**Token Replacement in ASP.NET Core Web API Routing**

##### **What are Tokens in ASP.NET Core Attribute Routing?**

Token Replacement is a new feature available in ASP.NET Core and it was not available in .NET Frameworks like ASP.NET MVC and ASP.NET Web API. The meaning of token replacement is, we can replace the value of the controller and action method dynamically.

In ASP.NET Core Web API Application, the Route Attribute support token replacement. It means we can enclose the token (i.e. controller and action) within a pair of square braces ([]). The tokens (i.e. [controller] and [action]) are then replaced with the values of controller and action method name where the route is defined.

##### **Example: without using Token Replacement in ASP.NET Core Web API**

Before understanding the need and use of token replacement, let us first understand an example without using token replacement. Suppose we have two resources in our Employee Controller and we want to access the two resources using the controller/action method name. Then we can do the same without using token replacement as shown in the below code.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

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

**{**

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

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

**{**

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

**}**

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

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

**{**

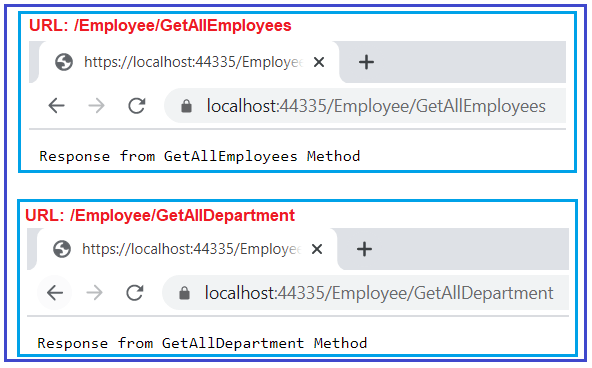
**return** "Response from GetAllDepartment Method";

**}**

**}**

**}**

Now run the application and you can access both the resource using the controller and action method name as shown in the below image.



##### **Example: Token Replacement in ASP.NET Core Web API Application**

Let us understand how the token replacement work in ASP.NET Core Web API Application with an example. Please modify the Employee Controller class that we have been working so far as shown below. As you can see, here we are applying the token **[controller]** on the EmployeeController and at the same time, we are also applying the token **[action]** on all the action methods of the Employee Controller.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**[**Route**(**"[controller]"**)]**

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

**{**

**[**Route**(**"[action]"**)]**

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

**{**

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

**}**

**[**Route**(**"[action]"**)]**

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

**{**

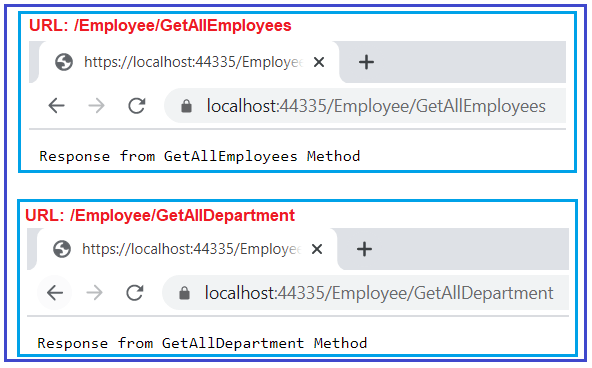
**return** "Response from GetAllDepartment Method";

**}**

**}**

**}**

Now, with the above changes in place, you can access the **GetAllEmployees** method with the URL: **/Employee/GetAllEmployees** and **GetAllDepartment** method with the URL: **/Employee/GetAllDepartment** as shown in the below image. This is because at the run time, the token **[controller]** will be replaced by Employee and the token **[action]** will be replaced by the respective action method of the controller.



##### **Advantages of Tokens in Attribute Routing:**

The main advantage of using Token Replacement in ASP.NET Core Application is that, if you rename the controller’s name or the action method name then you do not have to change the route templates. The application is going to works with the new controller and action method names.

##### **Do we need to write the action token on each action method?**

Not Really. If you want all your action methods of the controller to apply an action token, then instead of including the [action] token on each and every action method, you can apply it only once on the controller.

Let us understand this with an example. Please modify the Employee Controller class as shown in the below code. As you can see in the below code, we have removed the [Route(“[action]”)] attribute from the action method and modify the Route attribute as [Route(“[controller]/[action]”)] which is applied at the controller level.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**[**Route**(**"[controller]/[action]"**)]**

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

**{**

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

**{**

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

**}**

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

**{**

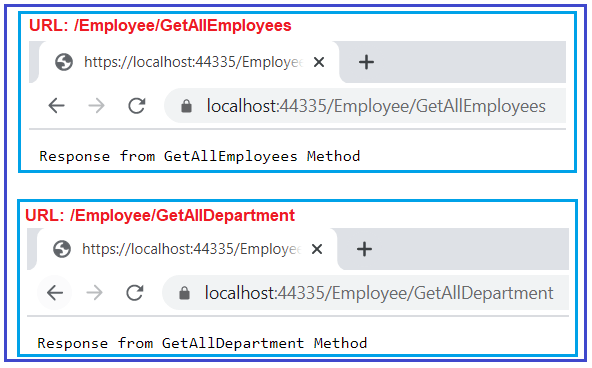
**return** "Response from GetAllDepartment Method";

**}**

**}**

**}**

Now, with the above changes in place, you can also access the **GetAllEmployees** method with the URL: **/Employee/GetAllEmployees** and **GetAllDepartment** method with the URL: **/Employee/GetAllDepartment** as shown in the below image.



##### **Token Replacement with dynamic Values:**

Now let us see an example of token replacement with dynamic values. Our requirement is to fetch employee information by employee id. Here, the employee id is the dynamic value and this value will come as part of the URL. Let us see how we can achieve this. Here, we need to decorate the Route attribute at the action method as shown below.

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

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

**{**

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

**}**

So, at the run time, the [Route(“[controller]/[action]”)] will evaluate first and then [Route(“{Id}”)] will be added to the route which will form the URL as **Employee/GetEmployeeById/10**. So, modify the Employee Controller class as shown below.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

**[**Route**(**"[controller]/[action]"**)]**

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

**{**

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

**{**

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

**}**

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

**{**

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

**}**

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

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

**{**

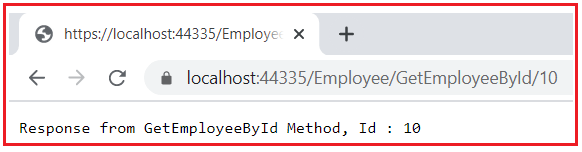
**return** $"Response from GetEmployeeById Method, Id : {Id}";

**}**

**}**

**}**

Now save the changes, run the application and access the **GetEmployeeById** as shown in the below image and you should the response as expected.



##### **Example: controller and action token applied to the action method**

In the below example, we are applying the Route Attribute only at the action method level and it is also going to work as expected.

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

**namespace** *RoutingInASPNETCoreWebAPI.Controllers*

**{**

**[**ApiController**]**

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

**{**

**[**Route**(**"[controller]/[action]"**)]**

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

**{**

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

**}**

**[**Route**(**"[controller]/[action]"**)]**

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

**{**

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

**}**

**[**Route**(**"[controller]/[action]/{Id}"**)]**

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

**{**

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

**}**

**}**

**}**