Attribute Routing With ASP.net MVC 5

Introduction

- This Article shows how to use the Latest ASP.net MVC 5 **Attribute Routing** with your Application.
- This Article has 2 parts.First part of this Article will show the basic usage of **Attribute Routing**.
- Now you can read the 2nd part of this Article **here 'Attribute Routing With ASP.net MVC 5 - Route Constraints'**

What is Routing ?

- **Routing** is how ASP.net MVC matches a URI to an Action

What is Attribute Routing ?

- **ASP.net MVC 5** supports a new type of Routing, called **Attribute Routing**
- As the name implies, attribute routing uses **attributes to define routes**
- Attribute routing gives you **more control** over the URIs in your web application

How To Enable Attribute Routing ?

- For that, You have to select the **RouteConfig.cs** inside the **App_Start** Folder.
- After that call **MapMvcAttributeRoutes** is as below.

RouteConfig.cs

```
public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapMvcAttributeRoutes();//Attribute Routing

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

Key points of the above code

- To enable Attribute Routing,You have to call **MapMvcAttributeRoutes** on **RouteConfig** File.
- If you want, You can keep the **Convention-based Routing** also with the same file is as above.
- But **`routes.MapMvcAttributeRoutes();`** Should configure **before** the Convention-based Routing.

How to use Optional URI Parameters ?

- To that you can add a **question mark** to the Route parameter
- Well, It's like this : `[Route("Pet/{petKey?}")]`

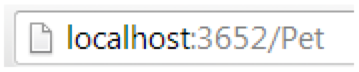
PetController.cs

```
public class PetController : Controller
  {
      // eg: /Pet
      // eg: /Pet/123
      [Route("Pet/{petKey?}")]
      public ActionResult GetPet(string petKey)
      {
          return View();
      }
  }
```

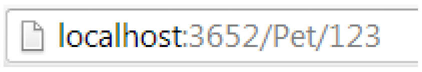
Key point of the above code

- In the above example, both **/Pet** and **/Pet/123** will Route to the **"GetPet" Action**

Above Route on Browser is as below


localhost:3652/Pet

OR


localhost:3652/Pet/123

How to use Default Values URI Parameters ?

- To that you can specify a **default value** to the **route parameter**
- It's like this : `[Route("Pet/Breed/{petKey=123}")]`

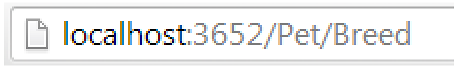
PetController.cs

```
public class PetController : Controller
 {
    // eg: /Pet/Breed
    // eg: /Pet/Breed/528
    [Route("Pet/Breed/{petKey=123}")]
    public ActionResult GetSpecificPet(string petKey)
    {
        return View();
    }
 }
```

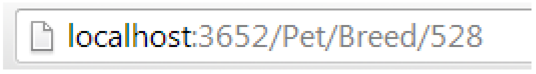
Key point of the above code

- In the above example, both **/Pet/Breed** and **/Pet/Breed/528** will route to the **"GetSpecificPet"** Action

Above Route on Browser is as below


localhost:3652/Pet/Breed

OR


localhost:3652/Pet/Breed/528

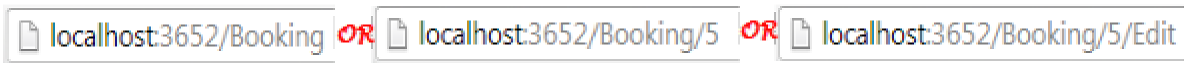
How to use Route Prefixes ?

- Normally, the routes in a controller **all start with the same prefix**
- Well,It's like this : **`/Booking`**

BookingController.cs

```
public class BookingController : Controller
    {
        // eg: /Booking
        [Route("Booking")]
        public ActionResult Index() { return View(); }

        // eg: /Booking/5
        [Route("Booking/{bookId}")]
        public ActionResult Show(int bookId) { return View(); }

        // eg: /Booking/5/Edit
        [Route("Booking/{bookId}/Edit")]
        public ActionResult Edit(int bookId) { return View(); }
    }
```

Above Routes on Browser are as below


localhost:3652/Booking OR localhost:3652/Booking/5 OR localhost:3652/Booking/5/Edit

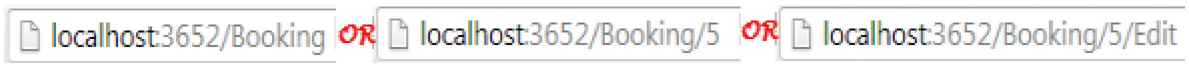
How to Set Common Route Prefix ?

- If you want, you can specify a **common prefix for an entire controller**
- To that you can use **[RoutePrefix] attribute**
- It's like this : `[RoutePrefix("Booking")]`

BookingController.cs

```
    [RoutePrefix("Booking")]
    public class BookingController : Controller
    {

        // eg: /Booking
        [Route]
        public ActionResult Index() { return View(); }

        // eg: /Booking/5
        [Route("{bookId}")]
        public ActionResult Show(int bookId) { return View(); }

        // eg: /Booking/5/Edit
        [Route("{bookId}/Edit")]
        public ActionResult Edit(int bookId) { return View(); }

    }
```

Above Routes on Browser are as below


localhost:3652/Booking OR localhost:3652/Booking/5 OR localhost:3652/Booking/5/Edit

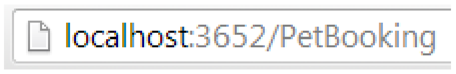
How to Override the Common Route Prefix ?

- You can use a **tilde (~)** on the method attribute to **override the route prefix**
- Well,It's like this : `[Route("~/PetBooking")]`

BookingController.cs

```
[RoutePrefix("Booking")]
public class BookingController : Controller
{
    // eg: /PetBooking
    [Route("~/PetBooking")]
    public ActionResult PetBooking() { return View(); }
}
```

Above Route on Browser is as below


localhost:3652/PetBooking

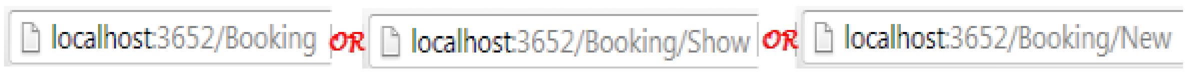
How to use Default Route ?

- You can apply the **[Route] attribute** on the **Controller level** and put the **Action as a parameter**
- That Route will then be applied on **all Actions** in the **Controller**
- Well,It's like this : `[Route("{action=index}")]`

BookingController.cs

```
[RoutePrefix("Booking")]
[Route("{action=index}")]
public class BookingController : Controller
{
    // eg: /Booking
    public ActionResult Index() { return View(); }

    // eg: /Booking/Show
    public ActionResult Show() { return View(); }

    // eg: /Booking/New
    public ActionResult New() { return View(); }

}
```

Above Routes on Browser are as below


localhost:3652/Booking OR localhost:3652/Booking/Show OR localhost:3652/Booking/New

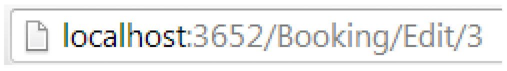
How to override Default Route ?

- For that you have to use **specific [Route] on a specific Action.**
- It'll **override the default** settings on the **Controller.**

BookingController.cs

```
[RoutePrefix("Booking")]
[Route("{action=index}")]
public class BookingController : Controller
{
    // eg: /Booking
    public ActionResult Index() { return View(); }

    // eg: /Booking/Edit/3
    [Route("Edit/{bookId:int}")]
    public ActionResult Edit(int bookId) { return View(); }

}
```

Above overridden Route on Browser is as below


localhost:3652/Booking/Edit/3

How to give Route Names ?

- You can specify a **Name for a Route**
- By using that **Name,** you can easily allow **URI** generation for it
- Well,It's like this : `[Route("Booking", Name = "Payments")]`

BookingController.cs

```
public class BookingController : Controller
   {
       // eg: /Booking
       [Route("Booking", Name = "Payments")]
       public ActionResult Payments() { return View(); }
   }
```

- After that you can generate a **Link** is using **Url.RouteUrl**
- It's like this :

```
<a href="@Url.RouteUrl("Payments")">Payments Screen</a>
```

Note : On the above code, "Payments" is a **Route Name**

Advantages of Attribute Routing Over the Convention-based Routing

- Attribute Routing gives you **more control over the URIs** in your web application
- **Easy to Troubleshoot** issues
- **No fear of modifying** anything will break another route down the line

Attribute Routing With ASP.net MVC 5 - Route Constraints

Introduction

- This Article shows how to use the Latest ASP.net MVC 5 **Attribute Routing's Route Constraints** with your Application.
- You can read the **first part of this Article** **here 'Attribute Routing With ASP.net MVC 5'**

How to set Route Constraints ?

- It allows you to **restrict the parameters** in the route template are matched
- The syntax is **{parameter:constraint}**

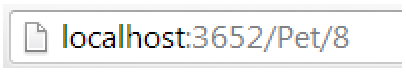
PetController.cs

```
public class PetController : Controller
 {
   // eg: /Pet/8
   [Route("Pet/{petId:int}")]
   public ActionResult GetSpecificPetById(int petId)
   {
       return View();
    }
}
```

Key points of the above code

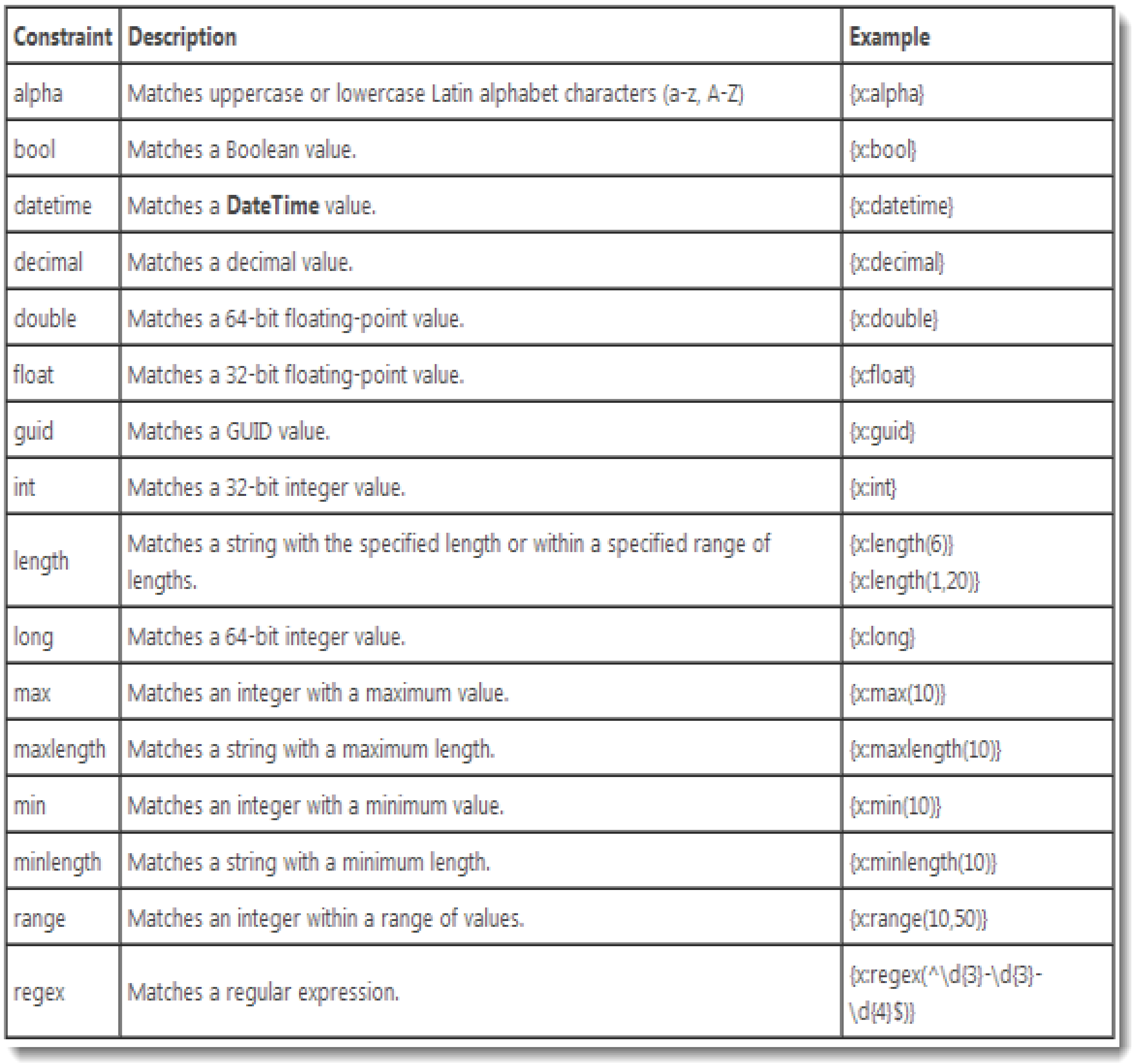
- In the above example, **/Pet/8** will Route to the **"GetSpecificPetById"** Action.
- Here the route will only be selected, if the **"petId"** portion of the URI is an **integer.**

Above Route on Browser is as below



localhost:3652/Pet/8

The following diagram shows 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}$)}

How to apply multiple constraints to a parameter ?

- You can apply multiple constraints to a parameter, **separated by a colon.**
- Well,It's like this `[Route("Pet/{petId:int:min(1)}")]`

PetController.cs

```
public class PetController : Controller
 {
   // eg: /Pet/8
   [Route("Pet/{petId:int:min(1)}")]
   public ActionResult GetSpecificPetById(int petId)
   {
       return View();
```

```
        }
    }
```

Key points of the above code

- In the above example,You **can't use /Pet/10000000000** ,because it is **larger than int.MaxValue**
- And also you **can't** use **/Pet/0** ,because of the **min(1) constraint.**

How to Specifying that a parameter is Optional ?

- You can do it by Specifying that a parameter is **Optional (via the '?' modifier).**
- This should be done **after inline constraints.**
- Well,it's like this `[Route("Pet/{message:maxlength(4)?}")]`

PetController.cs

```
// eg: /Pet/good
[Route("Pet/{message:maxlength(4)?}")]
public ActionResult PetMessage(string message)
{
    return View();
}
```

Key points of the above code

- In the above example, **/Pet/good** and **/Pet** will Route to the **"PetMessage"** Action.
- The route **/Pet** also **works** hence of the **Optional modifier.**
- But **/Pet/good-bye** will **not route** above Action , because of the **maxlength(4) constraint.**

Above Routes on Browser are as below

localhost:3652/Pet/good

localhost:3652/Pet