**Routing**

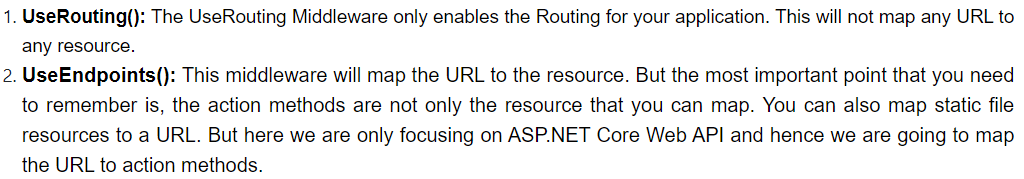
**How does the Routing work in ASP.NET Core Web API?**

Routing in ASP.NET Core Web API application is the process of mapping the incoming HTTP Request (URL) to a particular resource i.e., controller action method.

For the Routing Concept in ASP.NET Core Web API, we generally set some URLs for each resource. When we run the application, then it will create the Route table and the Route table will contain the mapping information between the URL and the Resource. So, when we are sending a request from the client to the server, then the application will check the URL in the Route table and if it found an exact, then the application will forward the request to that resource else it will throw an error saying resource not found.

We can access any resource using a unique URL in ASP.NET Core Web API Application. It is also possible that a resource can have multiple unique URLs. But multiple resources cannot have the same URL and if you do so, then the application gets confused to invoke which action method and as a result, you will get an ambiguity error.

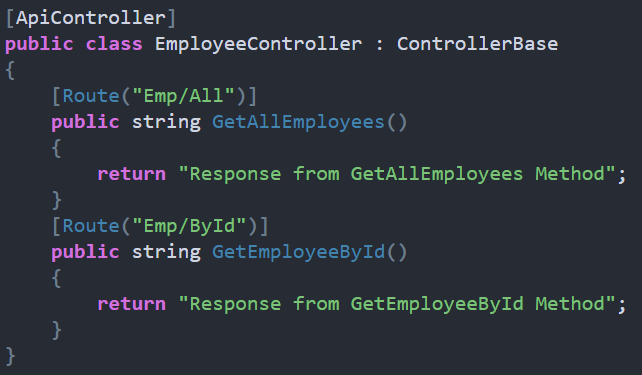
**Enabling Routing in ASP.NET Core Web API Application:**



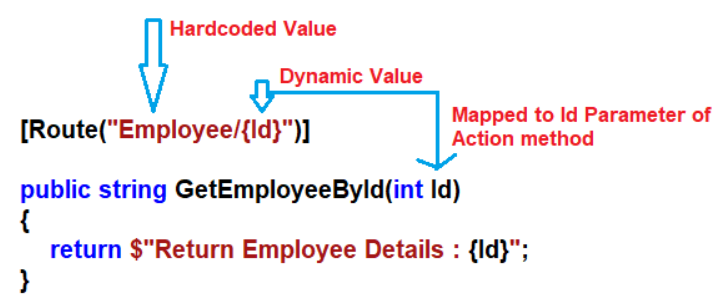
**Adding Attribute Routing in ASP.NET Core Web Application:**

URL’s:

1. 
2. 



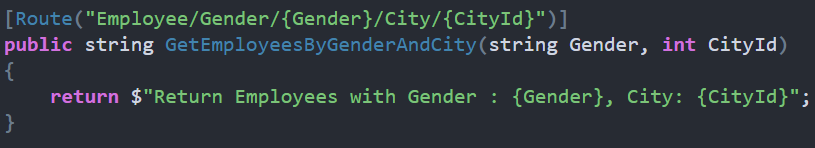
**Variables and Query Strings**



URL to call:



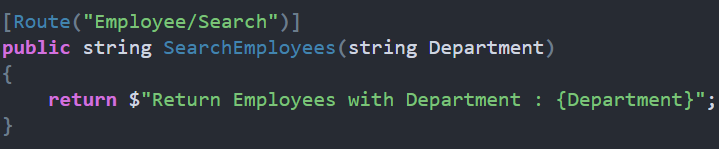
**Passing multiple dynamic values:**



URL to call:



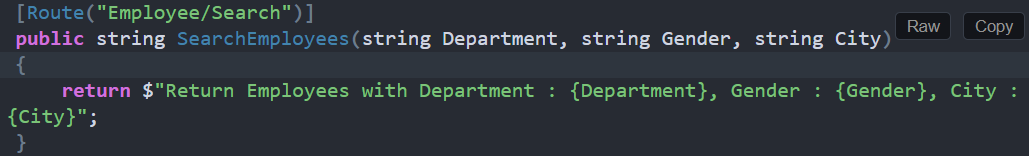
**Query Strings:**



URL to call:

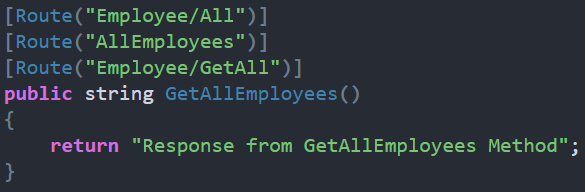


Multiple query string:



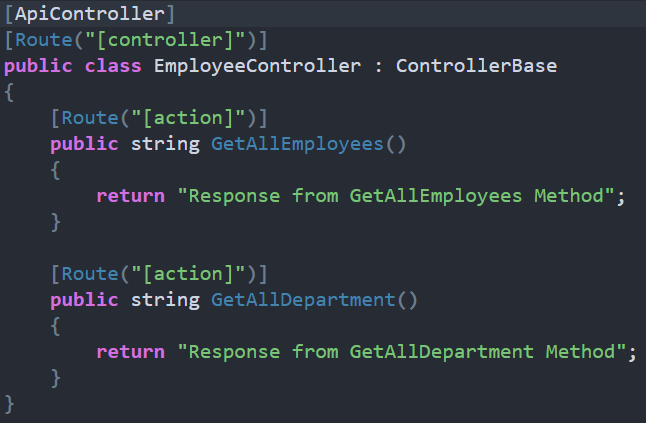
URL to call: 

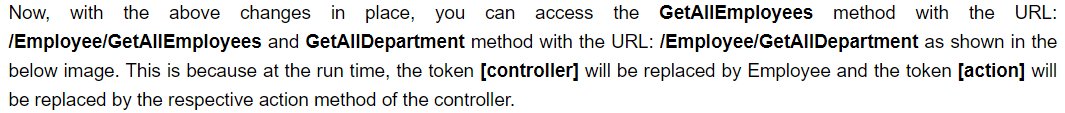
**Multiple URL for single resource:**

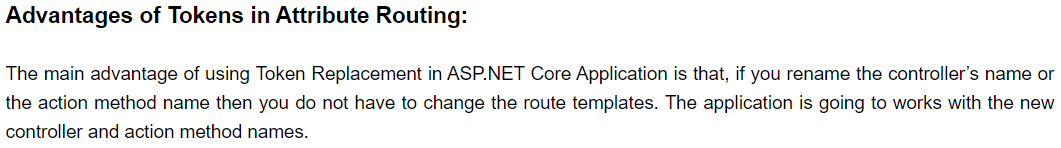


**Token Replacement:**

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.

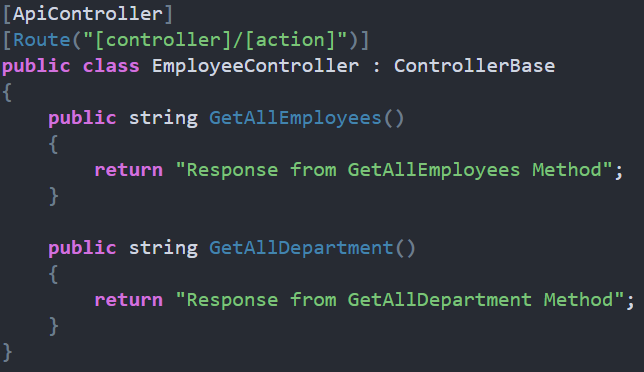




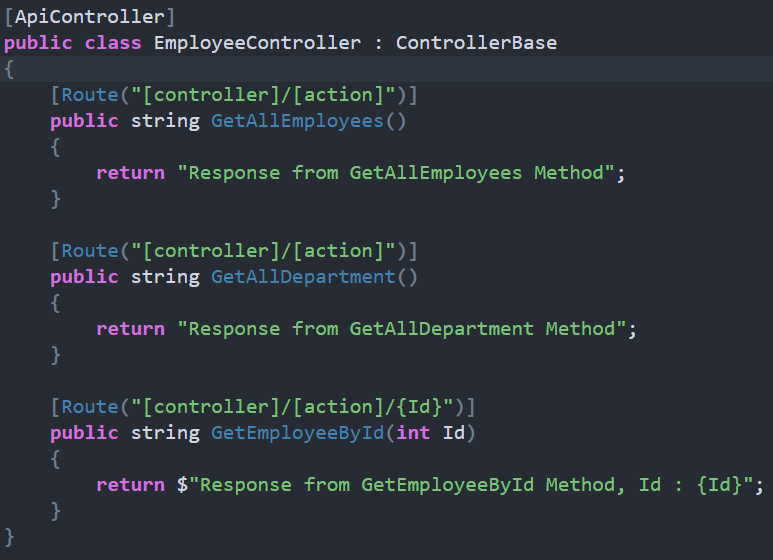


*Do we need to write token [action] on all action method?*

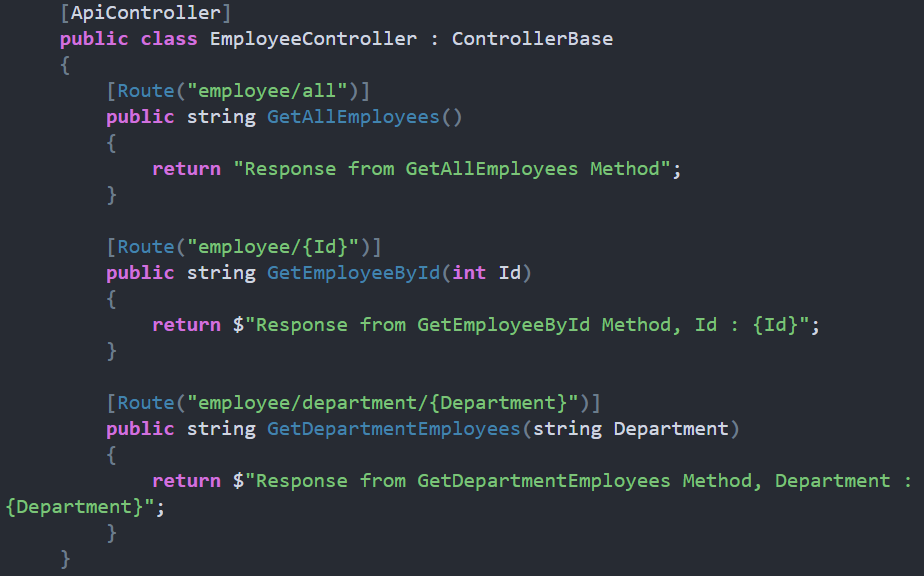
No need, then instead of including the [action] token on each action method, you can apply on top of controller at once.



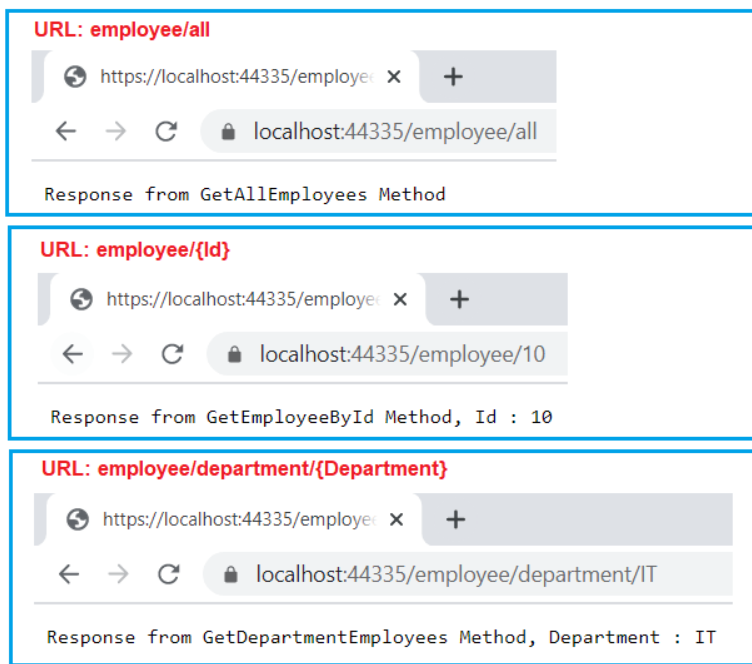
*Controller and action token applied to action method:*



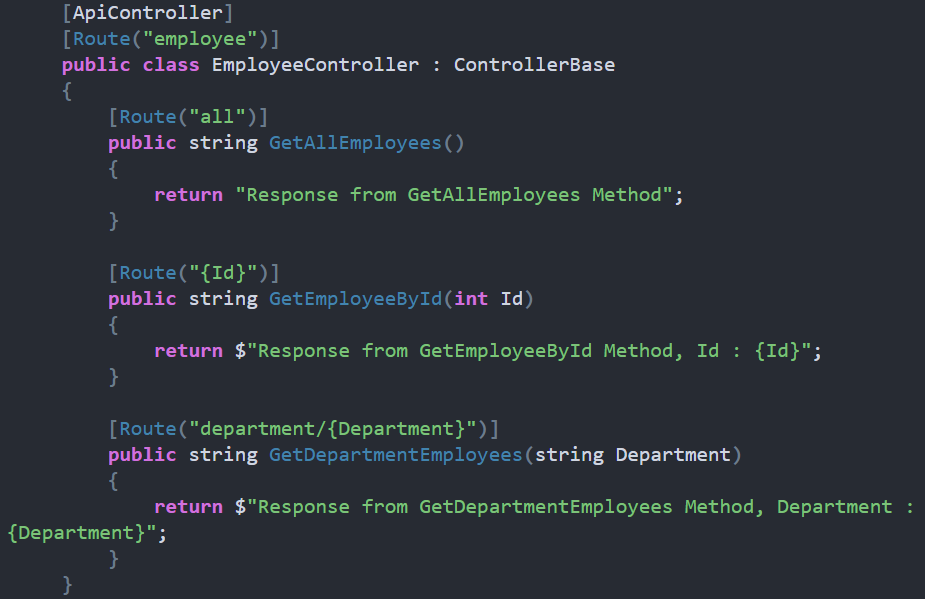
**Base Route**



I can able to access above action method as shown below:

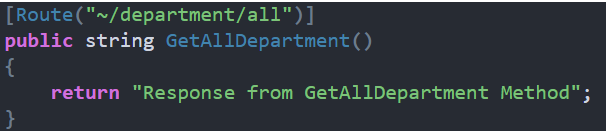


Now, “employee” is common prefix for all action methods, lets make it common at controller level.



*How to override the controller level route attribute at the action method level?*

Just add the “~” symbol.

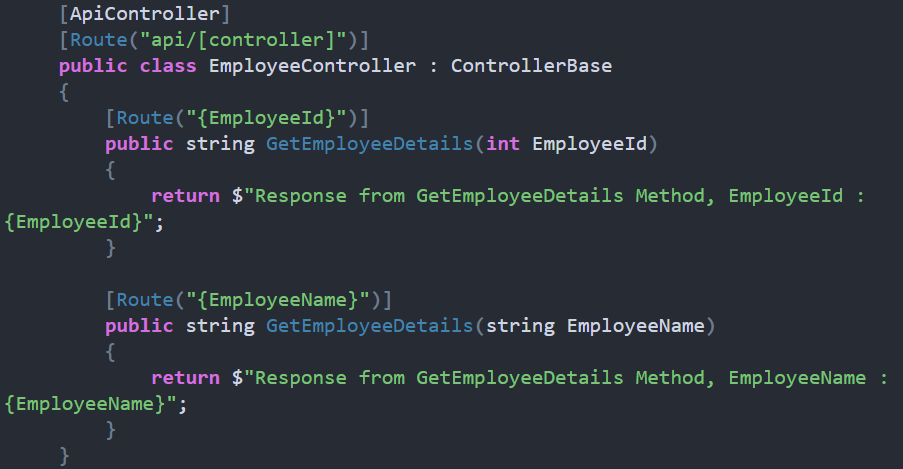


URL to call:



**Route Constraints**

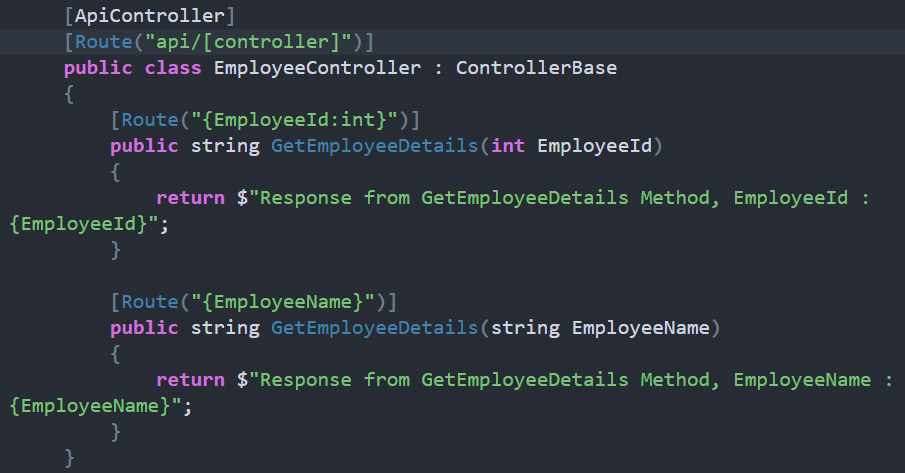
It is just a set of rules that can be applied to routing parameters to restrict how the parameters in the route template are matched. The syntax to use Route Constraints is: {parameter:constraint}.



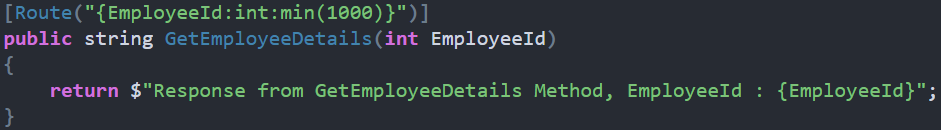
now the run application and navigate to the URL api/employee/10 and api/employee/smith, and in both cases you will get the error.

This is because, when the request comes, the application does not identify which version of the **GetEmployeeDetails()** method to use, and hence it gives an **Ambiguous** Match Exception as two end pints match the same request. This is the situation where the Route Constraints come into the picture in ASP.NET Core Web API.

*Solutions*



*Min(number) constraint:*

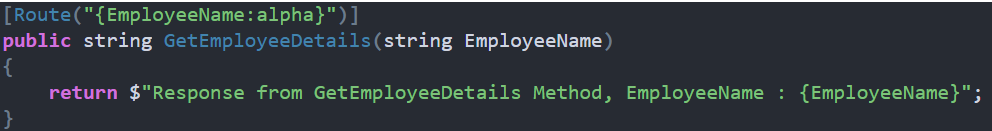


URL:

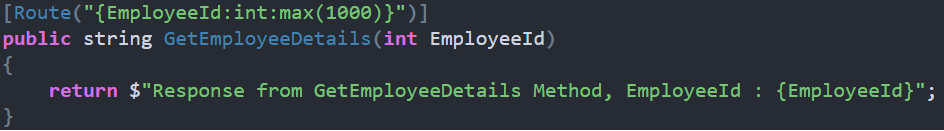


*Alpha constraint:*

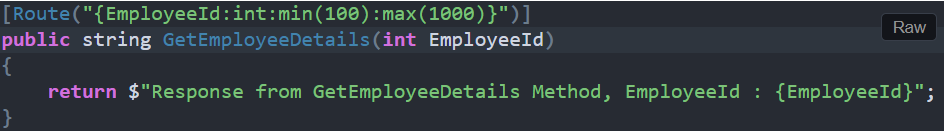
If you want any parameter to accept only alphabets (a to z characters) values then you need to specify the alpha constraint.



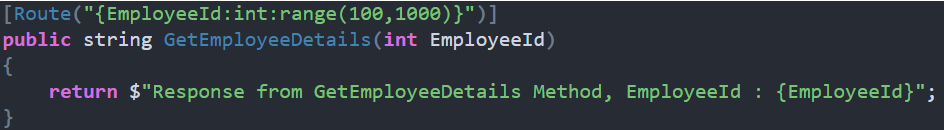
*Max(number) constraint:*



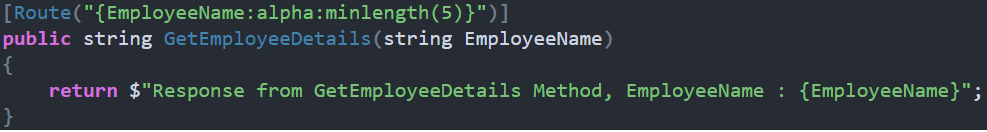
*Both Min and Max constraint together:*



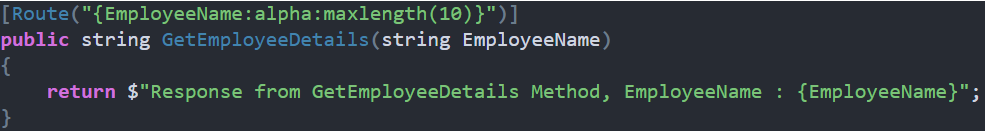
*Range constraint:*



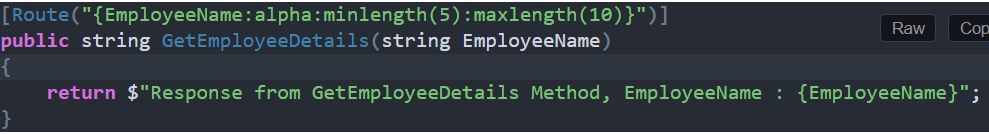
*MinLength constraint:*



*MaxLength constraint:*

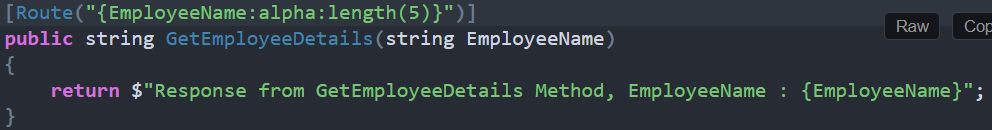


*Both MinLength and MaxLength constraint:*



*Length constraint:*

Used to specify the exact length of a string.



*Regex constraint:*

