**Base Route in ASP.NET Core Web API Routing**

Let’s understand the need and use of Base Route in ASP.NET Core Web API Routing with an Example. Let us first modify the EmployeeController class as shown below.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

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

**{**

**[**Route**(**"employee/all"**)]**

**public** string GetAllEmployees**()**

**{**

**return** "Response from GetAllEmployees Method";

**}**

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

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

**{**

**return** $"Response from GetEmployeeById Method, Id : {Id}";

**}**

**[**Route**(**"employee/department/{Department}"**)]**

**public** string GetDepartmentEmployees**(**string Department**)**

**{**

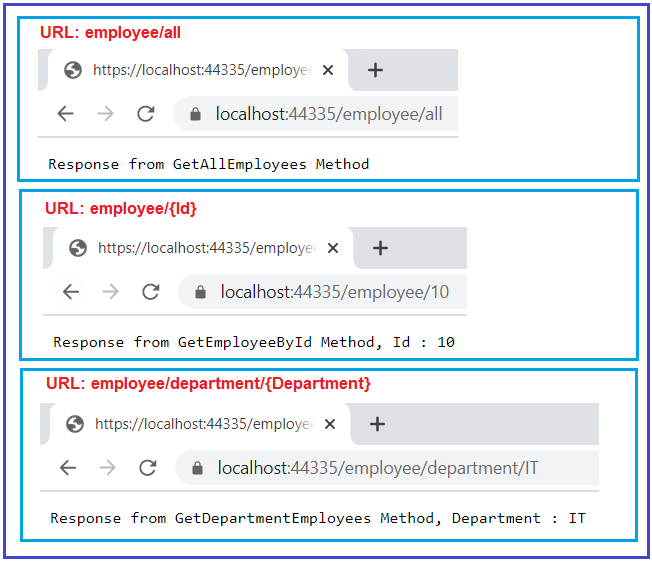
**return** $"Response from GetDepartmentEmployees Method, Department : {Department}";

**}**

**}**

**}**

Now you can access the above three methods as shown in the below image.



As you can see in the above example, we are using the Route attributes at the action level to define the routes, and furthermore, all the routes in the **EmployeeController** are starts with the same prefix – employee. That means the employee is the common prefix that is used for all the routes available in the Employee Controller at the moment.

Is not it good enough, if you can move the common attribute prefix to the controller level? Yes, we can. So, basically, the prefix which is common for all the routes should be placed at the controller level and the route which is specific to the action method should be placed at the action method level. So, in our example, the common prefix i.e. employee should be placed at the controller level and the rest should be placed at the action method level as shown below.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**[**Route**(**"employee"**)]**

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

**{**

**[**Route**(**"all"**)]**

**public** string GetAllEmployees**()**

**{**

**return** "Response from GetAllEmployees Method";

**}**

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

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

**{**

**return** $"Response from GetEmployeeById Method, Id : {Id}";

**}**

**[**Route**(**"department/{Department}"**)]**

**public** string GetDepartmentEmployees**(**string Department**)**

**{**

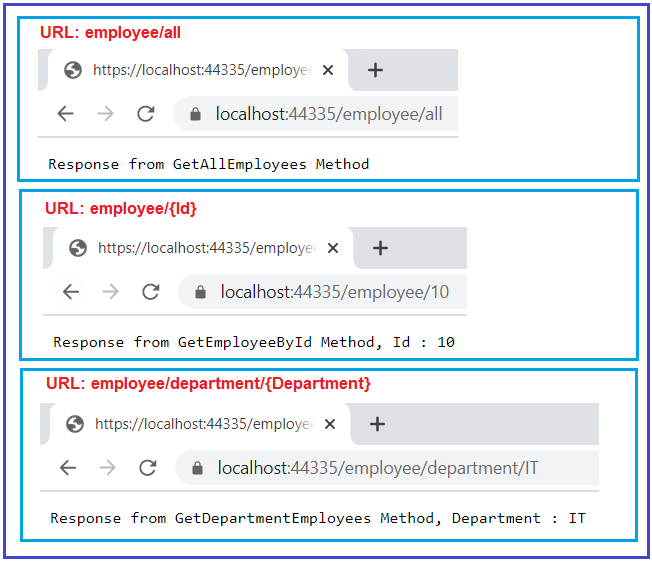
**return** $"Response from GetDepartmentEmployees Method, Department : {Department}";

**}**

**}**

**}**

Now with the above changes in place, you can access the above three resources in the same way that we access in our previous example as shown in the below image.



With the above changes in place, we eliminate the need to repeat the common prefix “employee” on each and every controller action method. However, sometimes we may need to override the common route prefix attribute.

##### **How to override the Base Route in ASP.NET Core Web API Attribute Routing?**

Let us understand how to override the common route prefix or base route in ASP.NET Core Web API Attribute Routing with an example. At this moment our Employee Controller class contains three action methods and all these action methods start with the same Route Prefix i.e. employee.

Now, we need to add one action method within the employee controller for returning all departments. And we want to access this resource using the URL: department/all. So, let us first add the following GetAllDepartment method and decorate it with [Route(“department/all”)] Attribute as shown below within the EmployeeController.

**[**Route**(**"department/all"**)]**

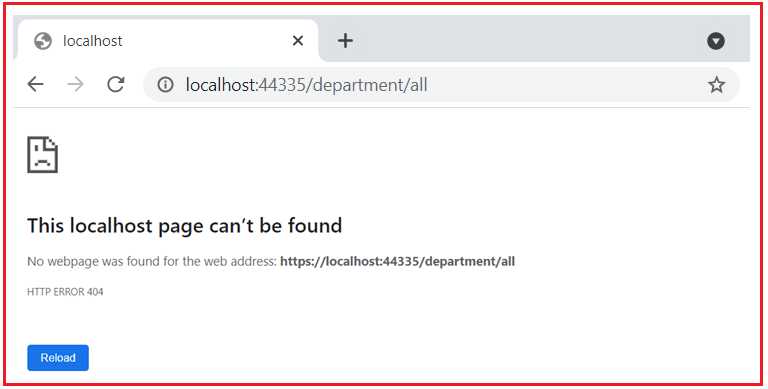
**public** string GetAllDepartment**()**

**{**

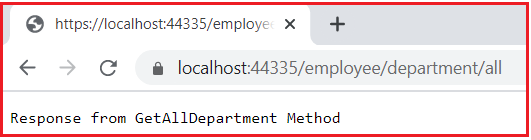
**return** "Response from GetAllDepartment Method";

**}**

With the above **[Route(“department/all”)]**attribute on GetAllDepartment() method and when we navigate to **department/all**, we will get the following error.



But if we navigate to **/employee/department/all** then we will get the output as expected as shown in the below image. This is because the [Route(“employee”)] attribute that is defined at the Employee Controller.



Now the question that should come to your mind is how to override the Route attribute used in the EmployeeController i.e. Route Attribute defined at the controller level.

##### **How to override the Controller level Route Attribute at the action method level?**

In ASP.NET Core Application, you can override the Controller level Route Attribute at the action method level by using the **~** (tilde) symbol. So, modify the GetAllDepartment action method as shown below to use the tilde symbol to override the route defined at the employee controller.

**[**Route**(**"~/department/all"**)]**

**public** string GetAllDepartment**()**

**{**

**return** "Response from GetAllDepartment Method";

**}**

With the above change in place, now the **GetAllDepartment**() action method is mapped to URI “**/department/all**” as expected as shown in the below image.

