797;";
```

7. Place the mouse cursor at the end of the **GetBackground** action code block and outside the **if…else** statements, press Enter, and then type the following code.

```
return Content(style);
```

8. Place the mouse cursor outside any action code block but inside the **HomeController** class, and then type the following code.

```
public ActionResult SetBackground(string color)
{
}
```

9. Place the cursor within the **SetBackground** action code block, and then type the following code.

```
Session["BackgroundColor"] = color;
return View("Index");
```

10. In the Solution Explorer pane, expand **Views**, expand **Home**, and then click **Index.cshtml**.
11. In the Index.cshtml code window, place the mouse cursor at the end of the **P** element, press Enter twice, and then type the following code.

```
<p>
   Choose a background color:
</p>
```

12. Place the mouse cursor at the end of the text in the **P** element, press Enter, and then type the following code.

```
@Html.ActionLink("Pink", "SetBackground", new { color = "#dc9797"})
```

13. Place the mouse cursor at the end of the link you just created, press Enter, and then type the following code.

```
@Html.ActionLink("Blue", "SetBackground", new { color = "#82bbf2"})
```

14. In the Solution Explorer pane, expand **Shared**, and then click **_SiteTemplate.cshtml**.
15. In the _SiteTemplate.cshtml code window, locate the following code.

```
<body>
```

16. Replace the located code with the following code.

```
<body style="@Html.Action("GetBackground", "Home")">
```

17. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
18. On the Operas I Have Seen page, click the **Blue** link, and then note that the blue background color has been applied to the home page.
19. On the Operas I Have Seen page, click the **All Operas** link.

Note: If the background color of the page is blue, click the **Refresh** button.

20. On the Index of Operas page, click the **Details** link corresponding to the title, **Cosi Fan Tutte**.
21. On the Cosi Fan Tutte page, note that the blue background color has been applied to the page.

Note: The blue background preference is applied to all pages of the application.

22. On the Opera: Cosi Fan Tutte page, click **Home**.
23. On the Operas I Have Seen page, click the **Pink** link, and then note that the pink background color has been applied to the home page.

Note: If the background color of the page is blue, click the **Refresh** button.

24. On the Operas I Have Seen page, click **All Operas**.
25. On the Index of Operas page, click the **Details** link corresponding to the title, **Cosi Fan Tutte**.
26. On the Cosi Fan Tutte page, note that the pink background color has been applied to the page.

Note: The pink background preference is applied to all pages of the application.

27. In the Windows Internet Explorer window, click the **Close** button.
28. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

29. If the message **Do you want to stop debugging?** is displayed, click **Yes**.

Module Review and Takeaways

Review Question(s)

Question: Recently, an error occurred in one of applications that you had developed for your company. After performing few tests, you realize that the issue was due to an HTML code that was passed from the user to the server. You want to prevent such issues in the future. What would you consider to do?

> **Answer:** You can enable the request validation to ensure that only valid content are accepted by the ASP pages.

Real-world Issues and Scenarios

While implementing web applications, you may want to use a rich format input editor, to enable users to format the input within text boxes. Therefore, you may need to disable request validation, to enable ASP.NET to capture and process user input.

Complex business functions usually involve multiple views. Such functions can pose problems because information must be shared across multiple views. Session state management helps resolve these problems, because it enables retaining information pertinent to multiple views.

Lab Review Questions and Answers

Lab: Building a Resilient ASP.NET MVC 4 Web Application

Question and Answers

Question: In this lab, you stored the list of favorite photos in the session state. While testing, your manager notices that authenticated users lose their favorite photos list whenever they close their browser. Where would you store a list of favorites for each authenticated user so that the list is preserved whenever a user logs on to the web application?

> **Answer:** If you store the list in user profiles, it is persisted over many user sessions and users do not lose the list when they log off from the application.

Question: How would you create a view of favorite photos with the card-style presentation users see on the **All Photos** page?

> **Answer:** You must create a new action in the **Photo** controller. This action can create a list of the user's favorite **Photo** objects as the **FavoritesSlideShow** action does. Pass this list to the **_PhotoGallery** partial view so that it displays only favorite photos.

Module 13

Using Windows Azure Web Services in ASP.NET MVC 4 Web Applications

Contents:

Lesson 3

Consuming Windows Azure Services in a Web Application

Contents:

Demonstration

Demonstration: How to Call a Windows Azure Service by Using jQuery

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebSite – Microsoft Visual Studio** window, expand **OperasWebSite**, expand **Views**, expand **Home**, and then double-click **Index.cshtml**.
2. In the Index.cshtml code window, locate the following code.

```
@Html.ActionLink("operas I've seen.", "Index", "Opera")
</p>
```

3. Place the mouse cursor after the located code, press Enter twice, and then type the following code.

```
<form>
</form>
```

4. Place the mouse cursor in the **FORM** element code block you just created, and then type the following code.

```
<input type="button" value="Get Latest Quote" name="GetLatestQuote"
onclick="callWebService();" />
```

5. Place the mouse cursor at the end of the **INPUT** element, press Enter, and then type the following code.

```
<p id="quote-display"></p>
```

6. In the Index.cshtml code window, locate the following code.

```
</form>
```

7. Place the mouse cursor at the end of the located code, press Enter twice, and then type the following code.

```
<script type="text/javascript">
</script>
```

8. Place the mouse cursor in the **SCRIPT** element code block you just created, and then type the following code.

```
function callWebService() {
}
```

9. Place the mouse cursor in the **callWebService** function code block, and then type the following code.

```
var serviceUrl = '@Url.Content("~/WebServices/QuotesService.asmx")';
```

10. In the **callWebService** function code block, place the mouse cursor at the end of the variable you just created, press Enter, and then type the following code.

```
$.ajax({
   type: "POST",
   url: serviceUrl + "/LatestQuote",
   data: {},
   contentType: "application/json; charset=utf-8",
   dataType: "json",
   success: OnSuccess,
```

```
   error: OnError
});
```

11. Place the mouse cursor at the end of the **callWebService** function code block, but within the **SCRIPT** element, press Enter twice, and then type the following code.

```
function OnSuccess(response) {
}
```

12. Place the mouse cursor in the **OnSuccess** function code block, and then type the following code.

```
$('#quote-display').html(response.d);
```

Note: response.d is the property you use to access JSON data from the server.

13. Place the mouse cursor at the end of the **OnSuccess** function code block, but within the **SCRIPT** element, press Enter twice, and then type the following code.

```
function OnError(response) {
}
```

14. Place the mouse cursor within the **OnError** function, and then type the following code.

```
$('#quote-display').html("Could not obtain the latest quote");
```

15. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
16. On the Operas I Have Seen page, click **Get Latest Quote**.

Note: jQuery calls the web service and displays a quote on the home page. Note that you need not reload the page to display the quote.

17. In the Windows Internet Explorer window, click the **Close** button.
18. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Review Question(s)

Question: Your teammate enquired whether you would be using Windows Azure or self host the server for the application. What should you recommend to your teammate based on the fact that the application loading would be seasonal?

Answer: Windows Azure

Lab Review Questions and Answers

Lab: Using Windows Azure Web Services in ASP.NET MVC 4 Web Applications

Question and Answers

Question: What is the advantage of calling the Bing Maps Geocoding service from a WCF service in Windows Azure, instead of calling the Geocoding service directly from the Create action in the MVC web application?

Answer: You can encapsulate Geocoding functionality with other widely-used functions. Because these functions are available from Windows Azure, they automatically have high-availability and can be rapidly scaled in response to high demand.

Question: Why is latitude and longitude data useful for photos in the Photo Sharing application?

Answer: You can use latitude and longitude data to plot photos on a map such as a Bing Map.

Module 14

Implementing Web APIs in ASP.NET MVC 4 Web Applications

Contents:

Lesson 1

Developing a Web API

Contents:

Demonstration

Demonstration: How to Explore a Web API by Using Internet Explorer

Demonstration Steps

1. In the Solution Explorer pane, expand **OperasWebSite**.
2. In the Solution Explorer pane, under OperasWebSite, right-click **Controllers**, point to **Add**, and then click **Controller**.
3. In the **Controller name** box of the **Add Controller** dialog box, type **OperasApiController**, in the **Template** box, click **Empty API Controller**, and then click **Add**
4. In the OperasApiController.cs code window, locate the following code.

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

5. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

```
using OperasWebSite.Models;
```

6. Place the mouse cursor in the **OperasApiController** class code block, press Enter, and then type the following code.

```
private OperasDB contextDB = new OperasDB();
```

7. Place the mouse cursor at the end of the code you just typed, press Enter twice, and then type the following code.

```
public IEnumerable<Opera> GetOperas()
{
}
```

8. Place the mouse cursor in the **GetOperas** action code block, and then type the following code.

```
return contextDB.Operas.AsEnumerable();
```

9. Place the mouse cursor at the end of the **GetOperas** action code block, press Enter twice, and then type the following code.

```
public Opera GetOperas(int id)
{
}
```

10. Place the mouse cursor in the **GetOperas** action code block you just created, and then type the following code.

```
Opera opera = contextDB.Operas.Find(id);
```

11. Place the mouse cursor at the end of the code you just entered, press Enter, and then type the following code.

```
if (opera == null)
{
   throw new HttpResponseException(HttpStatusCode.NotFound);
}
```

12. Place the mouse cursor at the end of the code you just entered, press Enter, and then type the following code.

```
return opera;
```

13. On the **FILE** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Save All**.
14. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
15. In the Address bar of the Windows Internet Explorer window, type **http://localhost:<***yourPortNumber***>/api/OperasApi**, and then click **Go to**.
16. In the Navigation bar, click **Open**.
17. If the "How do you want to open this type of file (.json)?" message is displayed, click **More options**, and then click **Microsoft Visual Studio Version Selector**.
18. On the **EIDT** menu of the **OperasApi.json – Microsoft Visual Studio** window, point to **Find and Replace**, and then click **Quick Find**.
19. In the **Search Item** box of the **Quick Find** dialog box, type **Rigoletto**, and then click **Find Next**.
20. In the **Microsoft Visual Studio** dialog box, click **OK**.
21. In the Quick Find dialog box, click the **Close** button.

 **Note:** Visual Studio finds the JSON data for the **Rigoletto** opera. Note that this is just one entry in the JSON data, which includes all operas in the web application.

22. In the **OperasApi.json – Microsoft Visual Studio** window, click the **Close** button
23. In the Address bar of the Windows Internet Explorer window, type **http://localhost:<***yourPortNumber***>/api/OperasApi/3**, and then click **Go to**.
24. In the Navigation bar, click **Open**.
25. If the "How do you want to open this type of file (.json)?" message is displayed, click **More options**, and then click **Microsoft Visual Studio Version Selector**.
26. In the **3.json - Microsoft Visual Studio** window, note that only the information relating to the **Nixon in China** opera is displayed.

 **Note:** The value for the **OperasID** parameter corresponding to the **Nixon in China** opera is **3**.

27. In the **3.json - Microsoft Visual Studio** window, click the **Close** button.
28. In the Windows Internet Explorer window, click the **Close** button.
29. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Real-world Issues and Scenarios

Consider that you develop a mobile application by using Web APIs and the application needs to use currency rate services. For this application, you cannot use WCF, because WCF can impede the performance of the application by using XML for data exchanges. Therefore, you should use REST and JSON in the application to reduce the data that is transmitted between the client system and the server.

Review Question(s)

Question: We are developing a mobile application, which requires to access business data via internet. You are proposing to use Web API but your colleague is proposing to use WCF. What would be the key point for using Web API in this scenario?

Answer: The data exchanged via Web API is less than WCF and more effective.

Lab Review Questions and Answers

Question and Answers

Question: How do the API actions you added to the **PhotoApiController** controller in Exercise 1 differ from other actions in MVC controllers?

Answer: Most MVC controller actions return an **ActionResult** object or an object that derives from **ActionResult**. API actions, by contrast, can return any object.

Module 15

Handling Requests in ASP.NET MVC 4 Web Applications

Contents:

Lesson 2

Using Web Sockets

Contents:

Demonstration

Demonstration: How to Add a Chat Room to a Web Application by using SignalR

Demonstration Steps

1. In the Solution Explorer pane of the **OperasWebsite – Microsoft Visual Studio** window, right-click **OperasWebSite**, point to **Add**, and then click **Class**.
2. In the **Name** box of the **Add New Item – OperasWebSite** dialog box, type **ChatHub**, and then click **Add**.
3. In the ChatHub.cs code window, locate the following code.

```
using System.Web;
```

4. Place the mouse cursor at the end of the located code, press Enter, and then type the following code.

```
using Microsoft.AspNet.SignalR;
```

5. In the ChatHub.cs code window, locate the following code.

```
public class ChatHub
```

6. Replace the located code with the following code.

```
public class ChatHub : Hub
```

7. In the **ChatHub** class code block, type the following code.

```
public void Send(string name, string message)
{
}
```

8. In the **Send** method code block, type the following code.

```
Clients.All.broadcastMessage(name, message);
```

Note: The **Send()** method sends any message received, back to all the clients that are connected to the hub. You need to define the **broadcastMessage()** method in the client-side code to receive messages. The client-side code must also call the **Send()** method to broadcast messages.

9. In the Solution Explorer pane, expand **Views,** expand **Home**, and then click **Chat.cshtml**.
10. In the Chat.cshtml code window, within the final **<script>** element, type the following code.

```
$(function() {
});
```

11. Within the anonymous function you just created, type the following code.

```
var chat = $.connection.chatHub;
```

12. Place the mouse cursor at the end of the variable you just created, press Enter, and then type the following code.

```
chat.client.broadcastMessage = function (name, message) {
};
```

Note: This function is the implementation of the **broadcastMessage()** function that you called in

the Hub code.

13. Within the anonymous function you just created, type the following code.

```
var listItem = '<li>' + name + ': ' + message + '</li>';
```

14. Place the mouse cursor at the end of the variable you just created, press Enter, and then type the following code.

```
$('#discussion').append(listItem);
```

15. Place the mouse cursor at the end of the **broadcastMessage** function code block, press Enter, and then type the following code.

```
var displayname = prompt('Enter your name:', '');
```

16. Place the mouse cursor at the end of the **displayname** variable code block you just created, press Enter, and then type the following code.

```
$('#chat-message').focus();
```

17. Place the mouse cursor at the end of the code block you just created, press Enter, and then type the following code.

```
$.connection.hub.start().done(function () {
});
```

18. Within the anonymous function code block you just created, type the following code.

```
$('#sendmessage').click(function () {
});
```

19. Within the new anonymous function code block you just created, type the following code.

```
chat.server.send(displayname, $('#chat-message').val());
```

20. Place the mouse cursor at the end of the code block you just created, press Enter, and then type the following code.

```
$('#chat-message').val('').focus();
```

21. On the **DEBUG** menu of the **OperasWebSite – Microsoft Visual Studio** window, click **Start Debugging**.
22. On the Operas I Have Seen page, click the **Enter the Operas Chat Room** link.
23. In the **Enter your name** box of the **localhost needs some information** dialog box, type **Rebecca Laszlo**, and then click **OK**.
24. In the **Message** box of the Operas I Have Seen page, type a message of your choice, and then click **Send**.

Note: SignalR sends the message you typed to the hub. The hub broadcasts the message to all connected clients.

25. On the taskbar, right-click the **Internet Explorer** icon, and then click **Internet Explorer**.
26. In the Address bar of the Internet Explorer window, type **http://localhost:<***portnumber***>**, and then press Enter.

27. On the Operas I Have Seen page, click the **Enter the Operas Chat Room** link.
28. In the **Enter your name** box of the **localhost needs some information** dialog box, type **Elisa Graceffo**, and then click **OK**.
29. In the **Message** box of the Operas I Have Seen page, type a message of your choice, and then click **Send**.
30. On the taskbar, click the first instance of the Internet Explorer window. Note that the message from **Elisa Graceffo** is displayed because both users are connected to the same hub.
31. Close all the Internet Explorer windows.
32. In the **OperasWebSite – Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Real-world Issues and Scenarios

You create an application that obtains the latest product pricing information from the internal database. The pricing is constantly updated every time business users feel the need for updates. Therefore, you need to ensure that the application updates pricing information every five minutes. In such cases, you can to use the web socket technology to implement price update. You can also add code to download the product image stored in the product database by using a generic handler (*.ashx file).

Review Question(s)

Question: You want to select technology that could be used for developing a web based chatting applications for your client. Which technology would you prefer in this scenario?

Answer: Web socket

Lab Review Questions and Answers

Lab: Handling Requests in ASP.NET MVC 4 Web Applications

Question and Answers

Question: In the chat functionality that you created, each photo in the Photo Sharing application has a separate chat room. How is this separation possible with one SignalR hub?

Answer: SignalR Groups are used to separate the chat rooms.

Question: In Exercise 2, you wrote JScript code that called the **chat.server.join()** and **chat.server.send()** functions. In which script file are these functions defined?

Answer: In the automatically generated **~/signalr/hubs** script file.

Module 16

Deploying ASP.NET MVC 4 Web Applications

Contents:

Lesson 1

Deploying a Web Application

Contents:

Demonstration

Demonstration: How to Create a Windows Azure Website

Demonstration Steps

1. On the taskbar, click the **Internet Explorer** icon.
2. In the Address bar of the Internet Explorer window, type **http://www.windowsazure.com**, and then click the **Go to** button.
3. In the upper-right corner of the Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services page, click **PORTAL**.
4. On the Sign in to your Microsoft account page, in the **Microsoft account** box, type <*your username*>, in the **Password** box, type **Pa$$w0rd**, and then click **Sign in**.
5. In the left pane of the Windows Azure page, click **WEB SITES**.
6. In the lower-left pane of the Windows Azure page, click **NEW**, and then click **CUSTOM CREATE**.
7. In the **URL** box of the Create Web Site page, type <*your username*>**operas**, and then, in the **REGION** box, click <*A region near you*>.
8. In the **DATABASE** box of the Create Web Site page, click **Create a new SQL database**, in the **DB CONNECTION STRING NAME** box, type **OperasDB**, and then click the **Next** button.
9. In the **NAME** box of the Specify database settings page, type **OperasDB**, in the **SERVER** box, click **New SQL database server**, and then, in the **LOGIN NAME** box, type <*your first name*>.
10. In the **PASSWORD** box of the Specify database settings page, type **Pa$$w0rd**, in the **PASSWORD CONFIRMATION** box, type **Pa$$w0rd**, and then click the **Complete** button.

 **Note:** Windows Azure creates the new website and database to support the Operas website.

11. In the Internet Explorer window, click the **Close** button.

Lesson 2

Deploying an ASP.NET MVC 4 Web Application

Contents:

Demonstration

Demonstration: How to Deploy a Website to Windows Azure

Demonstration Steps

1. On the taskbar, click the **Internet Explorer** icon.
2. In the Address bar of the Internet Explorer window, type **http://www.windowsazure.com**, and then click the **Go to** button.
3. In the upper-right corner of the Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services page, click **PORTAL**.
4. On the Sign in to your Microsoft account page, in the **Microsoft account** box, type <*your username*>, in the **Password** box, type **Pa$$w0rd**, and then click **Sign in**.
5. In the left pane of the Windows Azure page, click **WEB SITES**.
6. In the **NAME** column of the Web Sites – Windows Azure page, click <*your username*>**operas**.
7. On the Web Sites – Windows Azure page, click **DASHBOARD**.

Note: If the quick start page appears, repeat the steps 5 and 6 before proceeding to the next step.

8. In the quick glance section of the Web Sites – Windows Azure page, click the **Download publish profile** link.
9. In the Navigation bar, click **Save**, and then click the **Close** button.
10. On the taskbar, click the **Microsoft Visual Studio** icon.
11. In the Solution Explorer pane of the OperasWebSite - Microsoft Visual Studio window, right-click **OperasWebSite**, and then click **Publish**.
12. On the **Profile** page of the Publish Web wizard, click **Import**.
13. In the **Import Publish Settings** dialog box, click <*your username*>**operas.azurewebsites.net.PublishSettings**, and then click **Open**.
14. On the **Connection** page of the Publish Web wizard, click **Validate Connection**.
15. If the **Certificate Error** dialog box appears, click **Accept**.
16. On the **Connection** page, click **Next**.
17. On the **Settings** page, click **Next**.
18. On the **Preview** page, click **Start Preview**.
19. On the **Preview** page, click **Publish**.

Note: Visual Studio publishes the website. This process can take several minutes. When the publish operation is complete, the website is displayed in the Internet Explorer window.

20. If the **Certificate Error** dialog box appears, click **Accept**.
21. In the Internet Explorer window, if the **Server Error in '/' Application** error is displayed, click the **Refresh** button.

Note: If the Operas I Have Seen page does not appear, you need to re-publish the **OperasWebSite** project.

22. On the Operas I Have Seen page, click **All Operas**.

23. In the Navigation bar, if the message **Intranet settings are turned off by default.** is displayed, click **Turn on Intranet settings**.
24. If the message **Are you sure you want to turn on intranet-level security settings?** appears, click **Yes**.
25. On the Index of Operas page, click the **Details** link corresponding to **Cosi Fan Tutte**.
26. On the Opera: Cosi Fan Tutte page, click the **Back to List** link.
27. On the Index of Operas page, click the **Details** link corresponding to **Nixon in China**.
28. In the Internet Explorer window, click the **Close** button.
29. In the **OperasWebSite - Microsoft Visual Studio** window, click the **Close** button.

Module Review and Takeaways

Review Question(s)

Question: You need to create two packages for deployment-for testing and for production environment. This is because of the difference in server name and other environment settings. What should you do?

Answer: You should consider having different Web.config files configured in your project and you need to use the publish feature to create the deployment package in different environment.

Lab Review Questions and Answers

Lab: Deploying ASP.NET MVC 4 Web Applications

Question and Answers

Question: Why is it unnecessary to use bin deployment in this lab?

Answer: It is unnecessary to use bin deployment because all the pre-requisites for the Photo Sharing web application are already in place on Windows Azure servers.

Question: In the labs for this course, you used the same Windows Azure SQL Database for both development and production. If you wanted to use separate databases for development and production, but did not want to reconfigure the web application every time you deployed to the development and production web servers, how would you configure the web application?

Answer: Use separate Web.config variance files for development and production deployments.