(T1)討論RestfulAsp.NetApi的Get、Post、Put、Delete  
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd  
=======================================================================  
(T1)討論RestfulAsp.NetApi的Get、Post、Put、Delete

(T1-1)討論RestfulAsp.NetApi的Get、Post、Put、Delete的觀念 (1.)

(T1-2)討論RestfulAsp.NetApi的Get (2. to 4.)

(T1-3)用Postman測試RestfulAsp.NetApi的Get、Post、Put、Delete (5.1.)

(T1-4)用Fiddler測試RestfulAsp.NetApi的Get、Post、Put、Delete (5.2.)  
=======================================================================  
1. RESTful ASP.NET Web API

1.1. Introduction

1.2. REST constraints

1.3. Choice between WCF and RESTful ASP.NET Web API

1.4. Some Terms

-----------

2. OnlineGame Solution

2.1. OnlineGame Solution

2.2. OnlineGame.WebApi

-----------

3. OnlineGame.WebApi

-----------

4. OnlineGame.WebApi

4.1. Controllers/ValuesController.cs - Get

-----------

5. Controllers/Values2Controller.cs

5.1. Controllers/Values2Controller.cs - Postman

5.2. Controllers/Values2Controller.cs - Fiddler  
=======================================================================

1. RESTful ASP.NET Web API

討論

什麼事RESTful ASP.NET Web API

建立第一個API with Get、Post、Put、Delete

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

The tutorial will discuss

What is RESTful ASP.NET Web API

Create your first API with Get、Post、Put、Delete

1.1. Introduction

ASP.NET Web API is a framework to build a RESTful or non-RESTful API (Application Programming Interface). The tutorial will discuss how to build RESTful (Representational State Transfer) API services by using ASP.NET Web API framework which can be consumed by browsers, mobile applications, desktop applications ...etc.  REST is just a structure pattern that uses HTTP as its communication method.  We have to follow some REST constraints to build our API to fulfill this "RESTful" API structure.

1.2. REST constraints

1.

Client-Server constraint

Client-side logic always sends a request and server-side logic always sends a response so that Client-side logic can develop their evolution.

2.

Stateless constraint

Server-side logic does not store any client-side data or state, so Client-side request must contain all the necessary information for the server-side to response the request independently.

3.

Cacheable constraint

Some data does not change very often, so the server-side logic should let client-side logic know how long this data is still good.  Thus, client-side logic can cache the data based on that period.

4.

Uniform Interface

Uniform Interface is a HATEOAS (Hypermedia as the Engine of Application State) pattern between the client and the server; it means each request from the client will be a set of hyperlinks that tell the server what to do with the resource entity.  Resource entity can be a data row which represents a person or a product, and it is identified by a specific URI (Uniform Resource Identifier).  The HTTP verb (GET, PUT, POST, DELETE) tells the API what to do with the resource entity.

E.g. 1.

Resource: /Gamers

HTTP verb: GET

Gets a list of gamers

E.g. 2.

Resource: /Gamer/1

HTTP verb: GET

Gets the gamer with Id = 1

E.g. 3.

Resource: /Gamers

HTTP verb: POST

Create a new gamer to the list of gamers

E.g. 4.

Resource: /Gamer/1

HTTP verb: PUT

Update the gamer with Id = 1

E.g. 5.

Resource: /Gamer/1

HTTP verb: DELETE

Delete the gamer with Id = 1

5.

Layered System

6.

The code on Demand (optional)

1.3. Choice between WCF and RESTful ASP.NET Web API

WCF (Windows Communication Foundation) is suitable for building services that are transport/protocol independent between your server and each client.  It requires a lot of complicated configuration to achieve RESTful services.  WCF uses SOAP service to transfer data, and the SOAP makes the data is too heavy to transfer to another side.

Web API use JSON to transfer data and JSON is much lighter than SOAP.  JSON might use a few bytes, but SOAP might use 1k to transfer the same data.  It is the reason why Web API become more popular than WCF recently.

The relationship between WCF and Web API is just like the relationship between Web Form and Web MVC.  They both exist because of different reasons.

I suggest that you use WCF only if you stuck with existing SOAP service.

If you have no limitation, RESTful ASP.NET Web API might be a better choice

1.4. Some Terms

Reference:

Http verbs and their status code

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

**Request Verbs :**

4 Database actions include Create, Read, Update, Delete (CRUD) that maps to 4 HTTP verbs, POST, GET, PUT, DELETE.  Each Http verbs have their status code.

For complete Http verbs and status code, please see <https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

**Request Header/Body V.S. Response Body:**

The client sends a request that includes a header and a body for the server to give back a response body.

The header contains some extra information tell what format of my request body and what format of response body from the server.

Request body contains all necessary information for the server to produce response body.

E.g. 1.

The client sends a GET request header that said my request body is in XML/JSON format; please give me the data by a response body in XML/JSON format.

E.g. 2.

The client sends a POST request header that said my request body is in XML/JSON format; please insert this new item.

**Response HTTP** **Status codes :**

E.g.

Server return status code, **200/OK, 404/Not Found, 204/No Content**, to the clients.

2. OnlineGame Solution

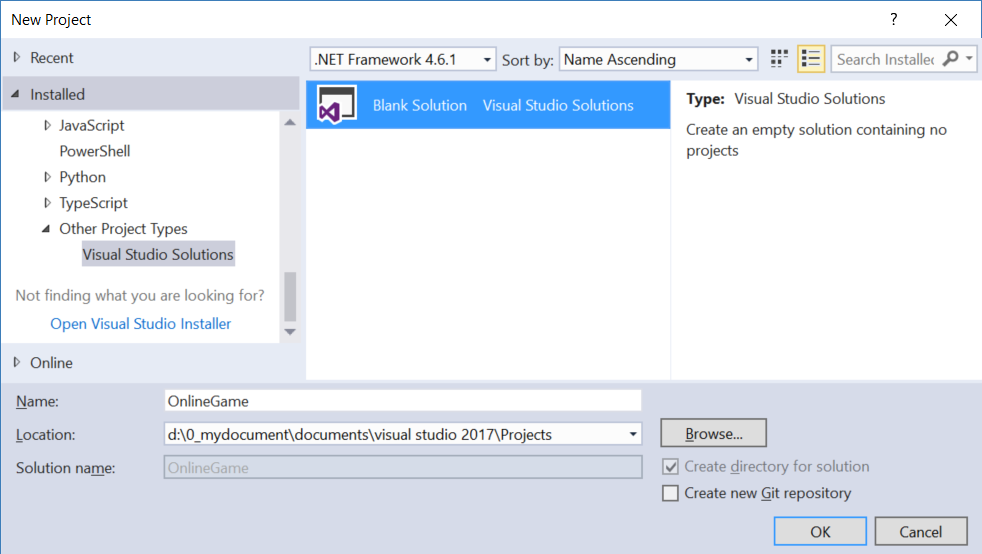
2.1. OnlineGame Solution

File --> New --> Project... -->

Other Project Types --> Visual Studio Solutions -->  Blank Solution

-->

Name: **OnlineGame**



2.2. OnlineGame.WebApi

Solutions Name --> Add --> New Project -->

Visual C# --> Web --> [ASP.NET](http://asp.net/)Web Application (.Net Framework)

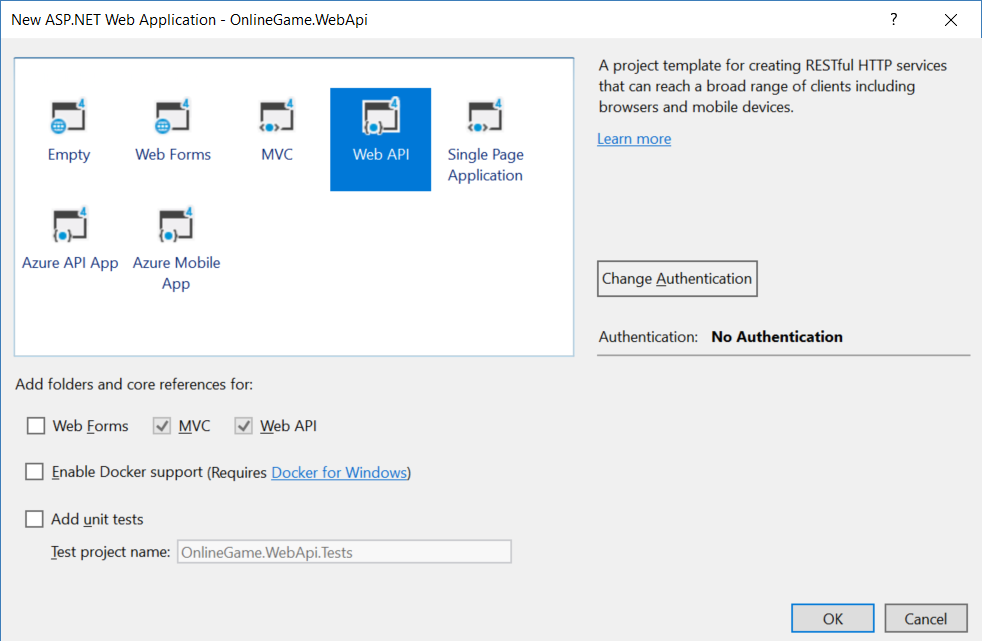
-->

Name: **OnlineGame.WebApi**

--> Select "**Web API**" --> OK

Graphical user interface, application

Description automatically generated



3. OnlineGame.WebApi

1.

In Controllers/HomeController.cs

public class HomeController : System.Web.Mvc.Controller

This is a MVC controller.

2.

In Controllers/ValuesController.cs

public class ValuesController : System.Web.Http.ApiController

This is a web API controller, and it has 2 Get, 1 Post, 1 Put, 1 Delete.

It maps to the HTTP verbs (GET, PUT, POST, DELETE) respectively.

using System.Collections.Generic;

using System.Web.Http;

namespace OnlineGame.WebApi.Controllers

{

    public class ValuesController : System.Web.Http.ApiController

    {

        // GET api/values

        //Get a list of value

        public IEnumerable<string> Get()

        {

            return new string[] { "value1", "value2" };

        }

        // GET api/values/5

        //Get the value with id==5

        public string Get(int id)

        {

            return "value";

        }

        // POST api/values

        //insert a new value

        public void Post([FromBody]string value)

        {

        }

        // PUT api/values/5

        //update the value with id==5

        public void Put(int id, [FromBody]string value)

        {

        }

        // DELETE api/values/5

        //delete the value with id==5

        public void Delete(int id)

        {

        }

    }

}

3.

In Global.asax

using System.Web.Http;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

namespace OnlineGame.WebApi

{

    public class WebApiApplication : System.Web.HttpApplication

    {

        protected void Application\_Start()

        {

            AreaRegistration.RegisterAllAreas();

            //Register API route in WebApiConfig class Register method.

            GlobalConfiguration.Configure(WebApiConfig.Register);

            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

            //Register MVC route in RouteConfig class RegisterRoutes method.

            RouteConfig.RegisterRoutes(RouteTable.Routes);

            BundleConfig.RegisterBundles(BundleTable.Bundles);

        }

    }

}

4.

In Controllers/ValuesController.cs

using System.Collections.Generic;

using System.Web.Http;

namespace OnlineGame.WebApi.Controllers

{

    public class ValuesController : System.Web.Http.ApiController

    {

        // GET api/values

        //Get a list of value

        public IEnumerable<string> Get()

        {

            return new string[] { "value1", "value2" };

        }

        // GET api/values/5

        //Get the value with id==1

        public string Get(int id)

        {

            return "value";

        }

        // POST api/values

        //insert a new value

        public void Post([FromBody]string value)

        {

        }

        // PUT api/values/5

        //update the value with id==5

        public void Put(int id, [FromBody]string value)

        {

        }

        // DELETE api/values/5

        //delete the value with id==5

        public void Delete(int id)

        {

        }

    }

}

5.

In App\_Start/RouteConfig.cs

using System.Web.Mvc;

using System.Web.Routing;

namespace OnlineGame.WebApi

{

    public class RouteConfig

    {

        public static void RegisterRoutes(RouteCollection routes)

        {

            routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

            routes.MapRoute(

                name: "Default",

                url: "{controller}/{action}/{id}",

                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

            );

        }

    }

}

6.

In App\_Start/WebApiConfig.cs

using System.Web.Http;

namespace OnlineGame.WebApi

{

    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 }

            );

        }

    }

}

4. OnlineGame.WebApi

4.1. Controllers/ValuesController.cs - Get

using System.Collections.Generic;

using System.Web.Http;

namespace OnlineGame.WebApi.Controllers

{

    public class ValuesController : System.Web.Http.ApiController

    {

        // GET api/values

        //Get a list of value

        public IEnumerable<string> Get()

        {

            return new string[] { "value1", "value2" };

        }

        // GET api/values/5

        //Get the value with id==1

        public string Get(int id)

        {

            return "value";

        }

        // POST api/values

        //insert a new value

        public void Post([FromBody]string value)

        {

        }

        // PUT api/values/5

        //update the value with id==5

        public void Put(int id, [FromBody]string value)

        {

        }

        // DELETE api/values/5

        //delete the value with id==5

        public void Delete(int id)

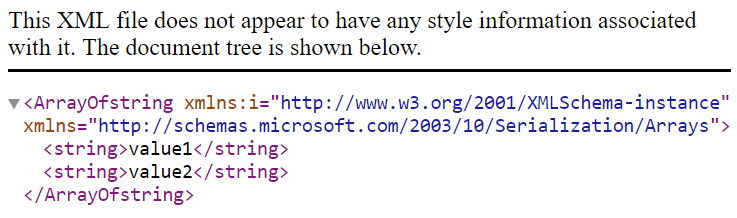
        {

        }

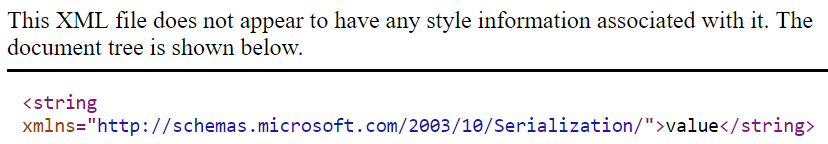
    }

}

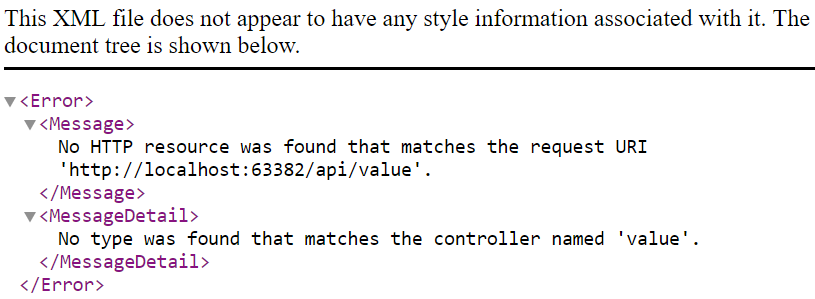
<http://localhost:63382/api/values>



<http://localhost:63382/api/values/2>



<http://localhost:63382/api/value>



5. Controllers/Values2Controller.cs

using System.Collections.Generic;

using System.Web.Http;

namespace OnlineGame.WebApi.Controllers

{

    public class ValuesController2 : System.Web.Http.ApiController

    {

        static List<string> \_valueStrList = new List<string>

        {

            "value0", "value1", "value2"

        };

        // GET api/values

        //Get a list of value

        public IEnumerable<string> Get()

        {

            return \_valueStrList;

        }

        // GET api/values/5

        //Get the value with id==1

        public string Get(int id)

        {

            return \_valueStrList[id];

        }

        // POST api/values

        //insert a new value

        public void Post([FromBody]string value)

        {

            \_valueStrList.Add(value);

        }

        // PUT api/values/5

        //update the value with id==5

        public void Put(int id, [FromBody]string value)

        {

            \_valueStrList[id] = value;

        }

        // DELETE api/values/5

        //delete the value with id==5

        public void Delete(int id)

        {

            \_valueStrList.RemoveAt(id);

        }

    }

}

5.1. Controllers/Values2Controller.cs - Postman

The postman has windows version and chrome version.

I like chrome version because you can keep your setting record on all your computers.

GET

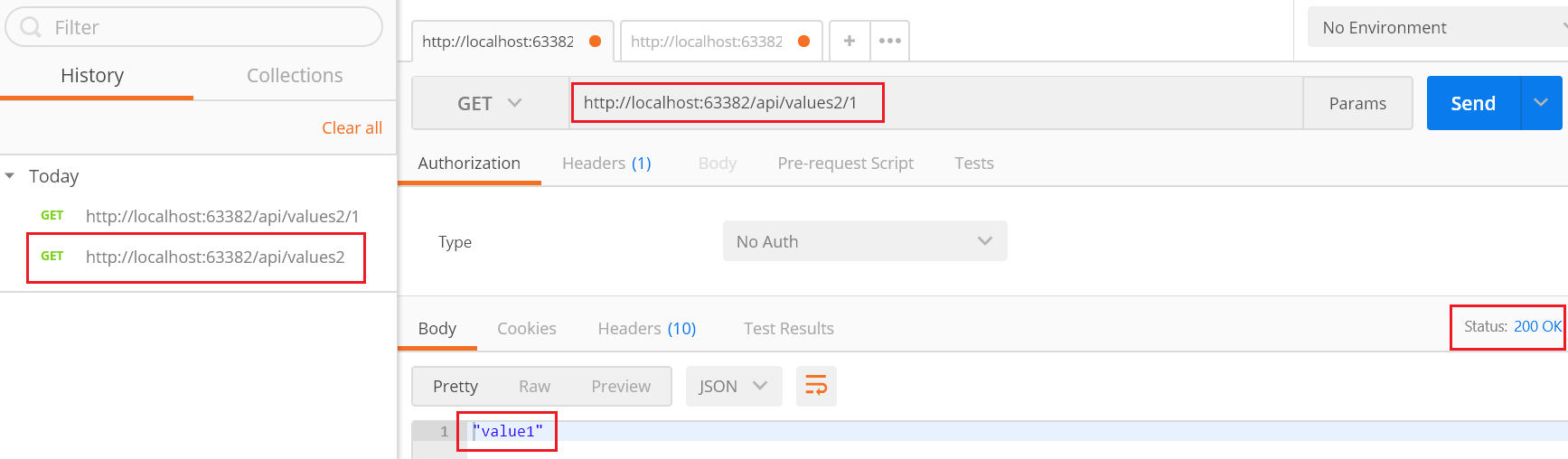
<http://localhost:63382/api/values2>

Graphical user interface, application

Description automatically generated

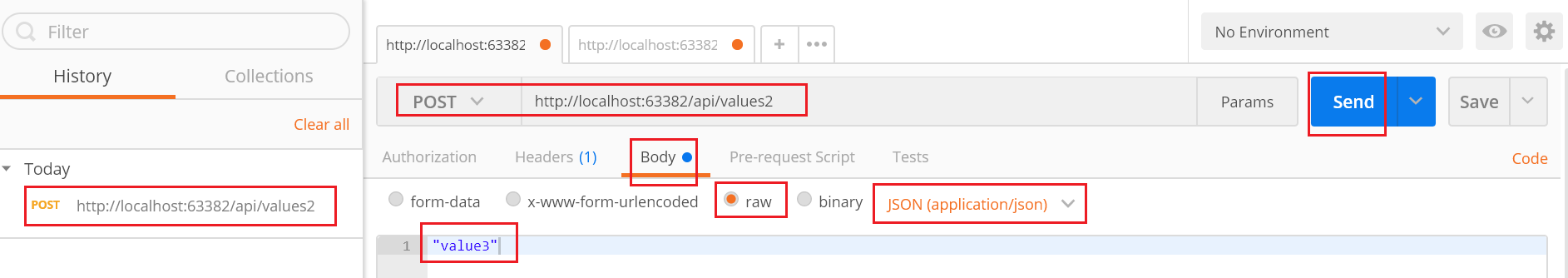
GET

<http://localhost:63382/api/values2/1>



POST

<http://localhost:63382/api/values2>



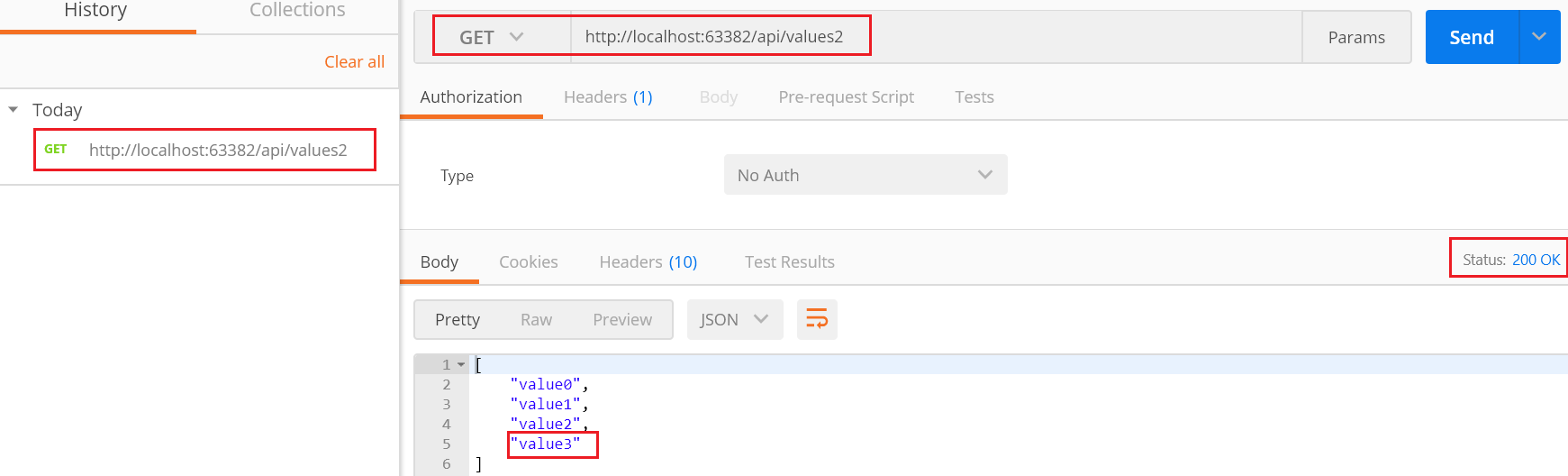
-->



204 No Content, because it is a "void" method.

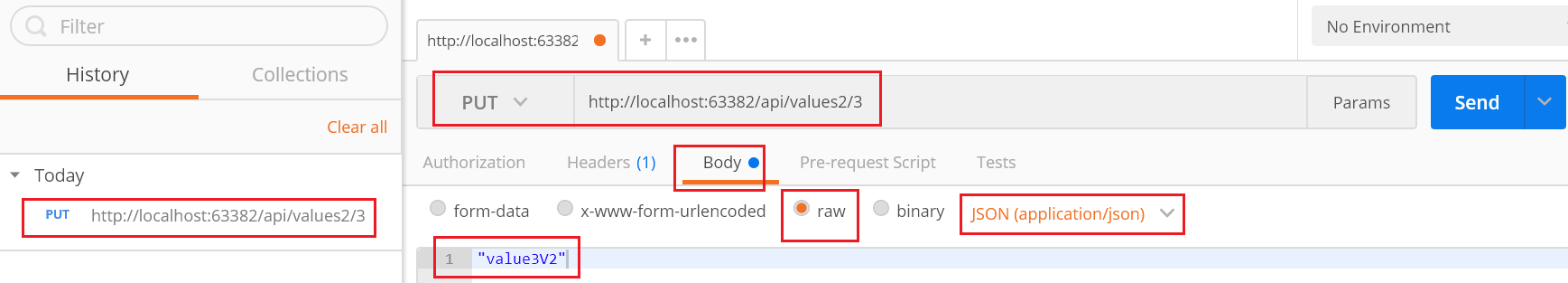
GET

<http://localhost:63382/api/values2>



PUT

<http://localhost:63382/api/values2/3>



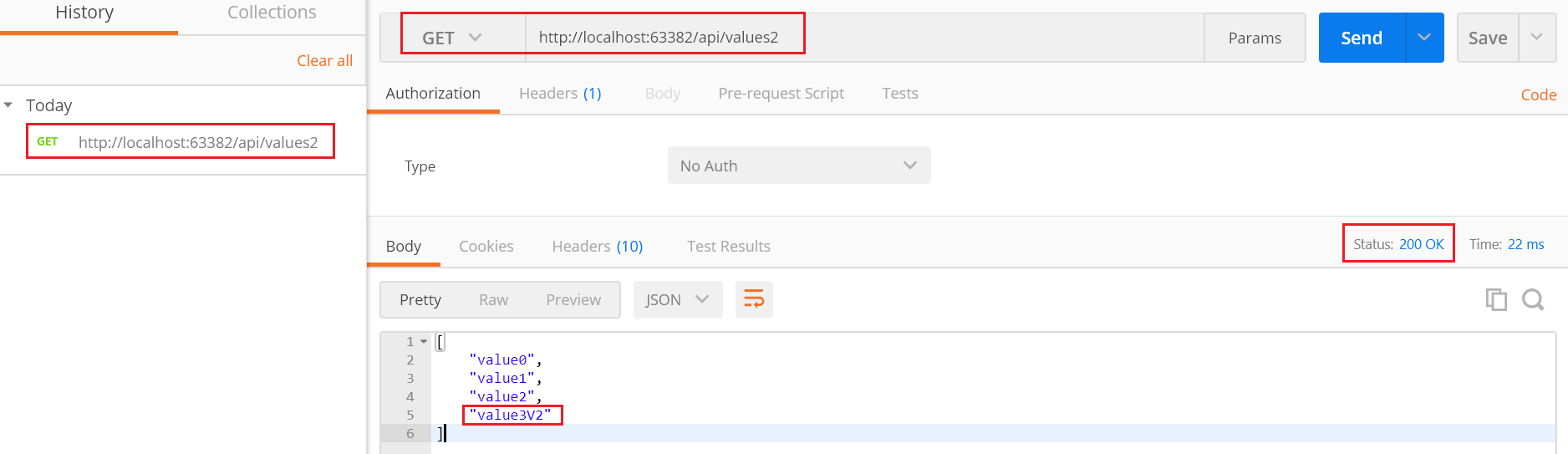
-->



204 No Content, because it is a "void" method.

GET

<http://localhost:63382/api/values2>



DELETE

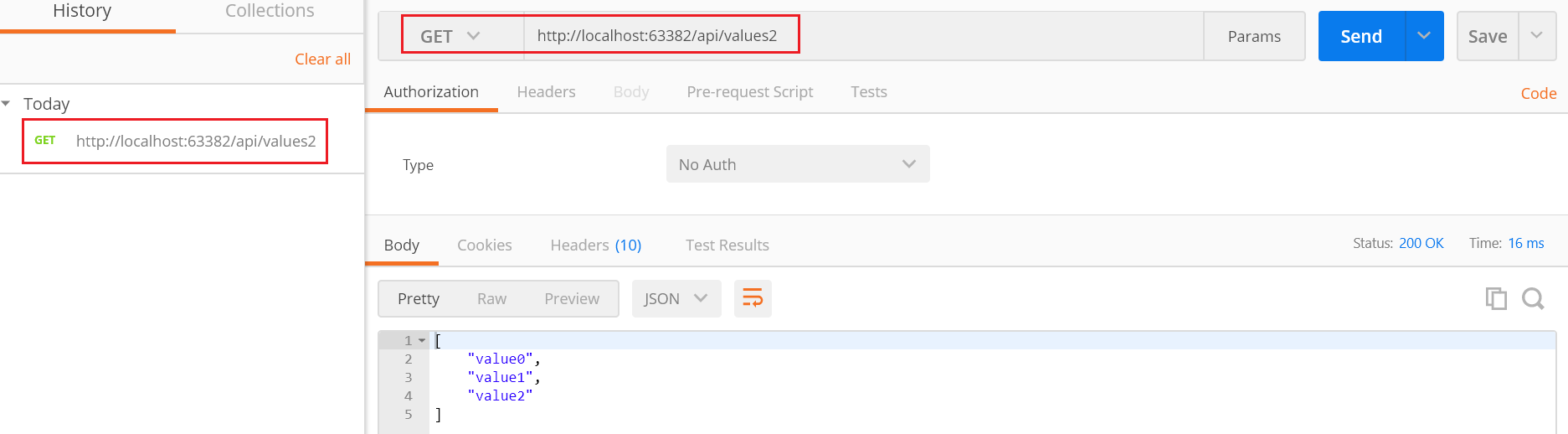
<http://localhost:63382/api/values2/3>

Graphical user interface, application

Description automatically generated

GET

<http://localhost:63382/api/values2>



5.2. Controllers/Values2Controller.cs - Fiddler

Fiddler is lovely software; it displays more information than the Postman.

The Postman has a very user-friendly user interface, and easy to use without learning more knowledge.

Fiddler is a bit more complicated than the Postman, but it shows more information.

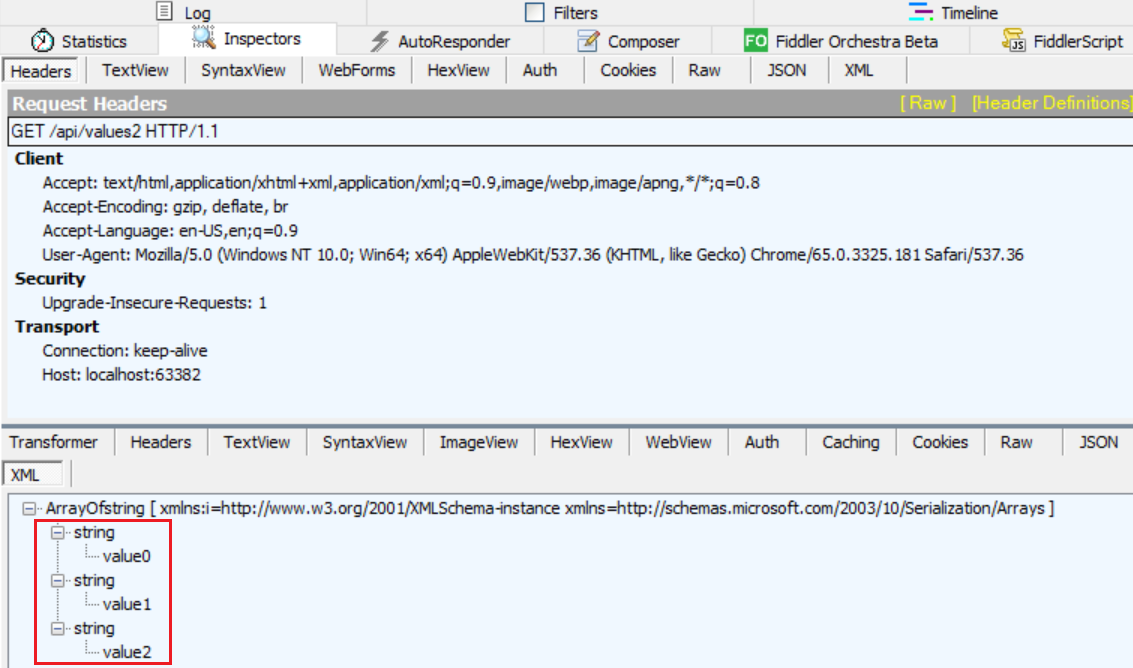
I use fiddler more often than using postman.

GET

<http://localhost:63382/api/values2>



-->

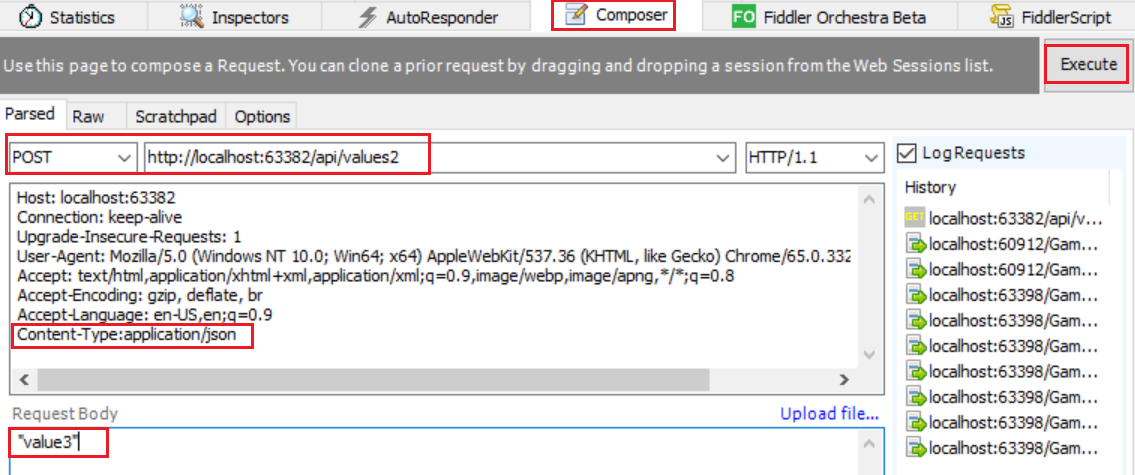


POST

<http://localhost:63382/api/values2>

**Content-Type:application/json**

**"value3"**

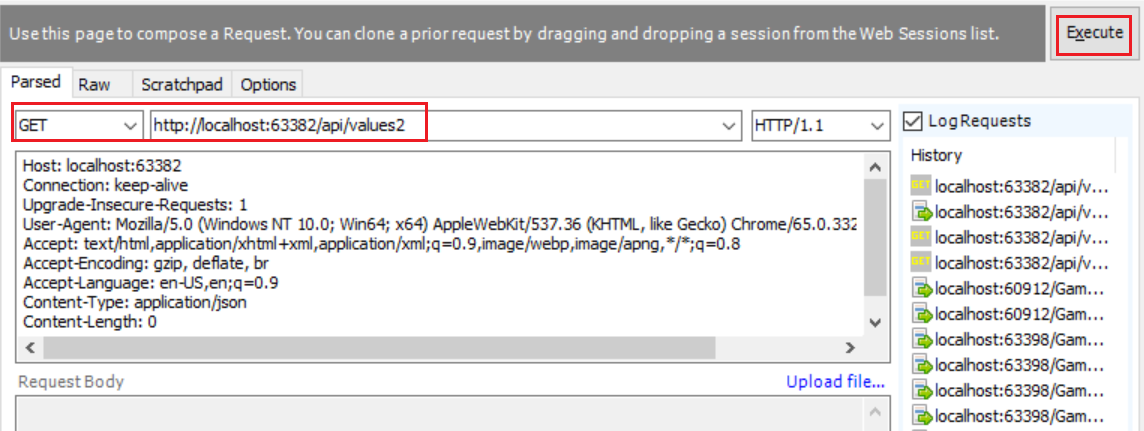


-->



GET

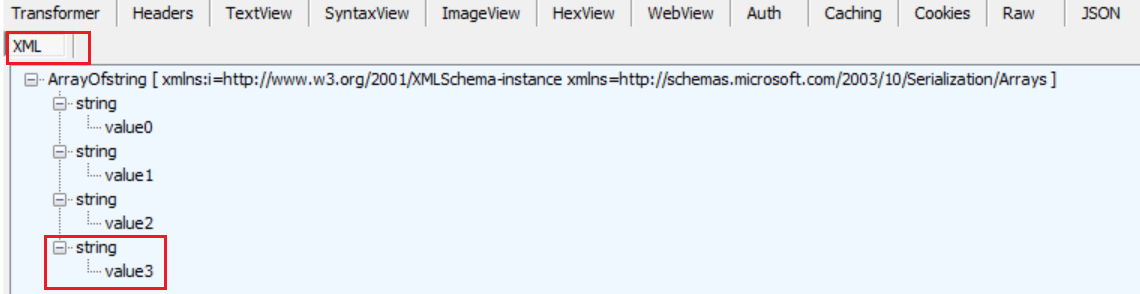
<http://localhost:63382/api/values2>



-->



-->

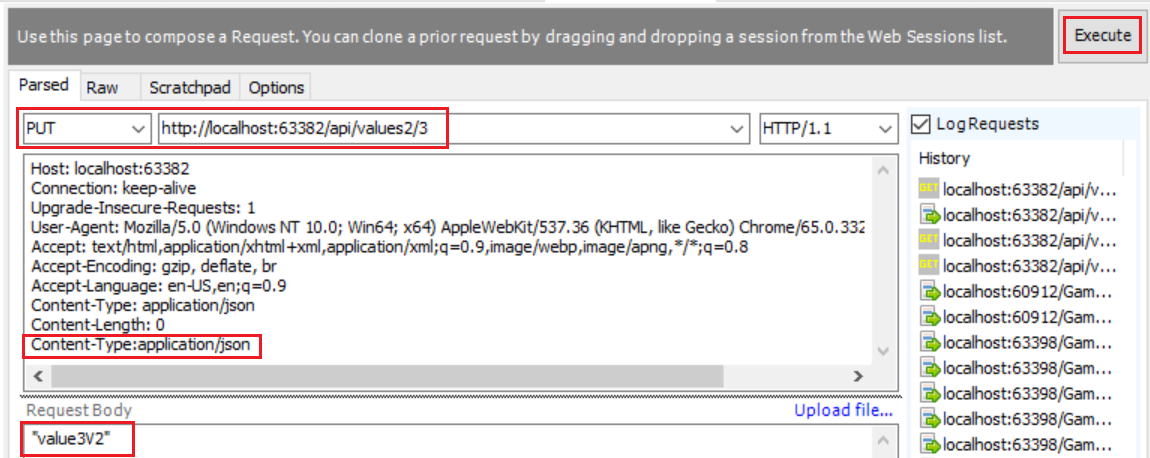


PUT

[http://localhost:63382/api/**values2/3**](http://localhost:63382/api/values2)

**Content-Type:application/json**

**"value3V2"**



-->



GET

[http://localhost:63382/api/**values2/2**](http://localhost:63382/api/values2)

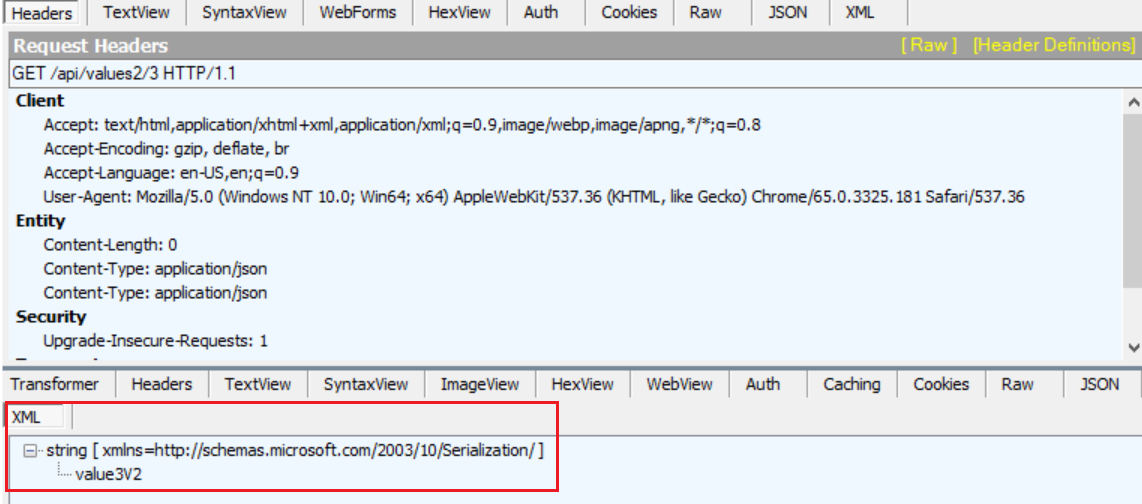
Graphical user interface, text, application, email

Description automatically generated

-->

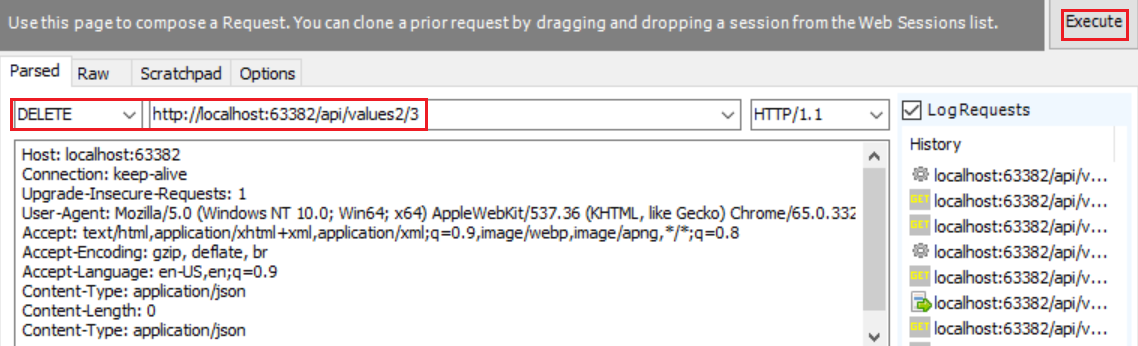


-->



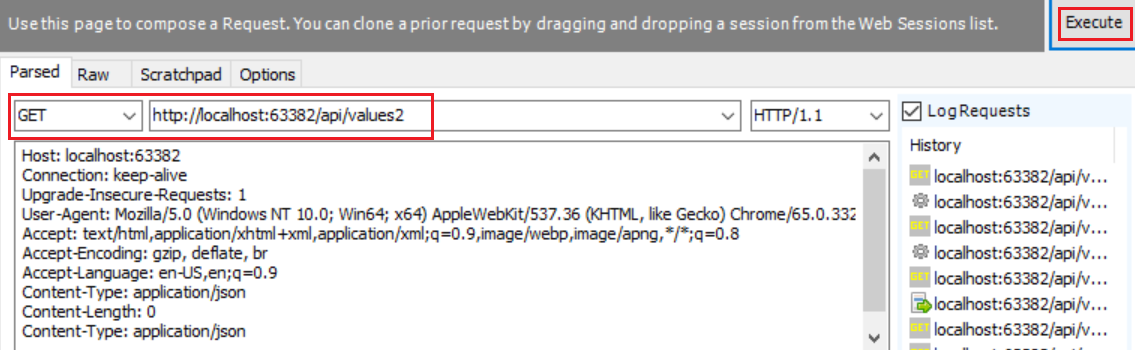
DELETE

[http://localhost:63382/api/**values2/3**](http://localhost:63382/api/values2/3)



GET

<http://localhost:63382/api/values2>



-->



-->

