Back End Development 1
sesi 8

Understanding ASP.NET

Understanding ASP.NET - Sesi 08

Introduction Web Services

Sebelum masuk ke pembahasan utama ada baiknya kita mengenal apa itu WEB SERVICES.

Aplikasi web yang mempunyai sifat untuk dapat diakses di mana saja dan kapan saja.

Membuat banyak aplikasi-aplikasi banyak bermunculan dan dimanfaatkan oleh orang banyak. Baik itu aplikasi yang berfungsi membantu untuk pekerjaan perkantoran atau aplikasi yang bersifat hiburan, jejaring sosial dan lain-lain.

Pada sesi kali ini telah diterangkan beberapa hal untuk berkenalan dengan ASP.NET Web API. Hal-hal yang telah dibahas pada bagian ini adalah :

1. Pembuatan aplikasi web ASP.NET Web API dengan menggunakan template project yang telah disediakan pada Visual Studio.
2. Pembuatan class controller Web API pada project yang telah ada sebelumnya. Pembuatan ini juga disertai dengan penjelasan pembuatan folder dan file pendukung serta konfigurasi yang harus dilakukan agar class controller tersebut dapat diakses dan digunakan.
3. Cara mengakses Web API yang telah dibuat.
4. Pembuatan halaman bantuan (help page) Web API yang berguna untuk melihat daftar class controller serta API yang terdapat didalamnya.
5. Installasi dan konfigurasi Tool test client yang sederhana yang berfungsi untuk mengakses dan menguji API yang telah

Untuk mengetahui bagaimana antara aplikasi web dan aplikasi web dapat saling mendukung satu sama lain, maka akan dipaparkan pada bagian dibawah ini.

Pertama akan dijelaskan bagaimana aplikasi web secara umum bekerja. Aplikasi web yang berada pada web server akan dapat diakses oleh komputer yang digunakan oleh pengguna dengan web browser seperti Internet Explorer.

Komunikasi yang terjadi antara web browser pada komputer dengan web server adalah dengan dua proses, yaitu request dan response.

Request adalah proses yang dilakukan oleh web browser pada komputer ke web server, kemudian permintaan tersebut akan diolah oleh web server dan hasilnya akan dikirimkan ke web browser yang merupakan proses response dari web server.

Hasil proses response ini lah yang akan kita lihat dalam bentuk halaman web pada web browser. Misal user ingin melihat data staff pada web browser, maka yang akan dilakukan adalah akan ada proses request ke web server.

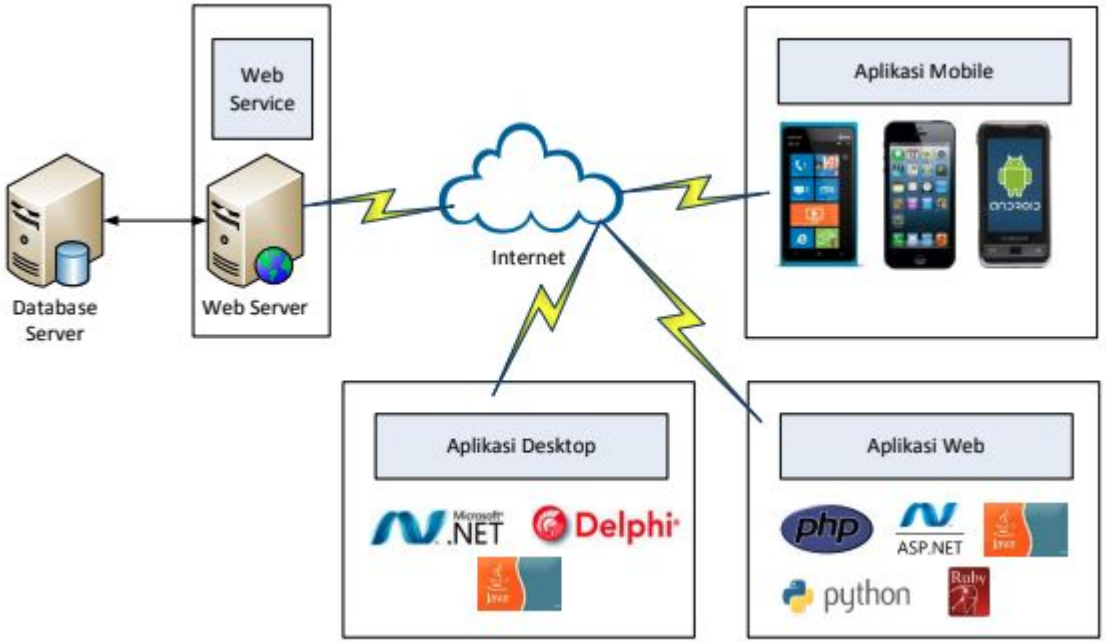
Permintaan tersebut akan diproses pada web server, karena ada kebutuhan akan data staff maka aplikasi web pada sisi server akan mengambil data staff tersebut pada database kemudian hasilkannya akan dikirimkan ke web browser sebagai proses response.

Keberadaan web service sebagai perantara seperti gambar di atas mempunyai beberapa manfaat, diantaranya adalah :

1. Layanan ini dapat lebih mengamankan data pada database server, hal ini dikarenakan database tidak langsung diakses oleh aplikasi.

2. Layanan ini bukan hanya dapat diakses oleh aplikasi mobile, tapi juga dapat diakses oleh aplikasi jenis lain seperti aplikasi web atau aplikasi desktop. Selain itu layanan berbasis web ini akan dapat diakses dan dimanfaatkan pada berbagai platform atau teknologi, artinya layanan ini bisa diakses oleh aplikasi yang dibangun dengan .NET, Java, PHP, Python, Ruby, Javascript dan lain-lain.

Dari penjelasan di atas maka dapat digambarkan peran dari web service seperti gambar berikut ini.



https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-5.0

Understanding ASP.NET - Sesi 08

Introduction ASP.NET

Merupakan kumpulan teknologi dalam Framework .Net untuk membantu pengembangan aplikasi web yang menggunakan Object — Oriented secara dinamis, teknologi yang diciptakan oleh Microsoft untuk pemrograman Internet yang lebih efisien.

ASP.Net dirilis pada bulan Januari 2002 pertama kalinya yang merupakan bagian dari Microsoft.Net Framework versi 1.0. Keunggulan ASP.Net pada pertama kali di rilis adalah :

1. Pada ASP klasik, code bercampur menjadi satu halaman dengan HTML, tetapi untuk versi terbaru terdapat pembagian yang jelas antara HTML dan code.
2. Visual Studio .NET yang memungkinkan pengembangan untuk membuat aplikasi web secara visual.
3. Visual Basic .Net dan C# merupakan bahasa yang dapat digunakan.

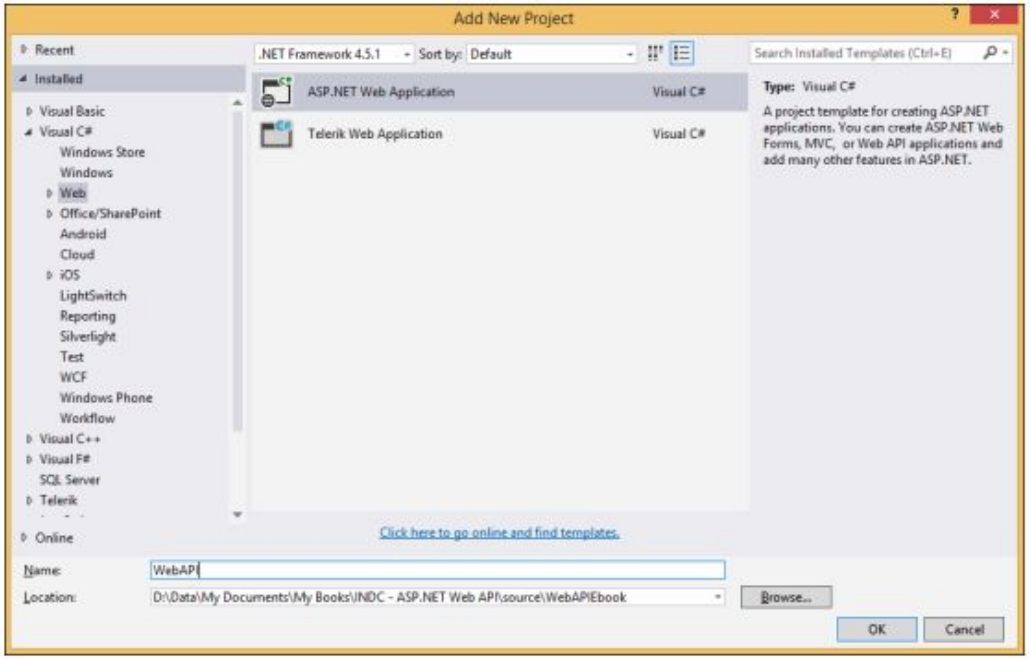
Nah untuk mengenal langsung dengan ASP kita akan membuat kode program langsung untuk project

1. Project WEB API
2. Penambahan Web API pada existing Project

PROJECT : WEB API

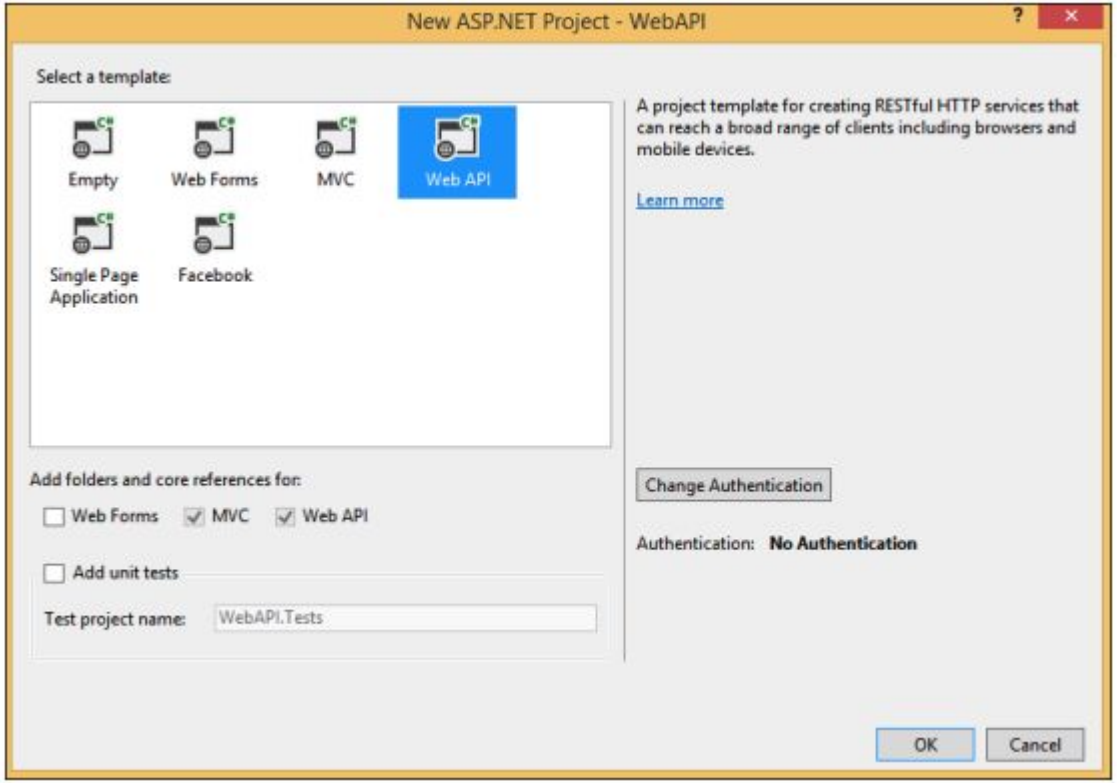
Berikut adalah langkah-langkah yang dilakukan untuk membuat project Web API. Langkah pertama adalah membuat project baru dengan cara klik File > New Project.

Pada window Add New Project pilih Visual C# > Web > ASP.NET Web Application.

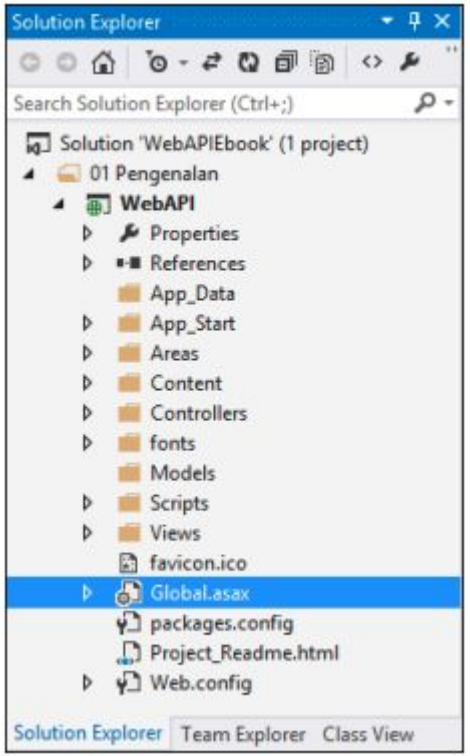


HACKTIV8

Tuliskan nama project pada kolom Name, kemudian klik tombol OK. Kemudian pada window New ASP.NET Project pilih Web API dan klik tombol OK.



Hasilnya dapat dilihat pada area Solution Explorer seperti berikut ini.



Pada project ini selain terdapat class-class dan file-file utama Web API juga terdapat file-file pendukung seperti halaman web sebagai landing page dan help. Untuk mengaktifkan web server dari project ini dan melihat pada web browser dengan cara klik kanan pada project WebAPI yang ada pada Solution Explorer kemudian pilih Debug > Start new instance.

Dengan cara di atas dapat dilihat pada project sudah terdapat banyak file-file yang mungkin belum atau tidak diperlukan.

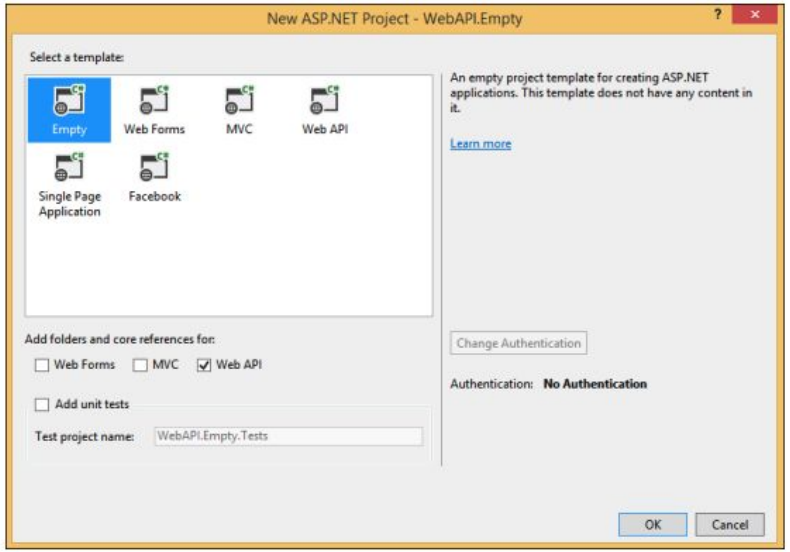
Jika ingin membuat project kosong untuk implementasi Web API maka dapat dilakukan langkah-langkah berikut ini.

Pilih File > New Project kemudian pada window Add New Project pilih Visual C# > Web > ASP.NET Web Application.

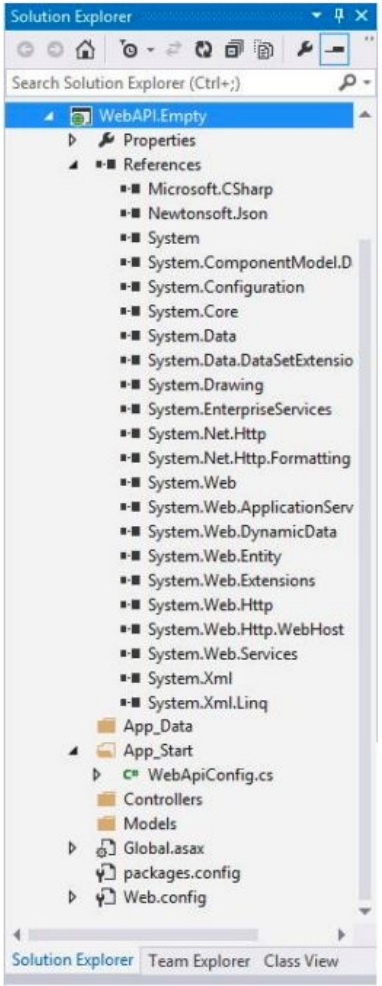
Kemudian pada window New ASP.NET Project pilih Empty

pada area Select a template dan centang Web API pada area Add folder and core reference for.

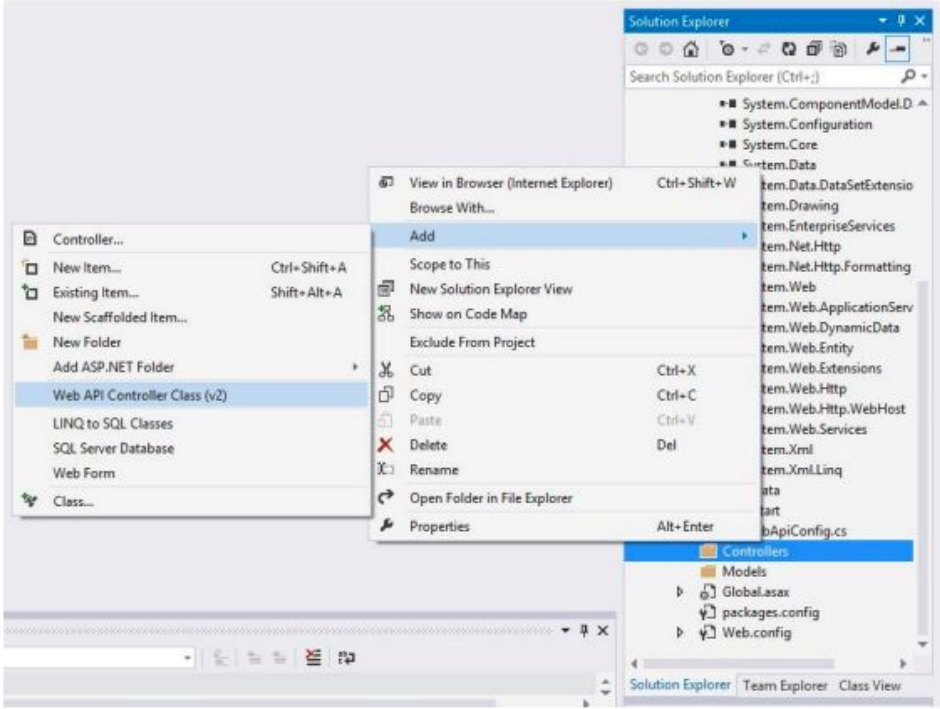
Selanjutnya klik tombol OK.



HACKTIV8


Solution Explorer
Search Solution Explorer (Ctrl+;)
WebAPI.Empty
Properties
References
Microsoft.CSharp
Newtonsoft.Json
System
System.ComponentModel.D
System.Configuration
System.Core
System.Data
System.Data.DataSetExtensio
System.Drawing
System.EnterpriseServices
System.Net.Http
System.Net.Http.Formatting
System.Web
System.Web.ApplicationServ
System.Web.DynamicData
System.Web.Entity
System.Web.Extensions
System.Web.Http
System.Web.Http.WebHost
System.Web.Services
System.Xml
System.Xml.Linq
App_Data
App_Start
WebApiConfig.cs
Controllers
Models
Global.asax
packages.config
Web.config
Solution Explorer
Team Explorer
Class View

Langkah selanjutnya adalah menambahkan class Web API controller pada project tersebut dengan cara : Klik kanan pada folder Controllers kemudian pilih Add > Web API Controller Class (v2) seperti yang terlihat pada gambar di bawah ini.



Dan beri nama ValuesController sebagai nama class tersebut.

HACKTIV8

Dari kedua hal yang telah dilakukan di atas dapat dilihat beberapa kesamaan berupa file-file sebagai berikut :

1. WebApiConfig.cs yang terdapat pada folder App_Start.
2. Global.asax yang terdapat pada /.
3. ValuesController.cs yang terdapat pada folder Controllers.

File WebApiConfig.cs berfungsi sebagai konfigurasi Web API routes yang menentukan bagaimana format url untuk mengakses class-class controller.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace WebAPI.Empty
{
    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

Pembahasan :

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

dari format tersebut maka controller dapat diakses dengan cara menulis format url sebagai berikut :

```
http://[nama_domain]/api/[nama_controller]/
```

Jika kode pada WebApiConfig.cs mempunyai baris sebagai berikut :

```
routeTemplate: "webapi/{controller}/{id}"
```

Maka format url untuk mengakses class controller akan menjadi sebagai berikut :

```
http://[nama_domain]/webapi/[nama_controller]/
```

File kedua yang telah disebutkan adalah `Global.asax` dengan isi sebagai berikut :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Routing;

namespace WebAPI.Empty
{
    public class WebApiApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            GlobalConfiguration.Configure(WebApiConfig.Register);
        }
    }
}
```

`Global.asax`

File ini berfungsi untuk memanggil dan mengaktifkan konfigurasi Web API pada class `WebApiConfig.cs` saat aplikasi web dijalankan. Pada kedua project di atas nama class controller yang telah dibuat adalah `ValuesController.cs` maka dapat ditulis url sebagai berikut :

```
http://[nama_domain]/api/Values/
```

Sedangkan jika nama class controller adalah `CalculateController.cs` maka class ini akan dapat diakses dengan cara menulis url berikut :

```
http://[nama_domain]/api/Calculate/
```

Dari contoh tersebut maka dapat disimpulkan nama class controller akan mempunyai format sebagai berikut

```
[nama_controller]Controller.cs
```

ASP.NET WEB FORM WEBSITE

1. ASP.NET WEB FORM WEBSITE

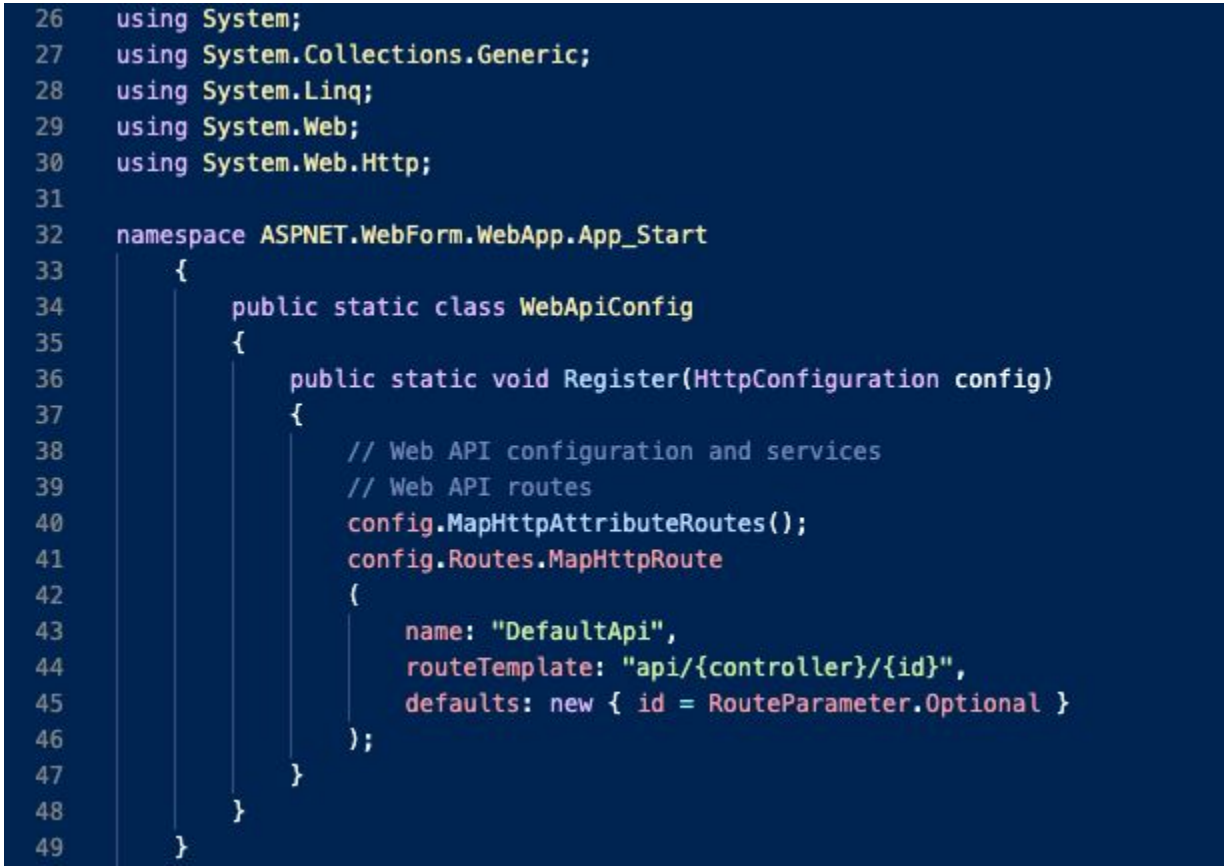
Dari penjelasan pada bagian sebelumnya maka jika ingin menambahkan class Web API Controller pada project web yang telah ada, yaitu dengan cara menambahkan tiga file dan dilakukan konfigurasi seperti yang telah disebutkan di atas.

Pada sesi ini akan diterangkan langkah-langkah untuk menambahkan Web API, dalam hal ini adalah class Web API controller pada existing project web dengan tipe sebagai berikut :

ASP.NET Web Form Website.

Pada project Web Application secara default sudah memiliki file Global.asax sehingga yang perlu dilakukan adalah :

1. Menambahkan folder App_Start.
2. Menambahkan file WebApiConfig.cs ke dalam folder App_Start dengan isi seperti yang telah dijelaskan pada awal sesi.
3. Menambahkan baris untuk mengaktifkan konfigurasi WebApiConfig.cs ke dalam file Global.asax.
4. Menambahkan folder Controllers.
5. Menambahkan class Web API Controller pada folder Controllers, pada saat melakukan proses ini secara otomatis akan ditambahkan reference yang berhubungan dengan Web API seperti System.Web.Http dan lain-lain.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;

namespace ASPNET.WebForm.WebApp.App_Start
    {
        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

```
20  using System;
21  using System.Collections.Generic;
22  using System.Linq;
23  using System.Web;
24  using System.Web.Http;
25  using System.Web.Security;
26  using System.Web.SessionState;
27  using ASPNET.WebForm.WebApp.App_Start;
28
29  namespace ASPNET.WebForm.WebApp
30  {
31      public class Global : System.Web.HttpApplication
32      {
33          protected void Application_Start(object sender, EventArgs e)
34          {
35              GlobalConfiguration.Configure(WebApiConfig.Register);
36          }
37      }
38  }
```

Global.asax

ValuesController.cs

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace ASPNET.WebForm.WebApp.Controllers
{
    public class ValuesController : ApiController
    {
        // GET api/<controller>
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET api/<controller>/5
        public string Get(int id)
        {
            return "value";
        }

        // POST api/<controller>
        public void Post([FromBody]string value)
        {
        }

        // PUT api/<controller>/5
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE api/<controller>/5
        public void Delete(int id)
        {
        }
    }
}
```

ASP.NET MVC WEB APPLICATION

2. ASP.NET MVC WEB APPLICATION

Aplikasi web yang dibangun dengan ASP.NET MVC dibuat sebagai project Web Application seperti halnya yang telah dijelaskan pada bagian ASP.NET Web Form Web Application di atas, maka cara untuk menambahkan kemampuan Web API pada project ini mempunyai langkah-langkah yang sama seperti cara di atas.

Yang perlu diperhatikan adalah urutan penulisan baris program pada file `Global.asax` di bawah ini. Baris `GlobalConfiguration.Configure(WebApiConfig.Register);` harus berada sebelum `RouteConfig.RegisterRoutes(RouteTable.Routes);`.

```
using ASPNET.MVC.WebApp.App_Start;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Routing;

namespace ASPNET.MVC.WebApp
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            GlobalConfiguration.Configure(WebApiConfig.Register);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

Global.asax

```
namespace ASPNET.MVC.WebApp.App_Start
    {
        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

ValuesController.cs

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace ASPNET.MVC.WebApp.Controllers
{
    public class ValuesController : ApiController
    {
        // GET api/<controller>
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET api/<controller>/5
        public string Get(int id)
        {
            return "value";
        }

        // POST api/<controller>
        public void Post([FromBody]string value)
        {
        }

        // PUT api/<controller>/5
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE api/<controller>/5
        public void Delete(int id)
        {
        }
    }
}
```

Pada contoh kode di atas terdapat 5 method yang akan menjadi API dan terdiri atas 4 kelompok berdasarkan HTTP verb yaitu :

1. GET, dalam penggunaannya method untuk kelompok HTTP verb ini adalah untuk mengambil atau membaca data. Method pada kelompok ini biasanya mengembalikan suatu keluaran/output yang kadang bisa disebut sebagai function.

2. POST, dalam penggunaannya method untuk kelompok HTTP verb ini adalah untuk membuat (create) item/resource baru. Kelompok method ini biasanya tidak mengembalikan keluaran/output yang kadang disebut procedure.

3. PUT, dalam penggunaannya method untuk kelompok HTTP verb ini adalah untukmengupdate item/resource yang telah ada. (procedure seperti pada point 2).

4. DELETE, dalam penggunaannya method untuk kelompok HTTP verb ini adalah untuk menghapus item/resource yang telah ada. (procedure seperti pada point 2). Untuk itu nama-nama method yang berada pada suatu class Web API controller akan sama dengan HTTP verb di atas yaitu Get, Post, Put dan Delete sebagai prefix dari nama method pada class controller.

Sebagai contoh jika nama method adalah GetProduct, GetProductById atau GetAll maka API dari method tersebut akan termasuk dalam kelompok HTTP verb GET. Sedangkan jika nama method adalah PostProduct maka API dari method tersebut akan termasuk dalam kelompok HTTP verb POST.

Cara lain untuk menentukan method ke dalam suatu HTTP verb bisa dengan cara memberikan atribute pada method tersebut. Sebagai contoh pada kode di bawah ini.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace ASPNET.WebForm.WebApp.Controllers
{
    public class ValuesController : ApiController
    {
        [HttpGet]
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        [HttpGet]
        public string Get(int id)
        {
            return "value";
        }

        [HttpPost]
        public void Post([FromBody]string value)
        {
        }

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

        [HttpDelete]
        public void Delete(int id)
        {
        }
    }
}
```

HACKTIV8

Dari kode di atas dapat dilihat penggunaan atribut `[HttpGet]`, `[HttpPost]`, `[HttpPut]` dan `[HttpDelete]` untuk menentukan HTTP verb untuk akses API dari suatu method.

Dengan penggunaan atribut ini maka nama method dapat bebas sesuai keinginan.

Hasilnya dari kode di atas dapat dilihat pada daftar API yang ada pada halaman bantuan seperti yang terlihat pada gambar di bawah ini :

Values

API	Description
GET api/Values	No documentation available.
GET api/Values/{id}	No documentation available.
POST api/Values	No documentation available.
PUT api/Values/{id}	No documentation available.
DELETE api/Values/{id}	No documentation available.

HACKTIV8

Akses Web API

Setelah penjelasan di atas, pada bagian ini akan dijelaskan bagaimana mengakses dan menguji Web API. Jika project Web API masih dalam tahap development di Visual Studio, maka pastikan terlebih dahulu telah project aplikasi web tersebut telah dijalankan dan IIS Express sebagai web server masih aktif.

Cara yang dapat dilakukan untuk melakukan hal ini adalah dengan klik kanan pada project kemudian pilih Debug > Start new instance.
Atau jika pada Website, klik kanan pada website kemudian pilih View in Browser.

Maka secara otomatis web browser akan dijalankan dan halaman default web pada project akan ditampilkan. Untuk mengakses API, cukup ketikkan pada address bar web browser dengan format seperti berikut :

```
http://[nama_domain]:port/api/[nama_controller]
```

Misal alamat akses project tersebut adalah `http://localhost:11739/` dan nama class controller adalah `ValuesController.cs` maka alamat yang diakses adalah :

```
http://localhost:11739/api/Values/
```

Tetapi akses via address bar pada web browser terbatas hanya untuk akses API yang bersifat GET, seperti contoh di atas. Untuk mengakses atau mencoba API yang bersifat POST, PUT atau DELETE tidak bisa dilakukan dengan cara menuliskan url pada address bar di web browser.

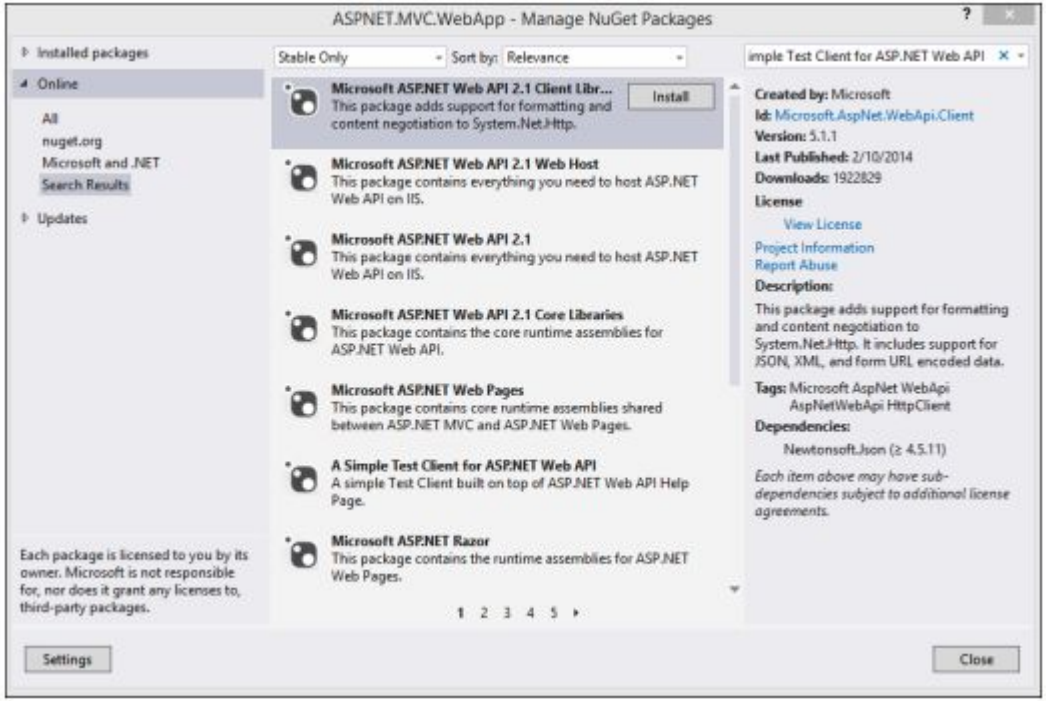
Untuk itu perlu bantuan tool client untuk membantu mengakses Web API.

A SIMPLE TEST CLIENT for ASP.NET WEB API

Tool ini adalah tool client untuk ASP.NET Web API yang melekat pada halaman bantuan Web API sebagai fitur tambahan. Tool ini hanya dapat digunakan untuk mengakses dan menguji Web API pada project tempat tool ini di install. Sehingga tidak bisa digunakan untuk menguji Web API lintas project atau lintas website.

Untuk menggunakan tool ini harus dipastikan pada project sudah memiliki halaman bantuan Web API seperti yang telah dibahas pada bagian Halaman Bantuan ASP.NET Web API diatas. Kemudian lakukan langkah-langkah berikut untuk menginstall tool ini.

Klik kanan pada project yang diinginkan, pilih Manage NuGet Packages,
kemudian ketikkan kata kunci A Simple Test Client for ASP.NET Web API pada kolom pencarian.
Maka akan dapat dilihat daftar berikut ini pada window Manage NuGet Packages.

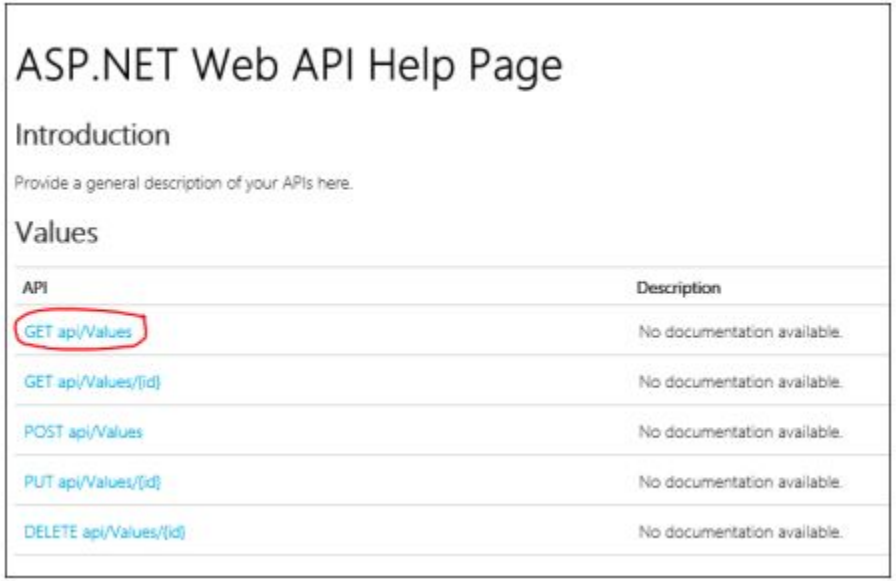


HACKTIV8

Kemudian klik install dan setelah proses pengunduhan dan installasi selesai, maka langkah selanjutnya adalah melakukan modifikasi file `Api.cshtml` yang terletak pada folder `Areas\HelpPage\Views\Help\` dengan kode seperti berikut ini.

```
@using System.Web.Http
@using WebAPI.Areas.HelpPage.Models
@model HelpPageApiModel

@{
    var description = Model.ApiDescription;
    ViewBag.Title = description.HttpMethod.Method + " " +
    description.RelativePath;
}

<div id="body">
    <section class="featured">
        <div class="content-wrapper">
            <p>
                @Html.ActionLink("Help Page Home", "Index")
            </p>
        </div>
    </section>

    <section class="content-wrapper main-content clear-fix">
        @Html.DisplayFor(m => Model)
    </section>
</div>

@Html.DisplayForModel("TestClientDialogs")
@section Scripts {
    @Html.DisplayForModel("TestClientReferences")
    <link type="text/css" href="~/Areas/HelpPage/HelpPage.css"
rel="stylesheet" />
}
```

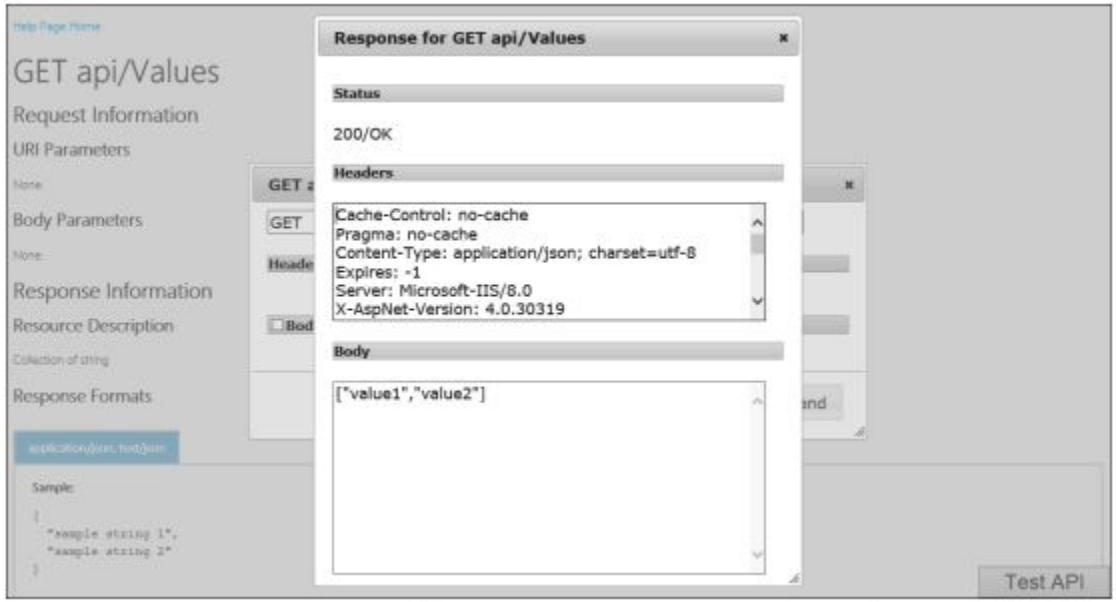
Baris yang ditambahkan pada file di atas adalah 6 baris terakhir tersebut. Hasil ini nanti akan dapat dilihat ketika mengakses halaman bantuan, lebih tepatnya ketika salah satu API yang terdapat pada daftar API yang dimiliki oleh controller di klik.




Help Page Home
GET api/Values
Request Information
URI Parameters
None.
Body Parameters
None.
Response Information
Resource Description
Collection of string
Response Formats
application/json, text/json
Sample:
Test API

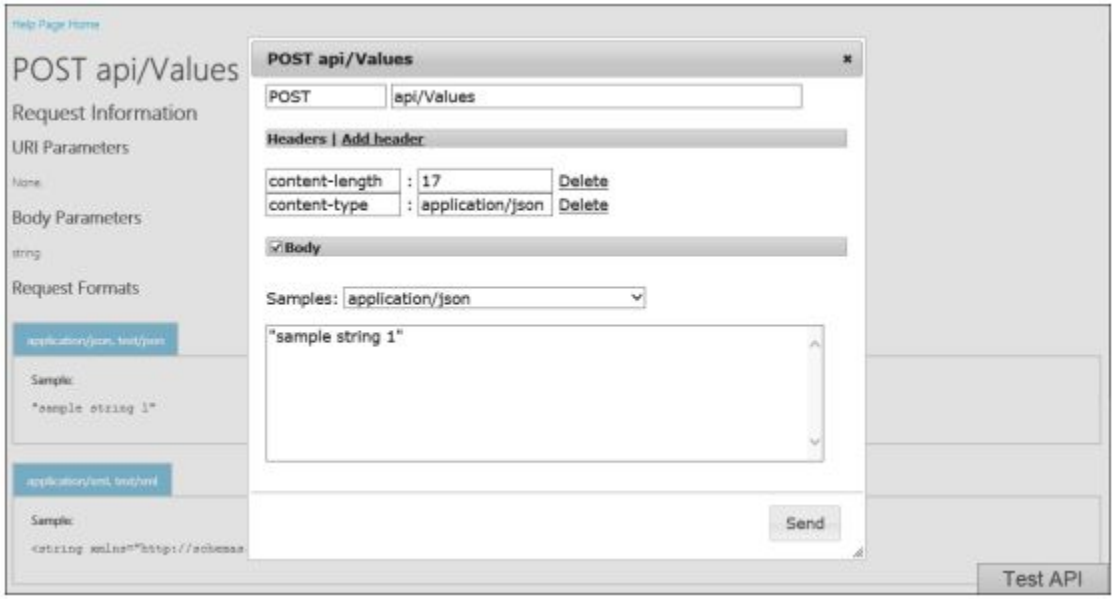
Ketika tombol Test API diklik maka akan dapat dilihat form seperti berikut :

Dari gambar di atas, pada form tersebut dapat dilihat HTTP verb GET yang digunakan untuk mengakses API ini.
Dan berikut adalah ketika tombol Send diklik.

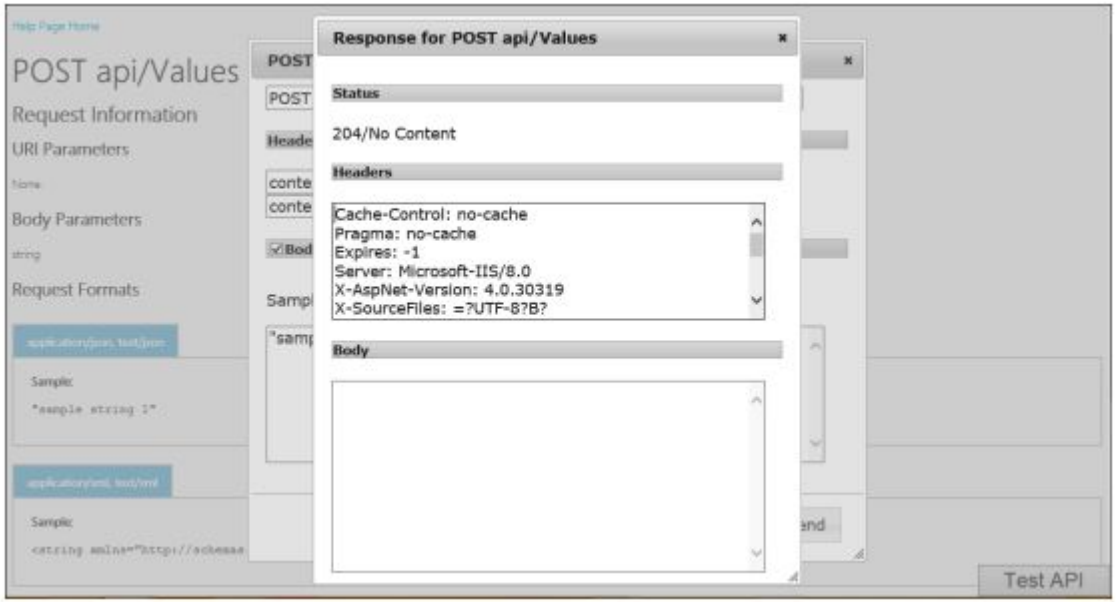


Pada output di atas, terdapat tiga output yang dapat dilihat yaitu status, headers dan body, secara kasat mata lebih lengkap informasi yang dilihat jika dibandingkan dengan akses pada web browser.

Dan berikut ini adalah form-form yang dapat dapat dilihat ketika melihat detail API untuk POST, PUT dan DELETE.


Help Page Home
POST api/Values
Request Information
URI Parameters
None.
Body Parameters
string
Request Formats
application/json, text/json
Sample:
"sample string 1"
application/xml, text/xml
Sample:
POST api/Values
POST
api/Values
Headers | Add header
content-length : 17 Delete
content-type : application/json Delete
Body
Samples: application/json
"sample string 1"
Send
Test API

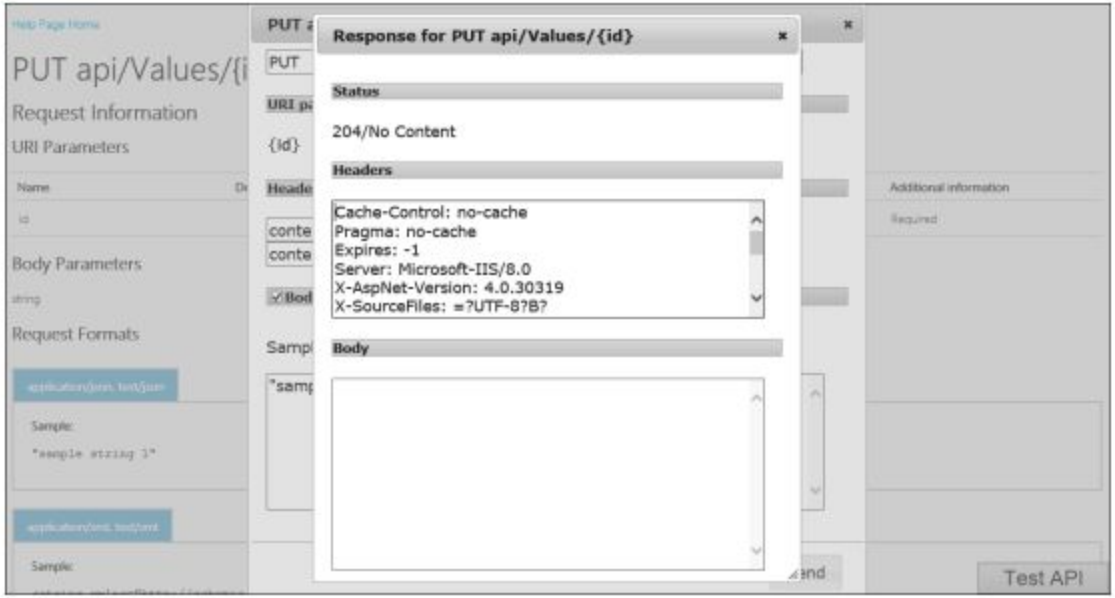
HACKTIV8


Help Page Home
POST api/Values
Request Information
URI Parameters
Body Parameters
Request Formats
Sample:
"sample string 1"
Response for POST api/Values
Status
204/No Content
Headers
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?
Body
Test API

HACKTIV8


Help Page Home
PUT api/Values/{i
Request Information
URI Parameters
Name
id
Body Parameters
string
Request Formats
application/json, text/json
Sample:
"sample string 1"
application/xml, text/xml
Sample:
PUT api/Values/{id}
PUT
api/Values/1
URI parameters
{id}
= 1
Headers | Add header
content-length : 17 Delete
content-type : application/json Delete
Body
Samples: application/json
"sample string 1"
Send
Additional information
Required
Test API

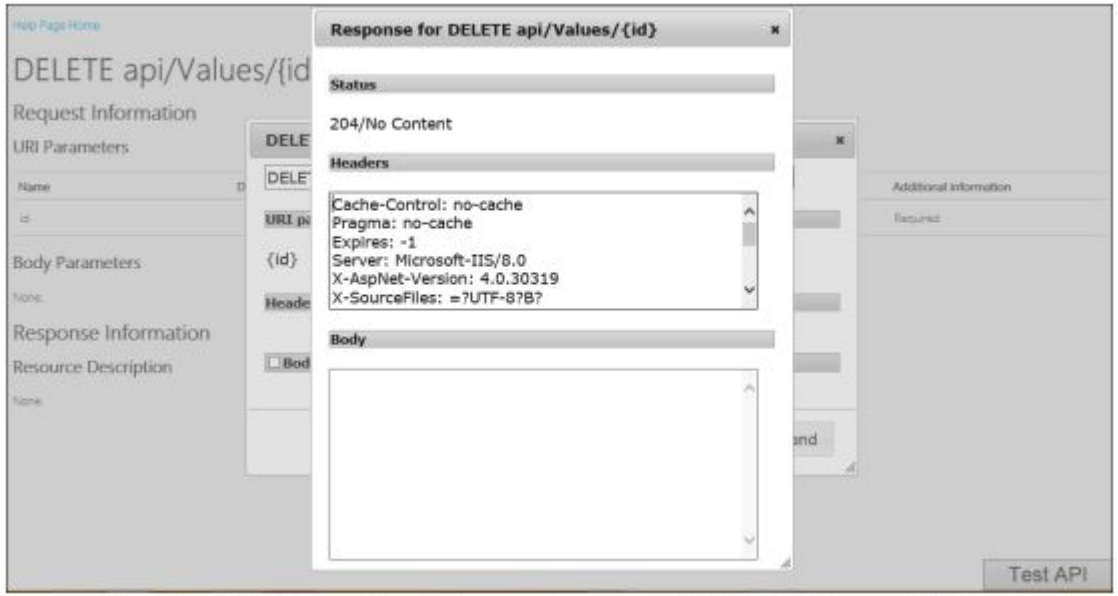
HACKTIV8


PUT api/Values/{i
Request Information
URI Parameters
Body Parameters
Request Formats
Response for PUT api/Values/{id}
Status
204/No Content
Headers
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?
Body
Test API

HACKTIV8


Help Page Home
DELETE api/Values/{id}
Request Information
URI Parameters
Name
Additional information
Required
Body Parameters
None.
Response Information
Resource Description
None.
DELETE api/Values/{id}
DELETE
api/Values/5
URI parameters
{id}
= 5
Headers | Add header
Body
Send
Test API

HACKTIV8


Response for DELETE api/Values/{id}
Status
204/No Content
Headers
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?
Body
DELETE api/Values/{id
Request Information
URI Parameters
Body Parameters
Response Information
Resource Description
Test API

HACKTIV8