# **Creating a Sample Web API**

Ref: <https://www.codeguru.com/csharp/.net/using-custom-action-names-in-asp.net-web-api.htm>

Ref: <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-3.1>

Ref: <https://www.tutorialsteacher.com/webapi/web-api-routing>

namespace WebAPICustomNamesDemo.Controllers

{

public class CustomerController : ApiController

{

NorthwindEntities db = new NorthwindEntities();

public IEnumerable<Customer> Get()

{

var data = from c in db.Customers

orderby c.CustomerID

select c;

return data.ToList();

}

public Customer Get(string id)

{

var data = from c in db.Customers

where c.CustomerID==id

select c;

return data.SingleOrDefault();

}

}

}

----------------------------------------------------------

//select all

var options = {};

options.url = "/api/customer";

options.type = "GET";

options.contentType = "application/json";

$.ajax(options);

Notice that the above code uses URL as /api/customer.

And depending on the HTTP method (GET, POST, etc.) the appropriate Web API action method is invoked.

--------------------------------------------------------

# **Specifying Action Name Using [ActionName] Attribute:**

Above shows, depending on the HTTP method, Web API framework automatically invokes the appropriate action method.

However, if you wish, you can configure the default routing mechanism to explicitly include the action method name.

namespace WebAPICustomNamesDemo.Controllers

{

public class CustomerController : ApiController

{

NorthwindEntities db = new NorthwindEntities();

[ActionName("SelectAll")]

public IEnumerable<Customer> Get()

{

...

}

[ActionName("SelectByID")]

public Customer Get(string id)

{

...

}

}

}

//select all

var options = {};

options.url = "/api/customer/SelectAll";

options.type = "GET";

options.contentType = "application/json";

$.ajax(options);

-------------------------------------------------------------------

# **Creating Action Methods by Prefixing Them with HTTP Method:**

However, these generic names may not convey the intended purpose of the respective method and you may want to make then more readable.

Luckily, there is a way to customize them without any additional configuration.

Web API framework will invoke the appropriate action method as long as your action method names begin with the HTTP method.

For example, instead of Get() method you can use GetCustomers() to serve the GET requests.

namespace WebAPICustomNamesDemo.Controllers

{

public class CustomerController : ApiController

{

NorthwindEntities db = new NorthwindEntities();

public IEnumerable<Customer> GetCustomers()

{

...

}

public Customer GetCustomerByID(string id)

{

...

}

}

}

//select all

var options = {};

options.url = "/api/customer";

options.type = "GET";

options.contentType = "application/json";

$.ajax(options);

---------------------------------------------------------------

# **Using Attributes to Specify HTTP Method and Action Method Mapping:**

You can change the Web API action method names to anything you want and use certain attributes to map them with the HTTP methods. This technique gives you total control on the action method names.

The attributes that you use are [HttpGet], [HttpPost], [HttpPut] and [HttpDelete].

As you might have guessed, these attributes correspond to GET, POST, PUT and DELETE HTTP methods.

The following code shows how to use these attributes:

namespace WebAPICustomNamesDemo.Controllers

{

public class Customer3Controller : ApiController

{

NorthwindEntities db = new NorthwindEntities();

[HttpGet]

public IEnumerable<Customer> SelectAllCustomers()

{

...

}

[HttpGet]

public Customer SelectCustomerByID(string id)

{

...

}

}

}

//select all

var options = {};

options.url = "/api/customer";

options.type = "GET";

options.contentType = "application/json";

$.ajax(options);

---------------------------------------------------------------

# **Using Attribute Routing:**

With attribute routing the controller name and action names play **no** role in which action is selected.

The Route attribute can be applied on any controller or action method.

This example would be executed for any of the following URL paths:

* /Home/Index/17
* /Home/Index
* /Home
* /

public class MyDemoController : Controller

{

[Route("")]

[Route("Home")]

[Route("Home/Index")]

public IActionResult MyIndex()

{

return View("Index");

}

[Route("Home/About")]

public IActionResult MyAbout()

{

return View("About");

}

[Route("Home/Contact")]

public IActionResult MyContact()

{

return View("Contact");

}

}

**One more example:**

public class StudentController : ApiController

{

[Route("api/student/names")]

public IEnumerable<string> Get()

{

return new string[] { "student1", "student2" };

}

}

In the above example, the Route attribute defines new route "api/student/names" which will be handled by the Get() action method of StudentController.

Thus, an HTTP GET request <http://localhost:1234/api/student/names>  will return list of student names.