Action Method Return type:

1. Void
2. Primitive and complex type
3. HttpResponseMessage
4. IHttpActionResult

Void: Mostly delete will not return anything

public class StudentController : ApiController

{

public void Delete(int id)

{

DeleteStudentFromDB(id);

}

}

It will send 204 “No content” status code.

Primitive or complex type:

public Student GetStudent(int id)

{

var student = GetStudentFromDB(id);

return student;

}

Graphical user interface, text, application, email

Description automatically generated

HttpResponseMessage:

The advantage of HttpResponseMessage is we can configure the response as a status code or a error message or as a content

public HttpResponseMessage Get(int id)

{

Student stud = GetStudentFromDB(id);

if (stud == null) {

return Request.CreateResponse(HttpStatusCode.NotFound, id);

}

return Request.CreateResponse(HttpStatusCode.OK, stud);

}

HTTP 404 Not Found status code, otherwise it will return 200 OK status with student data.

IHttpActionResult :

It returns an object that implements the IHttpActionResult

public IHttpActionResult Get(int id)

{

Student stud = GetStudentFromDB(id);

if (stud == null)

{

return NotFound();

}

return Ok(stud);

}

The following table lists all the methods of ApiController class that returns an object of a class that implements IHttpActionResult interface.

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

[FromUri] and [FromBody] :

These attributes are used to change the default behaviour of parameters.

Ie primitive as request body and complex as query string

Eg : public student Post ([FromUri]Student stud]

Localhost://api/student?id=1&name=steve

Public student Post([FromBody]string name)

Request Body

{

Name: ‘Steve’

}

Parameter Binding :

Binding the query string or request body to the parameters of the action method

The parameters can be primitive or complex type

Mostly complex type will be in the request body

Primitive type with query string : Get

<http://localhost/api/student?id=1>

http://localhost/api/student?ID=1

public Student Get(int id)

<http://localhost/api/student?id=1&name=steve>

http://localhost/api/student?id=1&name=steve

http://localhost/api/student?ID=1&NAME=steve

http://localhost/api/student?name=steve&id=1

public Student Get(int id, string name)

Post Action method with primitive paramater

Public Student POST(int id, string name)

http://localhost/api/student?id=1&name=steve

Post with complex parameter:

public Student Post(Student stud)

Request Body:

{

Id:1,

Name:’steve’

}

This also case insensitive.

Api extracts the json object from the request body , since its matching the student object parameters.

Post action method can have both primitive and complex parameters;

In this case the primitive api gets from the query string and the complex type from the request body.

In Post more than one complex parameters not allowed in one request.

Put and Patch are same as post.

Routing:

Its same as MVC routing

The “MapHttpAttributeRoute will enable attribute routing

The MapHttpRoute will enable the conventional routing.

<https://www.tutorialsteacher.com/webapi/web-api-routing> (refer)

If the webapi framework doesn’t find any matching route for the incoming request then it will raise 404 error response

Attribute Routing:

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

http://localhost:1234/api/student/names

The HttpConfiguaration class can be override to change the default behaviour of the api.

<https://www.tutorialsteacher.com/webapi/configure-web-api> (refer the property description table)

Using these property we can change the default behaviour of the web api.

public class WebAPIApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

//other configuration

}

}

WebApiConfig

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

// configure additional webapi settings here..

}

}

Return data automatically formatted based on Accept-type header attribute.

Patch()  
patch()  
PATCH()  
PatchStudent()  
\*any name starting with Patch\*

Patch will update records partially.

[HttpPost]

If the action method name is not starting with any http verbs then we can use http verb attributes.

GET/POST/PUT/PATCH/DELETE

The web api controller should be derived from Sysyetm.Web.Http.ApiController

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Web API configuration and services

// Web API routes

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

}

WebApiConfig.cs is the config file for api as in route config file for mvc.

This is find in app\_start folder.

https://www.tutorialsteacher.com/webapi/create-web-api-project

https://www.tutorialsteacher.com/webapi/consuming-web-api-in-dotnet-using-httpclient

Web api http response for all the uncaught exception:

500 – internal server error

Which .net framework support webapi

4.0 and above

Return types of webapi :

Void

HttpResponseMessage

HttpActionResult

Other types

Self hosting :

Can host web api as a separate process, than hosting in iis server

Advantages of webapi :

Content negotiation,

Self hosting

Filters

Routing

Model Binding

Why Api :

Used to create simple non SOAP based http services.

Easy to create , expose and consume.

Work fine with low bandwidth devices like mobile phone , iot

Advantages of Web api :

It increases TDD approach

Work with low bandwidth

Less config settings needed, uri template, contract and endpoint.

Webapi :

MS framework for creating http services

https://www.guru99.com/asp-net-web-api-interview-questions.html