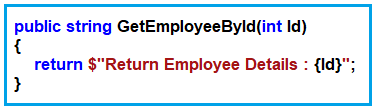
## **Variables and Query Strings in ASP.NET Core Web API Routing**

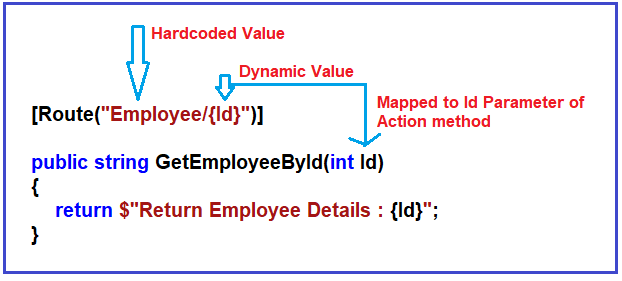
**Working with Variables in ASP.NET Core Web API Routing:**

When working with Real-time Restful services, then we need to deal with dynamic values like getting Order Details by Id, Get Employee details by Id, Get books by Author, etc. Let us see how we can handle the variables in the Routes in ASP.NET Core Web API Application.

For example, we have multiple employees and we want to fetch one employee detail by its Id. Then how we can get the Id values? The URL is the only place that is going to give us the Id value. So, we need to define one parameter to take the Id value within the method signature as shown in the below image

 .

In ASP.NET Core Web Application, if you want to pass anything as a variable then you need to use curly braces {} and inside the curly braces, you need to give the name of the parameter your method accepting. In our example, the GetEmployeeById method takes the Id parameter, so we need to pass the Id within the curly braces of the Route attribute as shown in the below image.



So, modify the EmployeeController class as shown below.

**using** *Microsoft.AspNetCore.Mvc;*

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**public** **class** EmployeeController : ControllerBase

**{**

**[**Route**(**"Employee/{Id}"**)]**

**public** string GetEmployeeById**(int** Id**)**

**{**

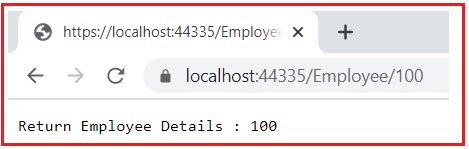
**return** $"Return Employee Details : {Id}";

**}**

**}**

**}**

Now you can access the GetEmployeeById action method using the URL: **Domainname/Employee/100** (instead of 100 you can pass any dynamic value i.e. the Employee Id which information you want to retrieve). So run the above application and see the output as shown in the below image.



This is how we pass dynamic values using routing.

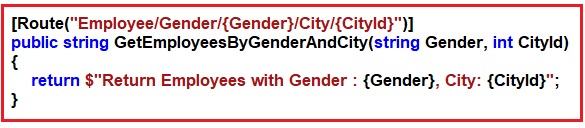
**Passing Multiple dynamic Values in ASP.NET Core Web API Routing:**

Now let us understand how to pass multiple dynamic values using the Route attribute. Now we need to fetch all the employees by Gender and City. Here, we want the Gender to be Male or Female i.e. as a string value and intentionally, we want to City Id value i.e. an integer value. So, we will create one action method which takes two parameters, one of which is Gender, and the order one is CityId as shown in the below image.



Now we want to access the above GetEmployeesByGenderAndCity method using the URL: **Employee/Gender/Male/City/10**

Here, Male and 10 are the dynamic values. So, we need to decorate the GetEmployeesByGenderAndCity method with the Route Attribute as shown in the below image. Here, we are passing the Gender and CityId parameter in curly braces.



So, modify the EmployeeController class as shown below.

**using** *Microsoft.AspNetCore.Mvc;*

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**public** **class** EmployeeController : ControllerBase

**{**

**[**Route**(**"Employee/{Id}"**)]**

**public** string GetEmployeeById**(int** Id**)**

**{**

**return** $"Return Employee Details : {Id}";

**}**

**[**Route**(**"Employee/Gender/{Gender}/City/{CityId}"**)]**

**public** string GetEmployeesByGenderAndCity**(**string Gender, **int** CityId**)**

**{**

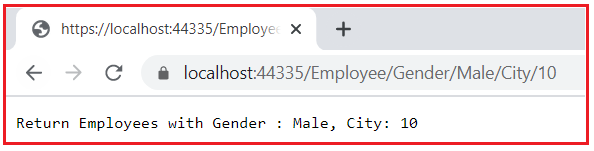
**return** $"Return Employees with Gender : {Gender}, City: {CityId}";

**}**

**}**

**}**

Now run the application and navigate to the URL Employee/Gender/Male/City/10 and you should get the message as expected as shown in the below image.

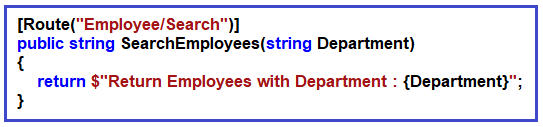


This is how you can pass multiple dynamic values in your routing.

**Working with Query Strings in ASP.NET Core Web API Routing:**

Let us understand how to work with Query Strings in ASP.NET Core Web API Routing. Query Strings are nothing but key-value pairs that you need to pass as of the URL. Again, you pass multiple query strings (i.e. multiple key-value pairs) separated by &. Further, the most important point that you need to remember is before the first query string or after the domain name you need to use a question mark (?). The question mark (?) in the URL indicates that the query string is started.

Let us understand the query string with an example. Now, we need to search employees by department name. But we don’t want the department name to be part of the Route Attribute. Then how we can do this? We can do this by using the query string. So, let us create a method with the name SearchEmployee with one parameter called Department as shown in the below image. Further notice, we have not included that parameter as part of the Route Attribute.



The complete code of EmployeeController is given below.

**using** *Microsoft.AspNetCore.Mvc;*

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**public** **class** EmployeeController : ControllerBase

**{**

**[**Route**(**"Employee/Search"**)]**

**public** string SearchEmployees**(**string Department**)**

**{**

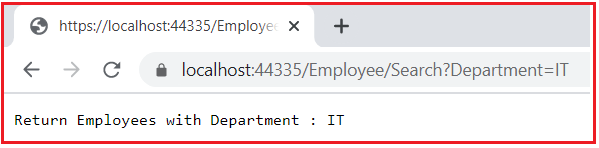
**return** $"Return Employees with Department : {Department}";

**}**

**}**

**}**

The URL to access the above action method is domainname/Employee/Search, but how we can pass the value for the Department parameter. The answer is the query string. So, what we need to do is just append **?Department=IT** to the end of the URL and press enter as shown in the below image.



**How to pass Multiple Query Strings in ASP.NET Core Web API?**

Let us understand how to pass multiple query strings with an example. In a real-time application, when you implement a search functionality, you generally accept multiple search parameters to filter out the data, Let say, we want employee city, gender, and department to filter out the number of employees to be returned. Then, in that case, our action method accepting three parameters. So, modify the SearchEmployees method of the Employee Controller class as shown below.

**[**Route**(**"Employee/Search"**)]**

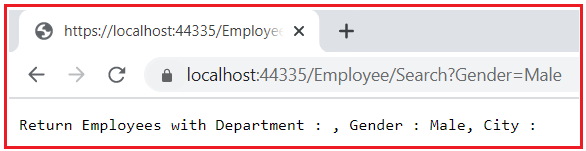
**public** string SearchEmployees**(**string Department, string Gender, string City**)**

**{**

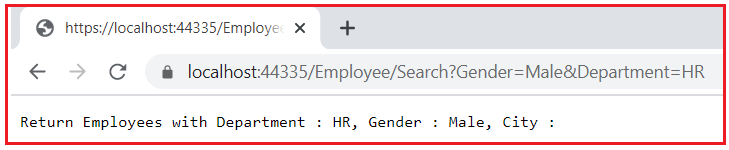
**return** $"Return Employees with Department : {Department}, Gender : {Gender}, City : {City}";

**}**

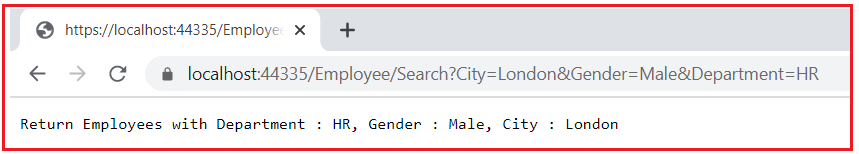
With the above changes in place, now run the application. Now let say we want to filter the employees by Gender only. Then you can only pass the Gender query string in the URL as shown in the below image.



Now, you want to search the employees by Gender and Department. Then you can pass two query strings in the URL as shown in the below image.



Now, you want to search the employees by using all three parameters i.e. Gender, Department, and City. Then you can pass the query strings in the URL as shown in the below image. The point that you need to remember the order of the query does not matter i.e. you can pass the query strings in any order.



This is how query strings work in ASP.NET Core Web API Routing.