Agenda: URL Routing

- Overview
- Mapping URL to Controller Action Method
- Applying Constraints on Route parameters
- Resolving Namespace Ambiguity
- Ignoring Routes
- Attribute Routing improvements in MVC 5

URL Routing Overview

The ASP.NET MVC framework uses the ASP.NET routing engine, which provides flexibility for mapping URLs to controller classes. You can define routing rules that the ASP.NET MVC framework uses in order to evaluate incoming URLs and to select the appropriate controller. You can also have the routing engine automatically parse variables that are defined in the URL, and have the ASP.NET MVC framework pass the values to the controller action methods as parameters.

Global URL Routing Defaults:

Routes are initialized in the **Application_Start** method of the Global.asax file. The following example shows a typical Global.asax file that class RourteConfig.RegisterRoutes which includes default routing logic.

```
public class MvcApplication : System.Web.HttpApplication
{
 protected void Application_Start()
  {
     // . . .
     RouteConfig.RegisterRoutes(RouteTable.Routes);
     // . . .
  }
}
public class RouteConfig
{
   public static void RegisterRoutes(RouteCollection routes)
   {
      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

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

```
    }
}
```

Implementing Routing for Employee Application:

Add following route methods to the Global.asax file

```
routes.MapRoute(
    "Create", // Route name
    "CreateEmp", // URL with parameters
     new { controller = "Employee", action = "Create" } // Parameter defaults
);
routes.MapRoute(
    "Edit", // Route name
    "EditEmp/{Id}", // URL with parameters
    new { controller = "Employee", action = "Edit" } // Parameter defaults
);
routes.MapRoute(
    "Delete", // Route name
    "DeleteEmp/{id}", // URL with parameters
    new { controller = "Employee", action = "Delete" } // Parameter defaults
);
```

Note: All the entries must be above "Default" route because otherwise it will take precedence and others will not be useful.

Resolving Namespace Ambiguity

If we have two HomeController Classes in different namespaces.

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },
    namespaces: new[] { "MyDemoApp.Controllers" }
);
//OR
ControllerBuilder.Current.DefaultNamespaces.Add("MyDemoApp.Controllers");
```

Ignoring Route URL’s

In case we don't want routing to be applied for a particular type of document or URL:

routes.**IgnoreRoute**("Home/Index/123");

is added to RegisterRoutes method

then the url: http://localhost:49165/Home/Index/123

will render HTTP Error 404.0 - Not Found

Attribute Routing improvements in MVC 5

In RouteConfig.cs – Add the following:

routes.MapMvcAttributeRoutes();

Note: This must be added before the route table configuration.

```
[RoutePrefix("Demo")]
public class DemoController : Controller
{
// eg.: /Demo
   [Route]
   public ActionResult Index() { ... }

// eg.: /Demo/5
   [Route("{id}")]
   public ActionResult Show(int id) { ... }

// eg.: /Demo/5/edit
   [Route("{id}/edit")]
   public ActionResult Edit(int id) { ... }
}
```

Route Names: You can specify a name for a route, in order to easily allow URI generation for it. For example, for the following route:

```
[Route("menu", Name="mainmenu")]
public ActionResult MainMenu() { ... }

<a href="@Url.RouteUrl("mainmenu")">Main menu</a>
```

Optional URI Parameters (via the '?' modifier) and Default Values

```
public class BooksController : Controller
{
   // eg: /books
   // eg: /books/1430210079
```

```
  [Route("books/{isbn?}")]
  public ActionResult View(string isbn)
  {
    if (!String.IsNullOrEmpty(isbn))
    {
      return View("OneBook", GetBook(isbn));
    }
    return View("AllBooks", GetBooks());
  }

  // eg: /books/lang
  // eg: /books/lang/en
  // eg: /books/lang/hi
  [Route("books/lang/{lang=en}")]
  public ActionResult ViewByLanguage(string lang)
  {
    return View("OneBook", GetBooksByLanguage(lang));
  }
}
```

Applying Constraints on Route parameters

```
routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{name}/{id}",
        defaults: new { controller = "Home", action = "Index", name="test", id = UrlParameter.Optional },
        constraints: new { name="[a-z]{5}", id = @"\d{1,8}" }
    );
```

```
[Route("users/{id:int}"]
public ActionResult GetUserById(int id) { ... }
// eg: users/ken
[Route("users/{name}"]
public ActionResult GetUserByName(string name) { ... }
```

Here, the first route will only be selected if the "id" segment of the URI is an integer. Otherwise, the second route will be chosen.

The following table lists the constraints that are supported.

Constraint	Description	Example

alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	{x:alpha}
bool	Matches a Boolean value.	{x:bool}
datetime	Matches a **DateTime** value.	{x:datetime}
decimal	Matches a decimal value.	{x:decimal}
double	Matches a 64-bit floating-point value.	{x:double}
float	Matches a 32-bit floating-point value.	{x:float}
guid	Matches a GUID value.	{x:guid}
int	Matches a 32-bit integer value.	{x:int}
length	Matches a string with the specified length or within a specified range of lengths.	{x:length(6)} {x:length(1,20)}
long	Matches a 64-bit integer value.	{x:long}
max	Matches an integer with a maximum value.	{x:max(10)}
maxlength	Matches a string with a maximum length.	{x:maxlength(10)}
min	Matches an integer with a minimum value.	{x:min(10)}
minlength	Matches a string with a minimum length.	{x:minlength(10)}
range	Matches an integer within a range of values.	{x:range(10,50)}
regex	Matches a regular expression.	{x:regex(^\d{3}-\d{3}-\d{4}$)}

To apply multiple constraint:

Here, the first route will only be selected if the "id" segment of the URI is an integer. Otherwise, the second route will be chosen.

Note that specifying that a parameter is Optional (via the '?' modifier) should be done after inline constraints:

```
// eg: /greetings/bye
// and /greetings because of the Optional modifier,
// but not /greetings/see-you-tomorrow because of the maxlength(3) constraint.
[Route("SayHello/{message:minlength(3):maxlength(10)?}")]
public ActionResult SayHello(string message) { ... }
```