**Inline Constraint**

Inline Constraints are added after the URL Parameter and separated by a colon : sign. For example, you can ensure only integer values are allowed by using the int constraint.

routes.MapRoute("default", "{controller=Home}/{action=Index}/{id:int?}");

The int constraint checks to see the value of the Id can be parsed to an integer value. The id segment is optional. Hence, the route will match if the Id is not present, but if the id is present, then it must be an integer value.

with int constraint in place, the Request /Home/Index/Test will not match the route.

**Using Constraint method of the MapRoute**

The Constraints can also be specified using the constraints argument to the MapRoute method.

To do that you need to import Microsoft.AspNetCore.Routing.Constraints namespace.

using Microsoft.AspNetCore.Routing.Constraints;

app.UseMvc(routes =>

{

routes.MapRoute("default",

"{controller}/{action}/{id}",

new { controller = "Home", action = "Index" },

new { id = new IntRouteConstraint() });

});

We create an instance of an Anonymous type, which contains properties, whose property name are same as the URL Parameters, on which constraints are applied. These Properties are assigned to the instance of the Constraint class.

In the above example, we have created an Anonymous class. It has id property, which matches to the id URL Parameter. The Instance of IntRouteConstraint is assigned to the id Property.

**Constraints in Attribute Routing**

You can achieve the same using Attribute routing as follows

[Route("Home/Index/{id:int}")]

public string Index(int id)

{

return "I got " + id.ToString();

}

**List of Route Constraints**

The **Microsoft.AspNetCore.Routing.Constraints** namespace defines a set of classes that can be used to define individual constraints.

**Constraints for Checking Data Type**

The following Constrains checks the data type.

| CONSTRAINT | INLINE | CLASS | NOTES |
| --- | --- | --- | --- |
| int | {id:int} | IntRouteConstraint | Constrains a route parameter to represent only 32-bit integer values |
| alpha | {id:alpha} | AlphaRouteConstraint | Constrains a route parameter to contain only lowercase or uppercase letters A through Z in the English alphabet. |
| bool | {id:bool} | BoolRouteConstraint | Constrains a route parameter to represent only Boolean values. |
| datetime | {id:datetime} | DateTimeRouteConstraint | Constrains a route parameter to represent only DateTime values. |
| decimal | {id:decimal} | DecimalRouteConstraint | Constrains a route parameter to represent only decimal values. |
| double | {id:double} | DoubleRouteConstraint | Constrains a route parameter to represent only 64-bit floating-point values |
| float | {id:float} | FloatRouteConstraint | Matches a valid float value (in the invariant culture - see warning) |
| guid | {id:guid} | GuidRouteConstraint | Matches a valid Guid value |

**Constraints for Checking Data Value/Range/Length**

The following Constrains checks the Value/Range/Length etc.

| CONSTRAINT | INLINE | CLASS | NOTES |
| --- | --- | --- | --- |
| length(length) | {id:length(12)} | LengthRouteConstraint | Constrains a route parameter to be a string of a given length or within a given range of lengths. |
| maxlength(value) | {id:maxlength(8)} | MaxLengthRouteConstraint | Constrains a route parameter to be a string with a maximum length. |
| minlength(value) | {id:minlength(4)} | MinLengthRouteConstraint | Constrains a route parameter to be a string with a maximum length. |
| range(min,max) | {id:range(18,120)} | RangeRouteConstraint | Constraints a route parameter to be an integer within a given range of values. |
| min(value) | {id:min(18)} | MinRouteConstraint | Constrains a route parameter to be a long with a minimum value. |
| max(value) | {id:max(120)} | MaxRouteConstraint | Constrains a route parameter to be an integer with a maximum value. |

**Constraints using a Regular Expression**

Using Regular expression or regex as constraint offers more flexibility to limit any input.

| CONSTRAINT | INLINE | CLASS | NOTES |
| --- | --- | --- | --- |
| regex(expression) | {ssn:regex(^\\d{{3}}-\\d{{2}}-\\d{{4}}$)} | RegexRouteConstraint | Constrains a route parameter to match a regular expression. |

**Example of Regular Express Route Constraint**

In the following example, we using regular expression to restrict the Year value to 4 digits

app.UseMvc(routes =>

   {

   routes.MapRoute("default",

        "{controller}/{action}/{year:regex(^\\d{{4}}$)}",

        new { controller = "Home", action = "Index" });

});

Update the Index method of the HomeController.cs

public class HomeController : Controller

{

    public string Index(int year)

    {

        return "Year = " + year.ToString();

     }

}

A request for /Home/Index/2017 will match the above route, but request for /Home/Index/20 will fail

Note that Regular expression tokens must be escaped. For example, \,{,},[,] characters need to be escaped by doubling them to escape the Routing parameter delimiter characters.

Hence, ^\d{4}$ becomes ^\\d{{4}}$ in the above example.

**Combing Constraints**

Multiple Constraints can be combined using a colon : separator as shown below

"/{id:alpha:minlength(6)?}"

or using the Constraints method of the MapRoute.

Using Microsoft.AspNetCore.Routing.CompositeRouteConstraint;

 constraints: new {

    id = new CompositeRouteConstraint(

    new IRouteConstraint[] {

    new AlphaRouteConstraint(),

    new MinLengthRouteConstraint(6)

})