Module07

Structuring ASP.NET MVC 4 Web Applications

Module Overview

- Analyzing Information Architecture
- Configuring Routes
- Creating a Navigation Structure

Lesson 1: Analyzing Information Architecture

- What Is Information Architecture?
- What Is Search Engine Optimization?

What Is Information Architecture?

- Planning a Logical Hierarchy.
- Presenting a Hierarchy in Navigation Controls
- Presenting a Hierarchy in URLs

MVC Model:

- **Boiler**
- **Category**
- **FAQQuestion**
- **Installation Manual**
- **User Manual**

$$\Downarrow$$

Information Architecture:

- **Category**
  - **Furnace**
    - **FAQQuestion**
    - **Installation Manual**
    - **User Manual**

What Is Search Engine Optimization?

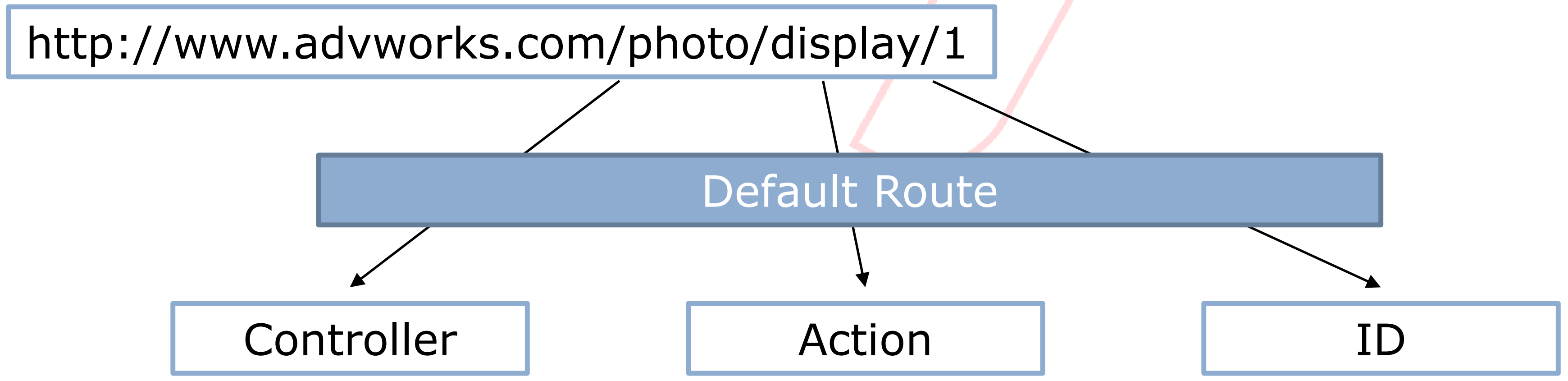
- Use meaningful <title> elements
- Use accurate <meta name="keyword"> tags
- Use accurate <meta name="description"> tags
- Use different <title> <meta> elements on each page
- Choose a domain name that includes keywords
- Use keywords in heading elements
- Ensure that navigation controls enable web bots to crawl your entire web application
- Ensure that URLs do not include GUIDs or long query text

Lesson 2: Configuring Routes

- The ASP.NET Routing Engine
- Adding and Configuring Routes
- Using Routes to Pass Parameters
- Demonstration: How to Add Routes
- Unit Tests and Routes

The ASP.NET Routing Engine

- The default route:



- Custom routes:
  - To make URLs easier for site visitors to understand
  - To improve search engine rankings
- Controller factories and routes

Adding and Configuring Routes

- Understand the properties of a route:
  - Includes Name, URL, Constraints and Defaults
- Analyze the default route code:
  - Specifies **Name**, **URL** ,and **Defaults** properties
- Create Custom Routes:
  - Involves calling the **routes.MapHttpRoute()** method
- Understand the precedence of routes:
  - Add routes to the **RouteTable.Routes** collection in the appropriate order

```
routes.MapRoute(
   name: "Default",
   url: "{controller}/{action}/{id}",
   defaults: new {
      controller = "Home",
      action = "Index",
      id = UrlParameter.Optional }
);
```

```
routes.MapRoute(
   name: "PhotoRoute",
   url: "photo/{id}",
   defaults: new {
      controller = "Photo",
      action = "Details" },
   constraints: new {
      id = "[0-9]+" }
);
```

Using Routes to Pass Parameters

- You can access the values of these variables by:
  - Using the **RouteData.Values** collection
  - Using the model binding to pass appropriate parameters to actions

```
public void ActionMethod Display (int PhotoID)
{
   return View(PhotoID);
}
```

- You can use optional parameters to match a route, regardless of whether parameter values are supplied

```
routes.MapRoute(
   name: "ProductRoute",
   url: "product/{id}/{color}",
   defaults: new { color = UrlParameter.Optional }
)
```

Demonstration: How to Add Routes

In this demonstration, you will see how to:

1. Use the default route in the MVC web application
2. Add a new action method that uses an existing view
3. Add a new route for the new action method, which enables users to enter an opera title in the URL
4. Browse an opera by entering the title of the opera

Unit Tests and Routes

A Unit Test for the Routing Table:

```
[TestMethod]
public void Test_Default_Route_ControllerOnly()
{
   //Arrange
   var context = new FakeHttpContextForRouting(
      requestUrl: "~/ControllerName");
   var routes = new RouteCollection();
   MyMVCApplication.RouteConfig.RegisterRoutes(routes);

   // Act
   RouteData routeData = routes.GetRouteData(context);

   // Assert
   Assert.AreEqual("ControllerName", routeData.Values["controller"]);
   Assert.AreEqual("Index", routeData.Values["action"]);
   Assert.AreEqual(UrlParameter.Optional, routeData.Values["id"]);
}
```

Lesson 3: Creating a Navigation Structure

- The Importance of Well-Designed Navigation
- Configuring the MVC Site Map Provider
- Adding Menu Controls
- Demonstration: How to Build Site Navigation

The Importance of Well-Designed Navigation

- Ensure that users can easily decide what link to click on each page

- Provide navigation controls, such as:
  - Top Menus
  - Tree Views
  - Breadcrumb Trails
  - Footer Menus

- Use the MVC Site Map Provider to rapidly build information architecture and navigation controls

Configuring the MVC Site Map Provider

```
<?xml version="1.0" encoding="utf-8" ?>
<mvcSiteMap xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://mvcsitemap.codeplex.com/schemas/MvcSiteMap-File-3.0"
  xsi:schemaLocation=
    "http://mvcsitemap.codeplex.com/schemas/
      MvcSiteMap-File-3.0 MvcSiteMapSchema.xsd“
  enableLocalization="true">

  <mvcSiteMapNode title="Home" controller="Home" action="Index">
    <mvcSiteMapNode title="Products" controller="Product" action="Index"
      key="AllProducts">
      <mvcSiteMapNode title="Bikes" controller="Category" action="Display“ />
    </mvcSiteMapNode>
    <mvcSiteMapNode title="Latest News" controller="Article"
      action="DisplayLatest" >
    <mvcSiteMapNode title="About Us" controller="Home" action="About" />
  </mvcSiteMapNode>

</mvcSiteMap>
```

Adding Menu Controls

- Rendering a Menu:

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

- Rendering a Breadcrumb Trail:

```
@Html.MvcSiteMap().SiteMapPath()
```

Demonstration: How to Build Site Navigation

In this demonstration, you will see how to:

1. Install the MVC site map provider
2. Configure site map nodes in the Home Index view and Opera Index view
3. Use site map helpers to add menus and breadcrumb trails
4. Navigate through the MVC web application by using the menus and breadcrumb trails

Lab: Structuring ASP.NET MVC 4 Web Applications

- Exercise 1: Using the Routing Engine
- Exercise 2: Optional—Building Navigation Controls

Estimated Time: 40 minutes

Lab Scenario

An important design priority for the Photo Sharing application is that the visitors should be able to easily and logically locate photographs. Additionally, photo galleries and photos need to appear high in search engine results. To implement these priorities, you have been asked to configure routes that enable the entry of user-friendly URLs to access photos. You have been asked to ensure that the URLs of the following forms work to display a photo:

- ~/photo/display/*PhotoId*. In this form of URL, PhotoId is the database ID of the photo object. This form of URL already works because it matches the default route.
- ~/photo/*PhotoId*. In this form of URL, PhotoId is the database ID of the photo object. This is the logical URL to enter when you know the ID of the photo that you want to access.
- ~/photo/title/*PhotoTitle*. In this form of URL, PhotoTitle is the title of the photo object. This is the logical URL to enter when you know the title of the photo that you want to access.

Lab Scenario (Continued)

- You have also been asked to implement the following navigation controls in the Photo Sharing application:
    - A menu with links to the main site areas
    - A breadcrumb control
- These navigation controls will be added to the menu after the completion of the main site areas.

Lab Review

- 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?
- Why is the constraint necessary in the PhotoRoute route?

Module Review and Takeaways

- Review Question(s)
- Best Practice
- Common Issues and Troubleshooting Tips