IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

WEB API VERSIONING (ASP.NET CORE)

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com
http://junindar.blogspot.com

Abstrak

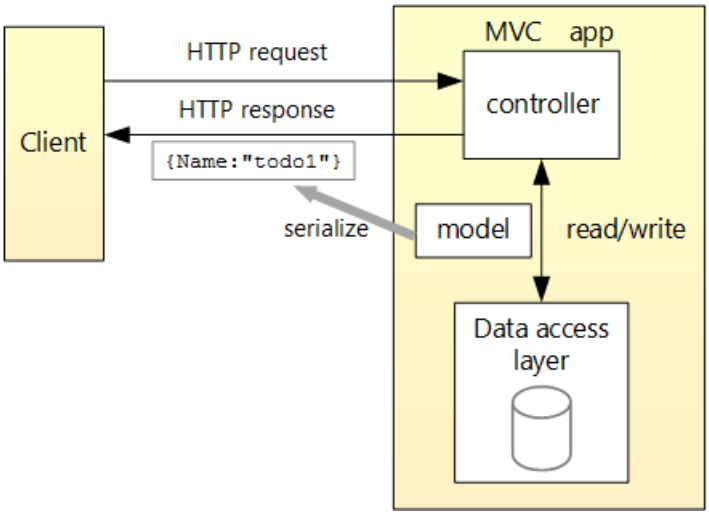
API adalah kepanjangan dari Application Programming Interface yang digunakan perangkat lunak untuk mengakses data, perangkat lunak server atau aplikasi lain dan telah ada selama beberapa waktu.

Sederhananya, API adalah perantara perangkat lunak yang menjembatani dua aplikasi untuk berbicara satu sama lain. Katakanlah API sebagai penerjemah antara dua orang yang tidak berbicara dengan bahasa yang sama, tetapi dapat berkomunikasi menggunakan perantara API.

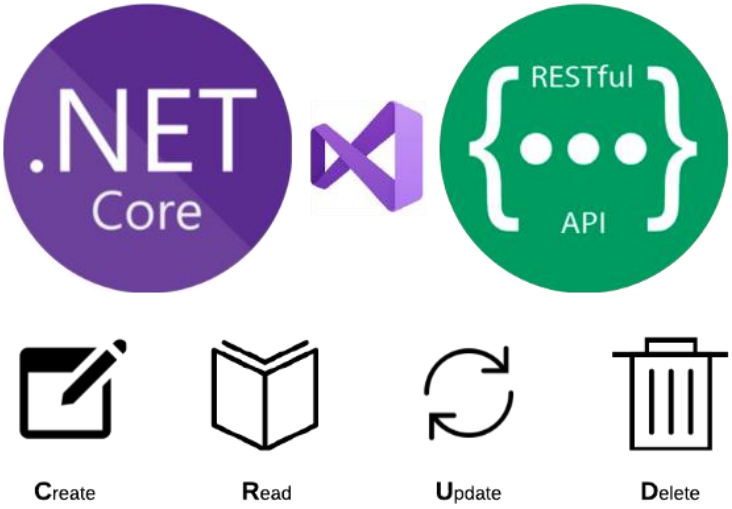
API dapat digunakan pada sistem berbasis web, sistem operasi, sistem basis data, dan perangkat keras komputer.

Pendahuluan

API berkomunikasi melalui serangkaian aturan yang menentukan bagaimana komputer, aplikasi atau mesin dapat berbicara satu sama lain. Web API bertindak sebagai perantara antara dua mesin yang ingin terhubung satu sama lain untuk tugas tertentu.



ASP.NET Core mendukung pembuatan layanan RESTful, juga dikenal sebagai Web API, dengan menggunakan C # sebagai bahasa pemogramannya. Untuk menangani request Web API menggunakan controller. Controller pada Web API adalah class yang berasal ControllerBase.



Pada saat web api telah dipublish dan digunakan oleh user, seringkali terjadi request untuk perubahan requirement maupun logic dari web api yang telah dibuat. Sementara web api dengan requirement yang lama masih digunakan. Jika mengalami situasi seperti ini, cara yang tepat adalah melakukan versioning pada web api, sehingga kita tidak perlu menghetikan layanan pada web api. Yang perlu diperhatikan versioning pada web api tidak sama dengan product versioning.

Pada API versioning kita juga harus mempertimbangkan bagaimana web api ini dapat melayani baik pengguna lama maupun yang baru. Sehingga code yang kita buat support untuk semua versi yang ada. Berikut adalah beberapa tipe dari api versioning :

- Uri

  Terdapat dua cara pada tipe ini yang pertama dengan menggunakan URI path dan yang kedua menggunakan Query String

  Uri Path : http://localhost:49423/api/v1.1/categories

  Query string : http://localhost:49423/api/categories?v=1.1
- Header

  Untuk header kita tinggal menambah X-Version dan versioning number yang akan digunakan.

  Content-Type: application/json

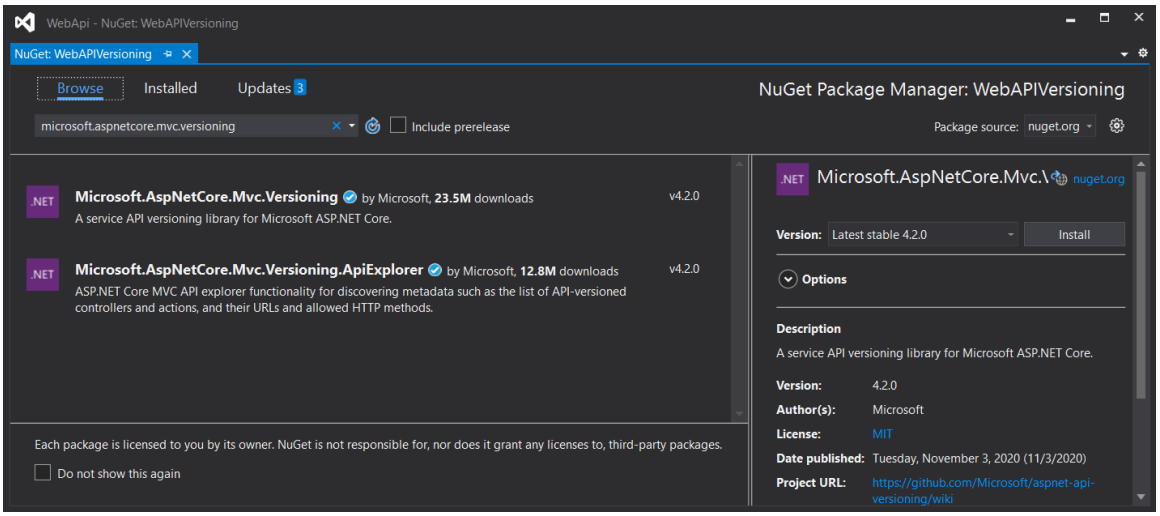
  X-Version:1.1
- Accept Header

  Content-Type: application/json

  Accept:application/json;version=1.1

Untuk memahami isi artikel ikuti langkah-langkah dibawah ini.

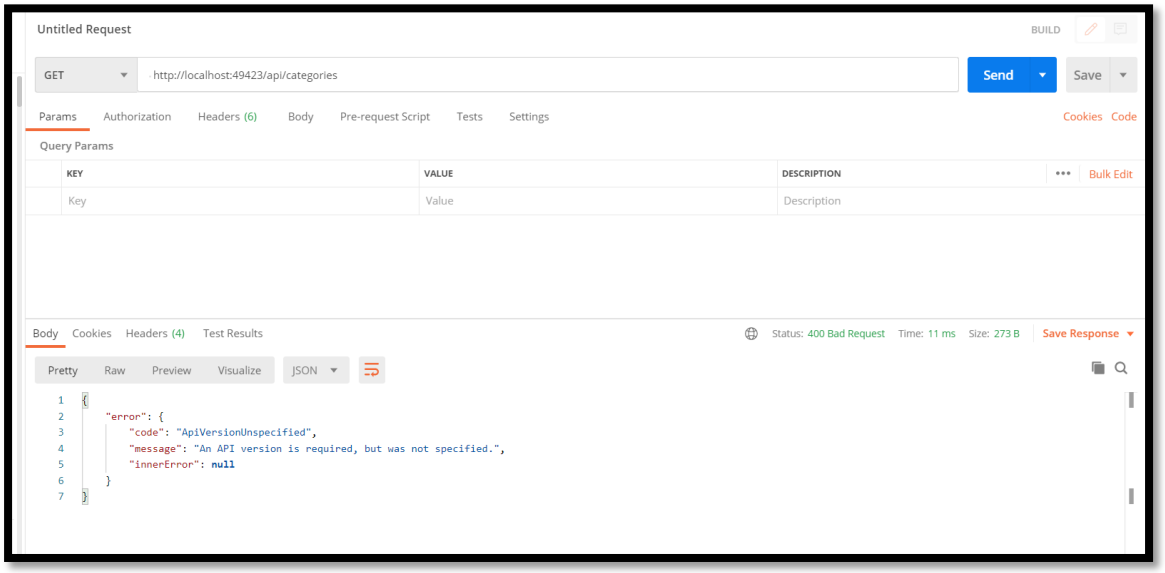
- Tambahkan sebuah reference pada project “Microsoft.AspNetCore.Mvc.Versioning“.



- Selanjutnya buka Startup.cs dan pada ConfigureServices tambahkan sintaks seperti dibawah ini.

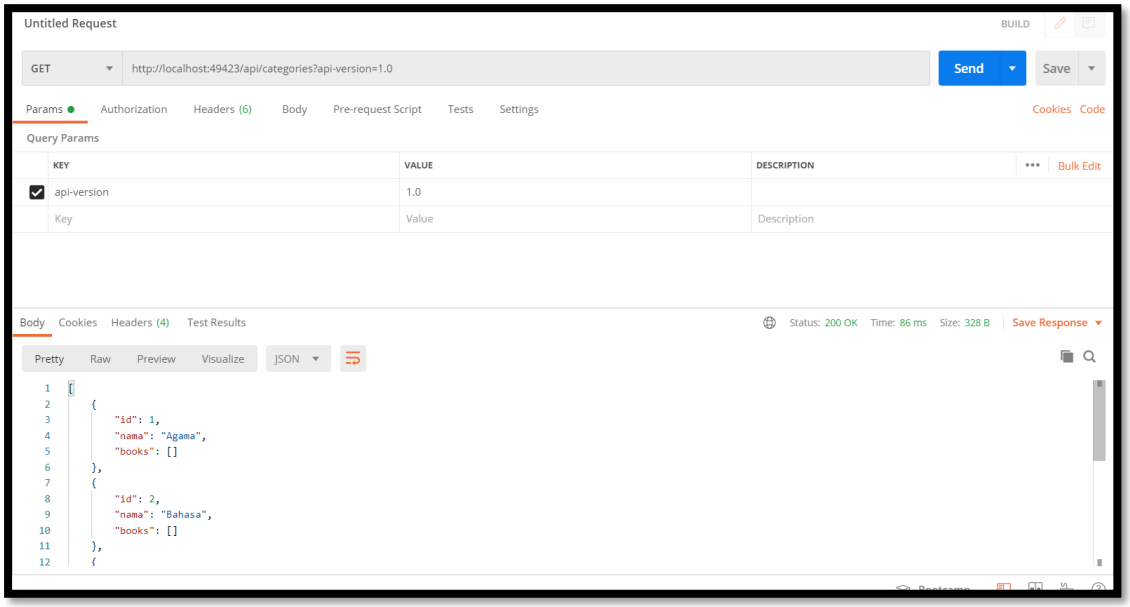
```
services.AddApiVersioning();
```

- Lalu jalankan project web api, dan coba request satu api dengan menggunakan postman seperti berikut : http://localhost:49423/api/categories



Kita akan mendapatkan error dengan status 400 Bad Request (An API version is required, but was not specified.). Artinya pada request yang kita lakukan tidak menyebutkan spesifik version dari api.

Lalu ganti url diatas dengan url berikut : http://localhost:49423/api/categories?api-version=1.0 dengan menambahkan version pada url.

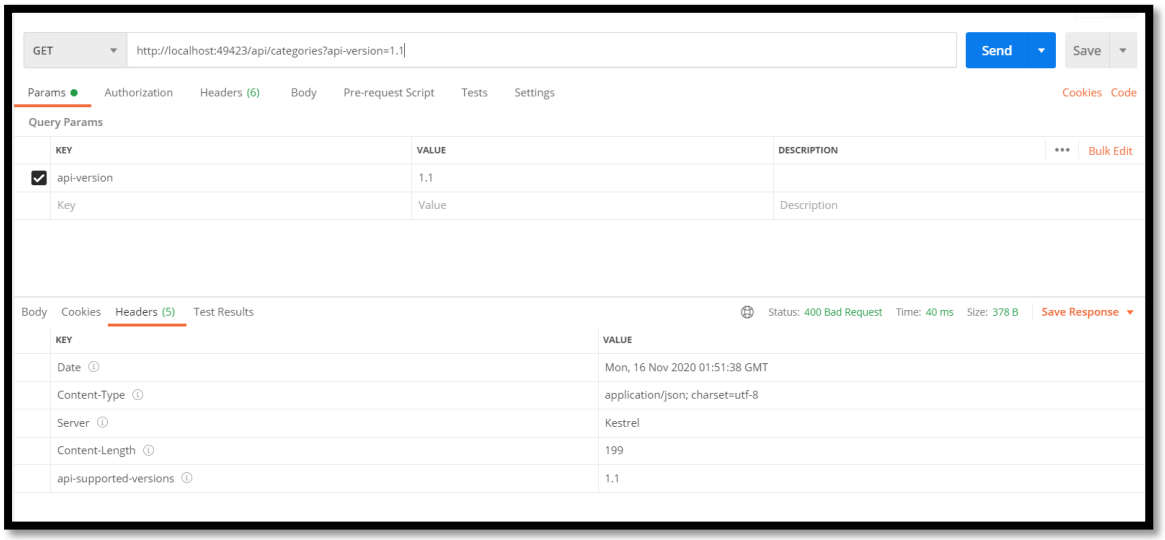


Jika version pada url tidak ditemukan (1.1) maka kita akan mendapatkan pesan error seperti berikut : **The HTTP resource that matches the request URI 'http://localhost:49423/api/categories' does not support the API version '1.1'.**

Untuk mengganti default dari web api version kita harus mengganti sintaks pada ConfigureServices seperti berikut.

```
services.AddApiVersioning(v =>
                {
                    v.DefaultApiVersion= new ApiVersion(1,1);
                    v.ReportApiVersions = true;
                }
            );
```

Dengan mengaktifkan ***ReportApiVersions***, maka pada result header akan ditambahkan sebuah informasi custom header (api-supported-version) version apa yang digunakan pada web api ini.



Versioning pada Actions

Disini kita akan membuat latihan untuk menggunkan versioning pada Actions, yang artinya kita akan menentukan spesifik version pada sebuah actions. Untuk memahami lebih lanjut, ikuti langkah-langkah dibawah ini.

- Buka file CategoryController, lalu tambahkan atribut ApiVersion seperti pada gambar dibawah. Sebagai informasi pada contoller ini akan mensupport dua versi dari api yaitu 1.0 dan 1.1.

```
namespace WebAPIVersioning.Controllers
{

    [ApiController]
    [Route("api/categories")]
    [ApiVersion("1.0")]
    [ApiVersion("1.1")]
    1 reference | 0 changes | 0 authors, 0 changes
    public class CategoryController : ControllerBase
    {
        private readonly ICategoryRepository _categoryRepository;
        private readonly IMapper _mapper;
        0 references | 0 changes | 0 authors, 0 changes
        public CategoryController(ICategoryRepository categoryRepository,IMapper mapper)
        {
            _categoryRepository = categoryRepository;
            _mapper = mapper;


        }
```

- Disini kita akan melakukan versioning pada actions Get. Sebelumnya copy terlebih dahulu untuk actions GetCategories dan pada actions yang baru ganti namanya menjadi " GetCategoriesV_1_1".
- Tambahkan atribut MapToApiVersion pada kedua actions tersebut seperti pada gambar dibawah. Lalu pada " GetCategoriesV_1_1" ganti method **GetAll** menjadi "**GetAllIncludeBook**". Untuk detail method "**GetAllIncludeBook**" dapat dilihat

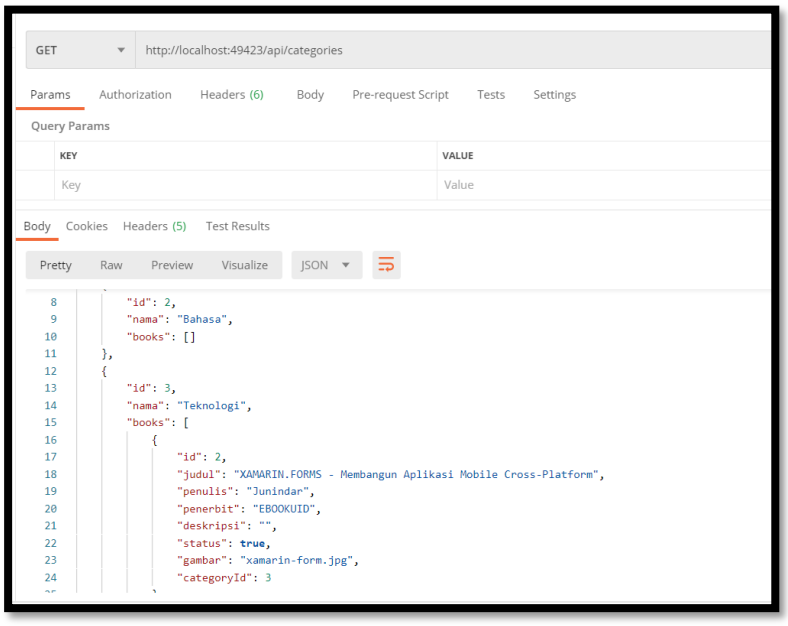
pada project lampiran. Method ini akan menampilkan data Category dan Buku sesuai dengan category nya.

```
[HttpGet()]
[MapToApiVersion("1.0")]
0 references | 0 changes | 0 authors, 0 changes
public async Task<IActionResult> GetCategories()
{

    var resultRepo = await _categoryRepository.GetAll();
    return Ok(_mapper.Map<IEnumerable<CategoryDto>>(resultRepo));


}

[HttpGet()]
[MapToApiVersion("1.1")]
0 references | 0 changes | 0 authors, 0 changes
public async Task<IActionResult> GetCategoriesV_1_1()
{

    var resultRepo = await _categoryRepository.GetAllIncludeBook();
    return Ok(_mapper.Map<IEnumerable<CategoryDto>>(resultRepo));


}
```

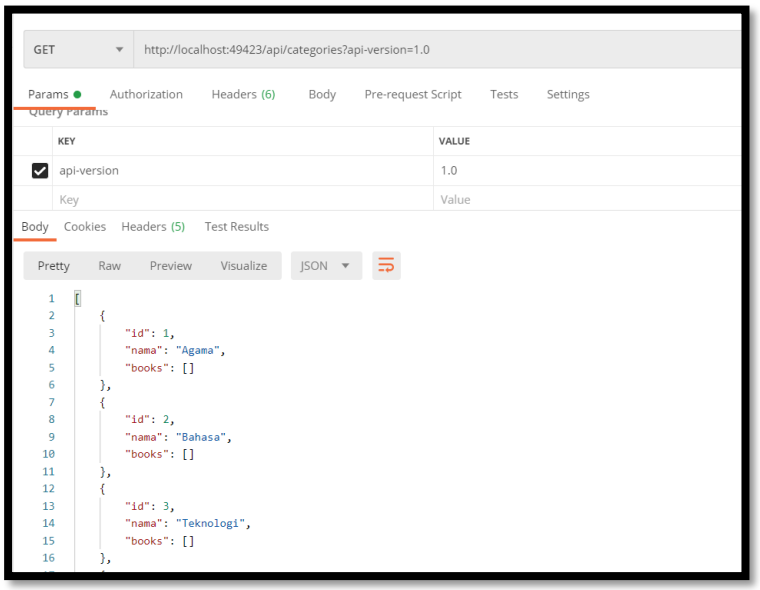
Dan pada ConfigureServices tambahkan sintaks berikut dalam "AddApiVersioning".

```
v.AssumeDefaultVersionWhenUnspecified = true;
```

Fungsi dari sintaks ini adalah kita tidak perlu menambahkan informasi versi api pada request. Maka secara otomatis akan memanggil defaul api version yang telah kita set pada ConfigureServices sebelum nya. Sekarang saatnya kita jalankan project web api ini. Lakukan request dengan menggunakan url berikut : http://localhost:49423/api/categories



Hasil yang kita dapat untuk pengetesan diatas adalah data category dan buku akan ditampilkan. Lanjutkan dengan request kedua dengan menggunakan url berikut : http://localhost:49423/api/categories?api-version=1.0 .Pastikan hasilnya seperti dibawah atau hanya menampilkan data category nya saja.



Versioning pada Controllers

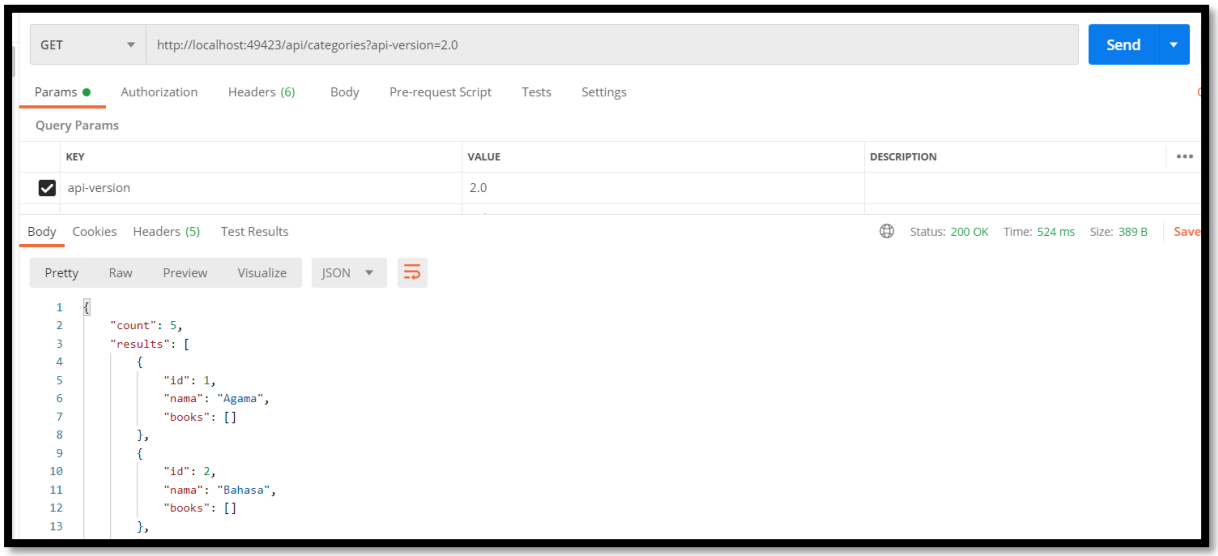
Setelah selesai dengan latihan versioning pada Actions, selanjutnya akan dibahas bagaimana melakukan versioning pada controller. Untuk lebih jelasnya ikuti langkah-langkah dibawah ini.

- Tambahkan sebuah controller dengan nama “Category2Controller”.
- Copy action “GetCategories” dari CategoryController, beserta dengan constructor nya seperti dibawah.
- Ganti Atribut ApiVersioning pada controller menjadi 2.0 di ikuti dengan menganti sintaks dari GetCategories. Pada action ini kita akan menambahkan sebuah informasi “Count” dimana datanya merupakan jumlah dari category yang ada.

```
[ApiController]
[Route("api/categories")]
[ApiVersion("2.0")]
1 reference | 0 changes | 0 authors, 0 changes
public class Category2Controller : ControllerBase
{
    private readonly ICategoryRepository _categoryRepository;
    private readonly IMapper _mapper;
    0 references | 0 changes | 0 authors, 0 changes
    public Category2Controller(ICategoryRepository categoryRepository,IMapper mapper)
    {
        _categoryRepository = categoryRepository;
        _mapper = mapper;
    }

    [HttpGet()] 5 7
    0 references | 0 changes | 0 authors, 0 changes
    public async Task<IActionResult> GetCategories()
    {
        var resultRepo = await _categoryRepository.GetAll();
        var result = new
        {
            Count=resultRepo.Count,
            Results=_mapper.Map<IEnumerable<CategoryDto>>(resultRepo)
        };
        return Ok(result);
    }
```

Setelah selesai dengan langkah-langkah diatas, jalankan project ini dan lakukan pengetesan seperti dibawah : http://localhost:49423/api/categories?api-version=2.0

Pastikan hasil yang didapat seperti pada gambar berikut ini.



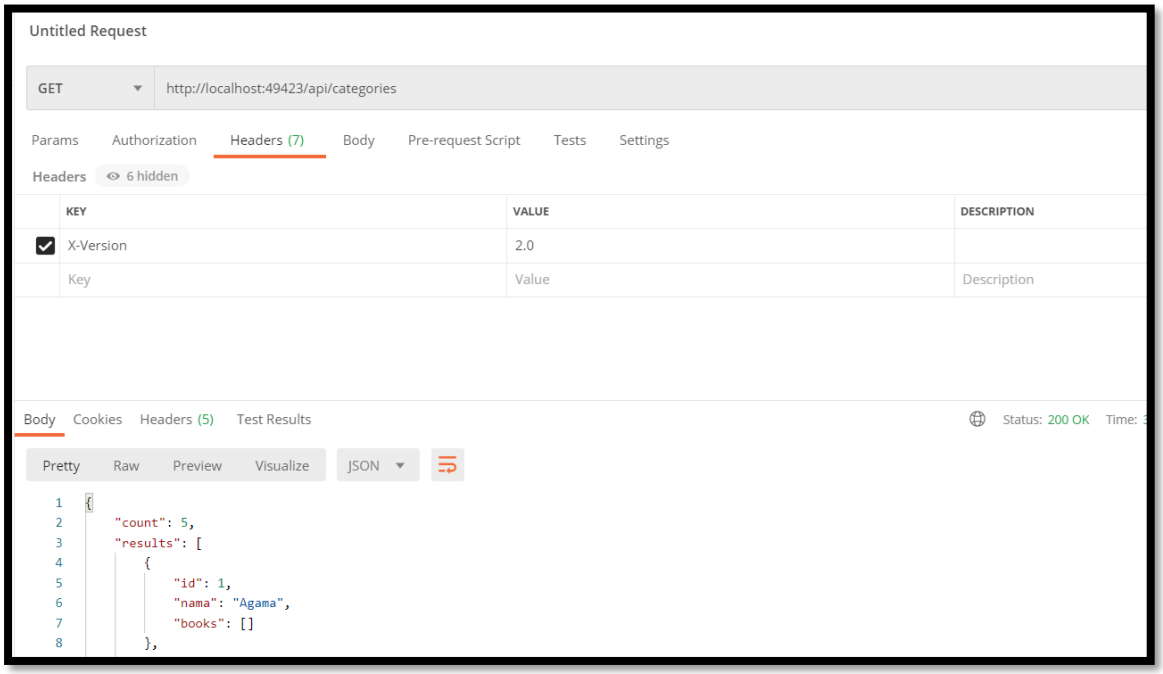
Jadi pada controller ini akan menangani api veri 2 pada project ini.

Versioning dengan Header

Untuk melakukan versioning dengan header, kita tidak perlu lagi mengganti sintaks baik pada controller maupun action. Pada latihan ini kita cukup menambahkan sintaks pada Startup.cs lebih tepatnya di “ApiVersioningOption“ seperti dibawah ini.

```
v.ApiVersionReader= new HeaderApiVersionReader("X-Version");
```

Sehingga untuk melakukan request pada url kita tidak perlu menambahkan query string, cukup dengan menambahkan custom header Key=X-Version dan Value=versi dari api yang akan dipanggil. Jalankan program dan lakukan pengetesan seperti pada gambar dibawah ini.



Lalu jika ada pertanyaan bagaimana jika kita ingin melakukan kombinasi dalam melakukan request pada web api seperti menggunakan query string ataupun dengan header. Pada latihan sebelumnya kita menggunakan “api-version“ sebagai key parameternya. Lalu bagaimana jika kita ingin mengganti nya misal menjadi “ver“.

Untuk melakukan dua hal tersebut ikuti langkah-langkah dibawah ini.

- Buka file Startup.cs
- Hapus atau comment sintaks dibawah ini.

  ```
  v.ApiVersionReader= new HeaderApiVersionReader("X-Version");
  ```

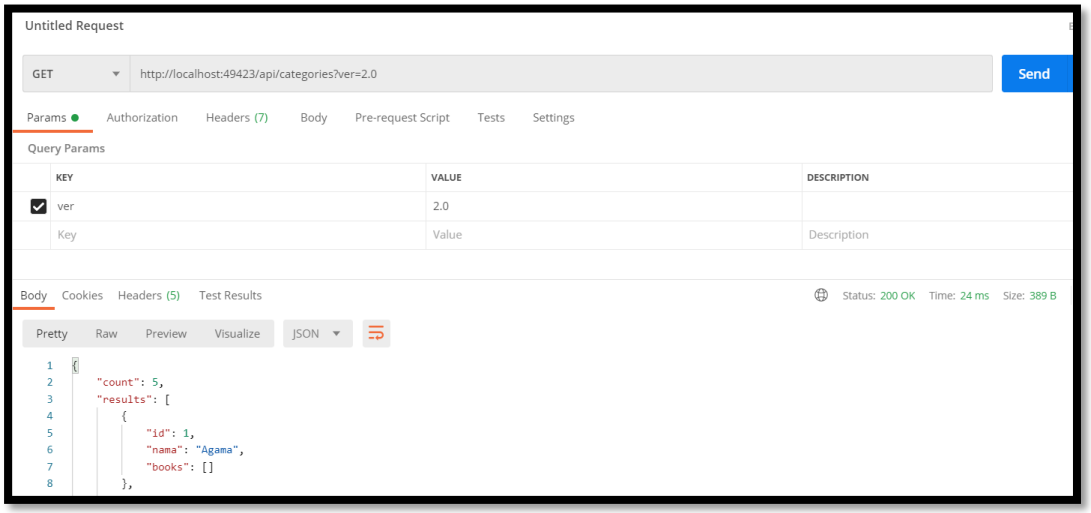
- Selanjutnya tambahkan sintaks seperti dibawah ini

```
v.ApiVersionReader = ApiVersionReader.Combine(
                        new HeaderApiVersionReader("X-Version"),
                        new QueryStringApiVersionReader("ver", "version")
);
```

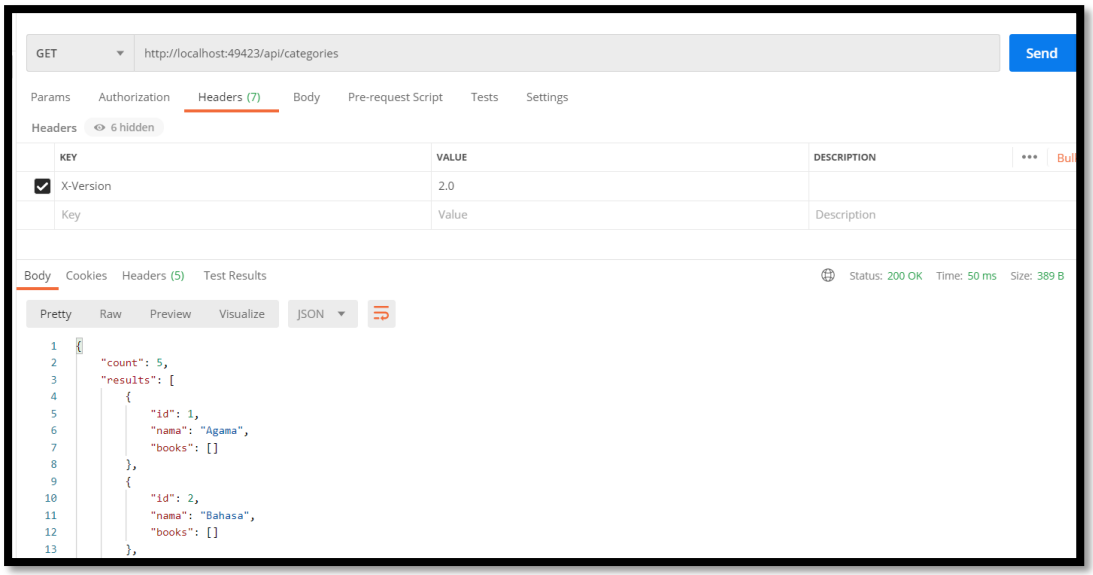
Disini kita menggunakan method Combine dari ApiVersionHeader. Dan pada parameter didalam method Combine kita tambahkan HeaderApiVersionReader dan QueryStringApiVersionReader.

- Jalankan program dan lakukan pengetesan baik menggunakan query string maupun custom header.

Query String : http://localhost:49423/api/categories?ver=2.0



Custom Header : X-Version



URL Versioning

Dan untuk pembahasan terakhir pada artikel ini adalah Url Versioning. Kita dapat melakukan versioning api pada url dengan melakukan beberapa step seperti dibawah ini.

- Masih pada Startup.cs, tambahkan parameter baru (new UrlSegmentApiVersionReader()) pada Combine seperti pada gambar dibawah ini.

```
services.AddApiVersioning(v =>
    {
        v.AssumeDefaultVersionWhenUnspecified = true;
        v.DefaultApiVersion= new ApiVersion(1,1);
        v.ReportApiVersions = true;
        v.ApiVersionReader = ApiVersionReader.Combine(
            new HeaderApiVersionReader("X-Version"),
            new QueryStringApiVersionReader("ver", "version"),
            new UrlSegmentApiVersionReader());

    }
);
```

- Selanjutnya buka Category2Controller dan ganti atribut route nya seperti pada sintaks dibawah ini.

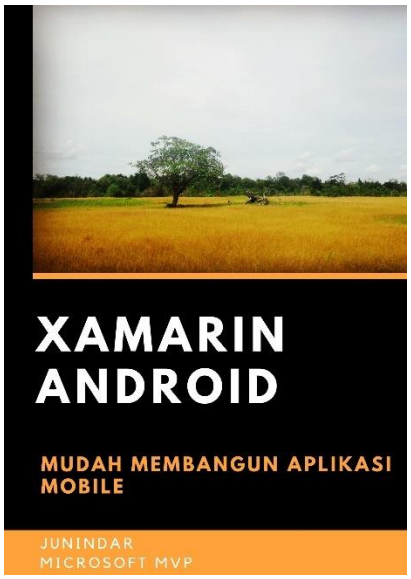
```
[Route("api/v{version:apiVersion}/categories")]
```

Lalu jalankan program dan coba lakukan pengetesan dengan menggunakan url seperti berikut http://localhost:49423/api/v2/categories

Penutup

Sedangkan untuk memudahkan dalam memahami isi artikel, maka penulis juga menyertakan dengan full source code project latihan ini, dan dapat di download disini http://junindar.blogspot.com/2020/12/web-api-versioning-aspnet-core.html

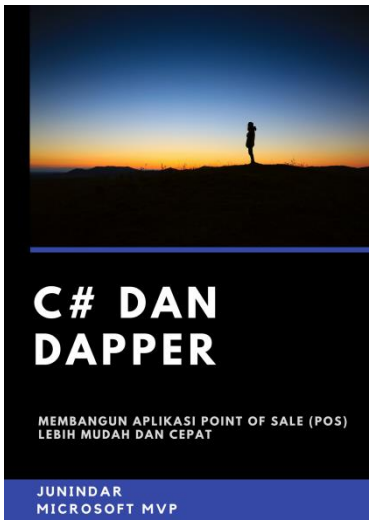
Referensi



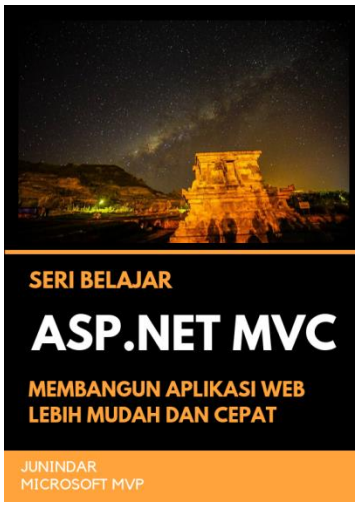
https://play.google.com/store/books/details?id=G4tFDgAAQBAJ

https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

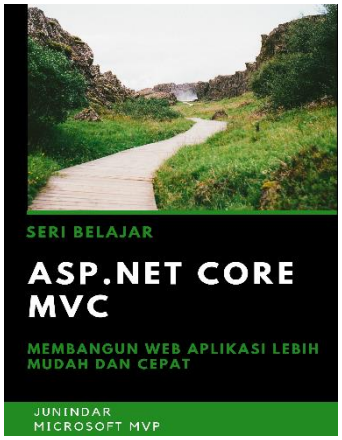
https://play.google.com/store/books/details/Junindar_Xamarin_Forms?id=6Wg-DwAAQBAJ



https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ



https://play.google.com/store/books/details/Junindar_ASP_NET_MVC_Membangun_Aplikasi_Web_Lebih?id=XLlyDwAAQBAJ



https://play.google.com/store/books/details/Junindar_ASP_NET_CORE_MVC?id=xEe5DwAAQBAJ

Biografi Penulis.

Junindar Lahir di Tanjung Pinang, 21 Juni 1982. Menyelesaikan Program S1 pada jurusan Teknik Inscreenatika di Sekolah Tinggi Sains dan Teknologi Indonesia (ST-INTEN-Bandung). Junindar mendapatkan Award Microsoft MVP VB pertanggal 1 oktober 2009 hingga saat ini. Senang mengutak-atik computer yang berkaitan dengan bahasa pemrograman. Keahlian, sedikit mengerti beberapa bahasa pemrograman seperti : VB.Net, C#, SharePoint, ASP.NET, VBA. Reporting: Crystal Report dan Report Builder. Database: MS Access, MY SQL dan SQL Server. Simulation / Modeling Packages: Visio Enterprise, Rational Rose dan Power Designer. Dan senang bermain gitar, karena untuk bisa menjadi pemain gitar dan seorang programmer sama-sama membutuhkan seni. Pada saat ini bekerja di salah satu Perusahaan Consulting dan Project Management di Malaysia sebagai Senior Consultant. Memiliki beberapa sertifikasi dari Microsoft yaitu Microsoft Certified Professional Developer (MCPD – SharePoint 2010), MOS (Microsoft Office Specialist) dan MCT (Microsoft Certified Trainer) Mempunyai moto hidup: "**Jauh lebih baik menjadi Orang Bodoh yang giat belajar, dari pada orang Pintar yang tidak pernah mengimplementasikan ilmunya".**