IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Open API/Swagger pada WEB API (ASP.NET CORE)

Junindar, ST, MCPD, MOS, MCT, MVP

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

Abstrak

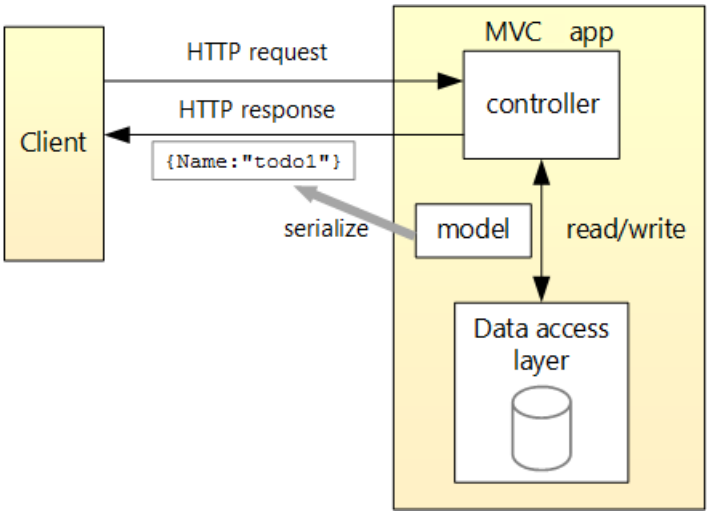
API adalah kepanjangan dari Application Programming Interface yang digunakan perangkat lunak untuk mengakses data, perangkat lunak server atau aplikasi lain dan telah ada selama beberapa waktu.

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.

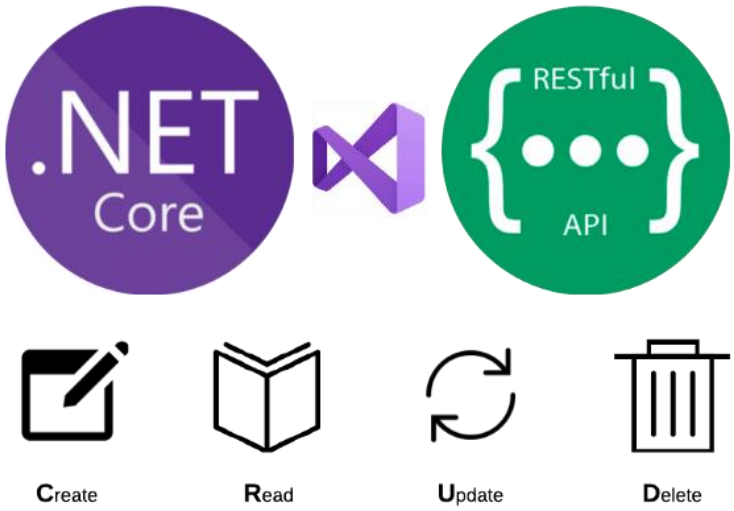
API dapat digunakan pada sistem berbasis web, sistem operasi, sistem basis data, dan perangkat keras komputer.

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.



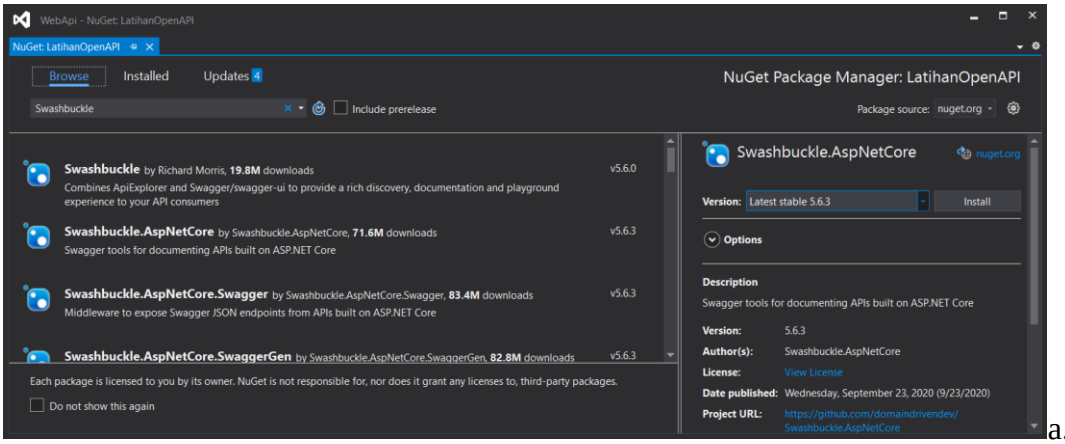
ASP.NET Core mendukung pembuatan layanan RESTful, juga dikenal sebagai Web API, dengan menggunakan C # sebagai bahasa pemogramannya. Untuk menangani request Web API menggunakan controller. Controller pada Web API adalah class yang berasal ControllerBase.



Dokumentasi termasuk hal penting dalam membangun sebuah aplikasi. Sebagai contoh jika kita telah membuat sebuah web api yang akan digunakan oleh beberapa developer, tentunya developer-developer tersebut akan selalu bertanya kepada kita tentang setiap fungsi yang ada pada web api kita tersebut. Untuk menghidari komunikasi yang berulang-ulang, sebaiknya kita buat sebuah dokumentasi agar memudahkan para developer yang akan menggunakan web api kita. Pada webapi kita dapat menggunakan Swagger/Open Api sebagai alat untuk membuat dokumentasinya.

Untuk memudahkan dalam memahami artikel, ikuti langkah-langkah dibawah ini.

1. Tambahkan sebuah reference “Swashbuckle.AspNetCore“ kedalam project web api yang ad

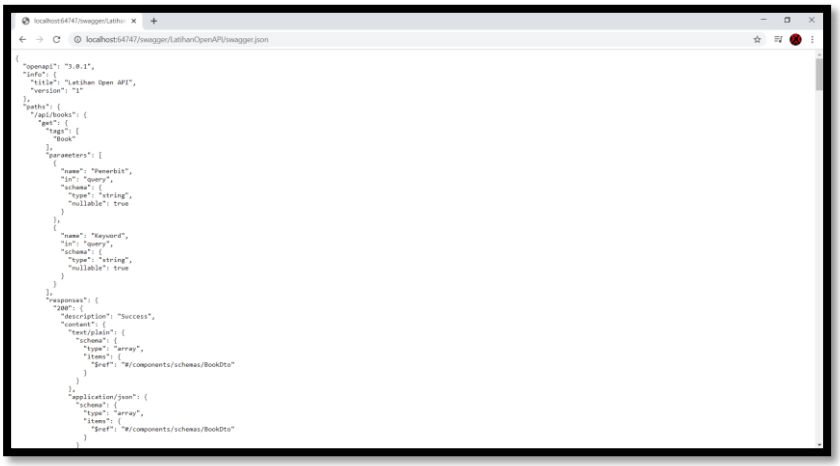


a.

2. Buka Startup.cs, lalu tambahkan sintaks dibawah pada method ConfigureService.

```
services.AddSwaggerGen(setupAction =>
{
                setupAction.SwaggerDoc("LatihanOpenAPI", new
Microsoft.OpenApi.Models.OpenApiInfo()
                {
                    Title = "Latihan Open API", Version = "1"
                });
});
```

3. Lalu pada method Configure ketikkan sintaks berikut : ***app.UseSwagger();*** untuk melihat hasilnya jalankan project web api. Dan ketikkan url seperti berikut http://localhost:64747/swagger/LatihanOpenAPI/swagger.json



Pada gambar diatas dapat dilihat sebuah json file dari swagger yang berisi informasi web api yang kita miliki.

Menambahkan Swagger UI

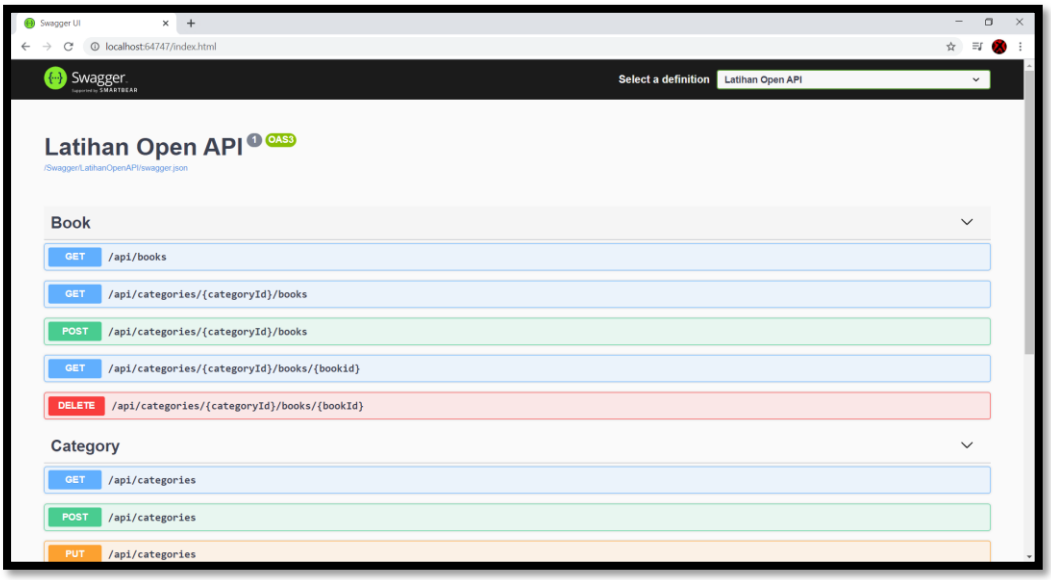
Setelah selesai dengan latihan diatas, akan kita lanjutkan dengan latihan untuk menambahkan Swagger UI dan mengganti URL untuk mengaksesnya. Ikuti langkah-langkah dibawah ini.

Buka Startup.cs dan tambahkan sintaks dibawah ini pada method Configure.

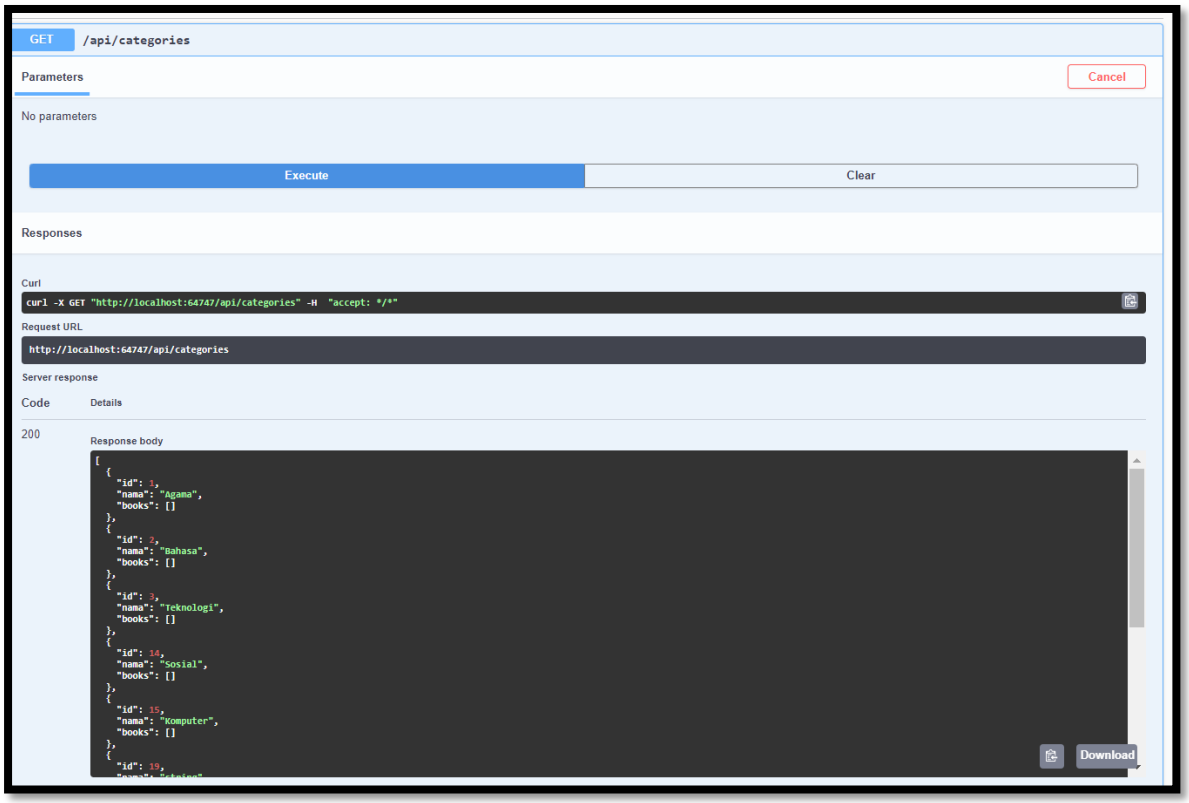
```
app.UseSwaggerUI(setupAction =>
{
    setupAction.SwaggerEndpoint("/Swagger/LatihanOpenAPI/swagger.json", "Latihan Open API");
    setupAction.RoutePrefix = "";
});
```

Dengan menambahkan sintaks diatas kita telah menambahkan Swagger UI pada project dan Swagger UI tersebut dapat diakses dengan url berikut : http://localhost:64747/index.html . Jalankan program untuk melihat hasilnya, seperti pada gambar dibawah ini.

Pada Swagger UI dapat kita lihat setiap action yang ada pada project kita dan action-action tersebut akan di group berdasarkan controller.



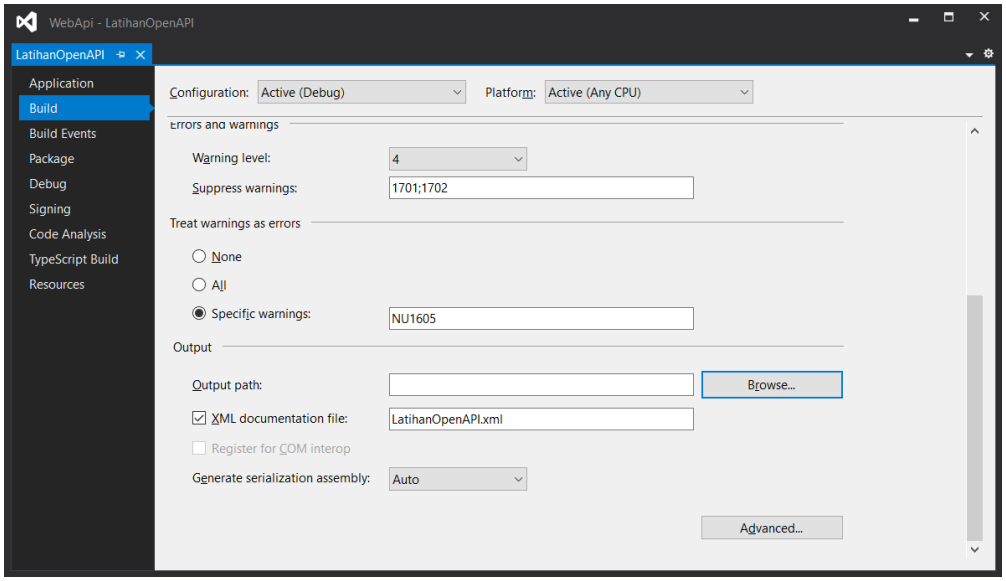
Sekarang kita akan mencoba menggunakan Swagger UI untuk mencoba melakukan request API yang telah kita buat. Pada group **Category** klik api/Categories (GET), lalu klik button "**Try it out**", setelah itu button "Execute" akan tampil dan klik button tersebut. Maka hasil dari request akan keluar seperti pada gambar dibawah ini.



Menambahkan XML Comments pada Actions

Sebagai dokumentasi kita perlu memberikan informasi sejelas-jelasnya kepada para developer yang akan menggunakan web api kita. Sebagai contoh fungsi dari setiap action yang telah kita buat. Pada latihan ini kita akan memberikan informasi pada action yang ada pada project dengan menggunakan XML Comment.

Pertama-tama buka project properties dan pilih tab Build.



Pada Section Output, aktifkan checkbox “XML documentation file“ dan ketikkan nama project di tambah dengan extension .xml. (LatihanOpenAPI.xml)

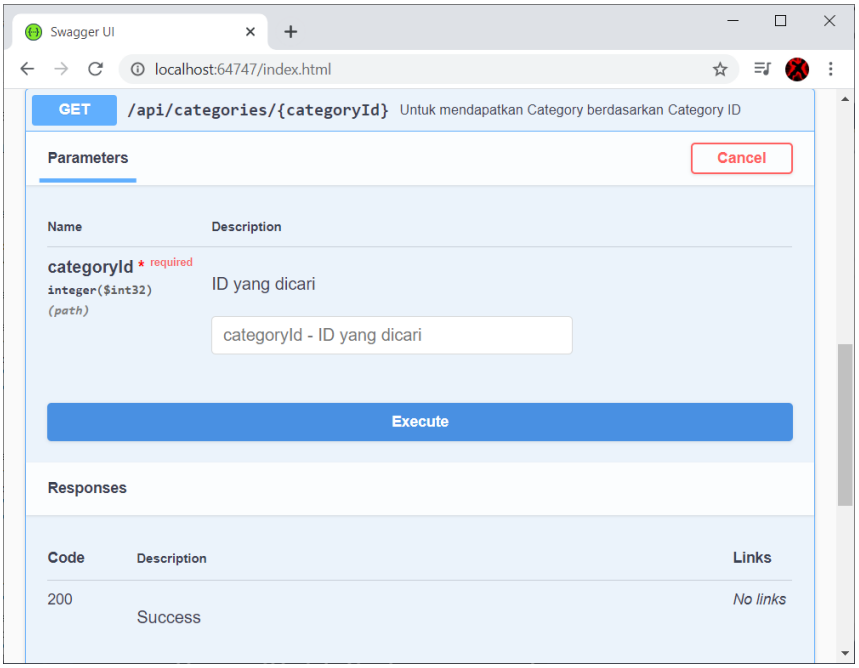
Lalu tambahkan sintaks seperti dibawah pada service.AddSwaggerGen (file Startup.cs).

```
var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
var xmlFullPath = Path.Combine(AppContext.BaseDirectory,xmlFile);
setupAction.IncludeXmlComments(xmlFullPath);
```

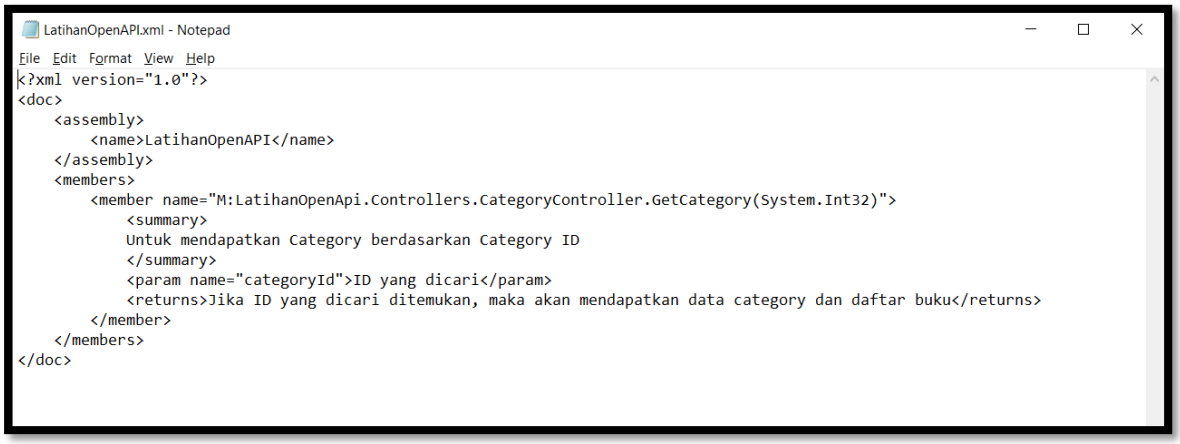
Buka Controller Category, lalu pada action GetCategory tambahkan Summary seperti berikut. Cara cepat untuk membuat summary ini adalah dengan mengetikkan “/“ (slash) sebanyak 3 kali lalu diikuti dengan tab sebanyak dua kali.

```
/// <summary>
/// Untuk mendapatkan Category berdasarkan Category ID
/// </summary>
/// <param name="categoryId">ID yang dicari</param>
/// <returns>Jika ID yang dicari ditemukan, maka akan mendapatkan data category dan daftar buku</returns>
[HttpGet("{categoryId}",Name = "GetCategory")]
0 references | Junindar, 2 days ago | 1 author, 1 change
public async Task<IActionResult> GetCategory(int categoryId)
{

    var resultRepo = await _categoryRepository.GetById(categoryId);

    if (resultRepo == null)
    {
        return NotFound();
    }
    return Ok(_mapper.Map<CategoryDto>(resultRepo));


}
```

Untuk melihat hasilnya jalankan program dan buka Swagger UI, pastikan mendapatkan hasil seperti pada gambar dibawah ini. Dimana terdapat informasi mengenai api yang telah kita buat sebelumnya.



Dibawah ini merupakan file xml yang digenerate berdasarkan summary tags yang telah kita buat.



Dokumentasi pada Data Annotation

Selain kita dapat membuat dokumentasi untuk action, kita juga dapat melakukannya pada Model yang digunakan oleh API. Sebagai contoh pada proses insert (add) kita menggunakan model dengan nama "KategoriAddDTO", dengan melakukan hal yang sama seperti pada langkah sebelumnya, lakukan seperti pada gambar dibawah ini.

```
/// <summary>
/// Model untuk proses CRUD pada API
/// </summary>
5 references | 0 changes | 0 authors, 0 changes
public class KategoriAddDTO
{
    /// <summary>
    /// Category ID
    /// </summary>
    [Required]
    3 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }

    /// <summary>
    /// Nama Kategori
    /// </summary>
    [Required]
    [MaxLength(50, ErrorMessage = "Nama tidak boleh lebih dari 50 karakter")]
    2 references | 0 changes | 0 authors, 0 changes
    public string Nama { get; set; }


}
```

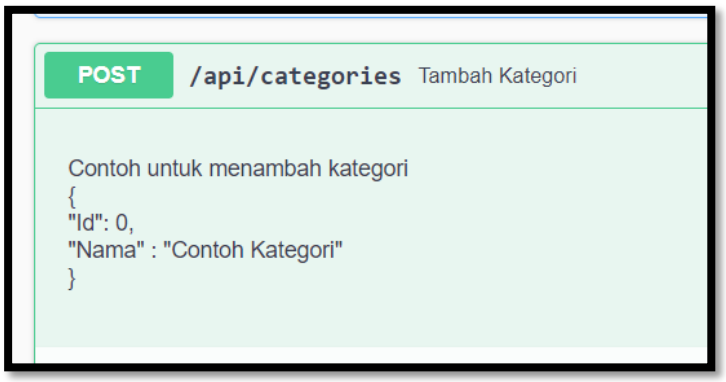
Jalankan program dan buka Swagger UI, scroll halaman sampai kebawah pada Section "**Schemas**". Expand KategoriAddDTO, semua dokumentasi pada Model yang telah kita buat sebelumnya akan tampil pada section ini. Seperti Deskripsi dari Model, entity hingga tipe data dari setiap entity tersebut.

Membuat Sample

Agar lebih memudahkan para developer yang akan menggunakan API kita, sebaiknya kita juga membuat contoh (sample) dari data yang akan dikirim ke API. Lakukan seperti pada gambar dibawah ini.

```
/// <summary>
/// Tambah Kategori
/// </summary>
/// <returns>Data Kategori yang baru dibuat</returns>
/// <remarks>
/// Contoh untuk menambah kategori \
/// {\
///     "Id": 0,\
///     "Nama" : "Contoh Kategori"\
/// }
/// </remarks>
[HttpPost]
0 references | 0 changes | 0 authors, 0 changes
public async Task<ActionResult<KategoriAddDTO>> CreateCategory(KategoriAddDTO category)
{
    var categoryEntity = _mapper.Map<Category>(category);
    categoryEntity = await _categoryRepository.InsertCategory(categoryEntity);

    var categoryForReturn = _mapper.Map<KategoriAddDTO>(categoryEntity);

    return CreatedAtRoute("GetCategory", new { categoryId = categoryForReturn.Id }, categoryForReturn);

}
```

Jalankan program dan pastikan mendapatkan hasil seperti pada gambar dibawah ini.



Menambahkan Informasi pada Open API.

Pada Swagger kita dapat menambah informasi mengenai API kita, seperti menambahkan deskripsi maupun Contact (Email, Name dan Url) dari pembuat API. Buka Startup.cs tambahkan sintaks berikut pada "`services.AddSwaggerGen`" didalam method ConfigureServices.

```
Description ="Latihan membuat Web API dengan menggunakan .Net Core",
Contact = new OpenApiContact()
{
      Email = "Junindar@gmail.com",
      Name = "Junindar",
      Url = new Uri("http://junindar.blogspot.com")
}
```

Jalankan program dan pastikan mendapatkan hasil seperti pada gambar dibawah ini.

Membuat Multiple Open API Specifications

Untuk pembahasan terakhir dari artikel ini, kita akan membuat Multiple Open API Specifications, dimana kita dapat membuat group dari API yang telah kita tentukan. Sebagai contoh kita akan memisahkan tampilan antara API yang berkaitan dengan Category dan Book.

```
services.AddSwaggerGen(setupAction =>
{
  setupAction.SwaggerDoc("LatihanOpenAPICategory", new Microsoft.OpenApi.Models.OpenApiInfo()
                {
                    Title = "Latihan Open API-Category", Version = "1",
                    Description ="Latihan membuat Web API dengan menggunakan .Net Core-Category",
                    Contact = new OpenApiContact()
                    {
                        Email = "Junindar@gmail.com",
                        Name = "Junindar",
                        Url = new Uri("http://junindar.blogspot.com")

                    }
                });

 setupAction.SwaggerDoc("LatihanOpenAPIBook", new Microsoft.OpenApi.Models.OpenApiInfo()
                {
                    Title = "Latihan Open API-Book",
                    Version = "1",
                    Description = "Latihan membuat Web API dengan menggunakan .Net Core - Book",
                    Contact = new OpenApiContact()
                    {
                        Email = "Junindar@gmail.com",
                        Name = "Junindar",
                        Url = new Uri("http://junindar.blogspot.com")

                    }
                });
                var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
                var xmlFullPath = Path.Combine(AppContext.BaseDirectory,xmlFile);
                setupAction.IncludeXmlComments(xmlFullPath);
});
```

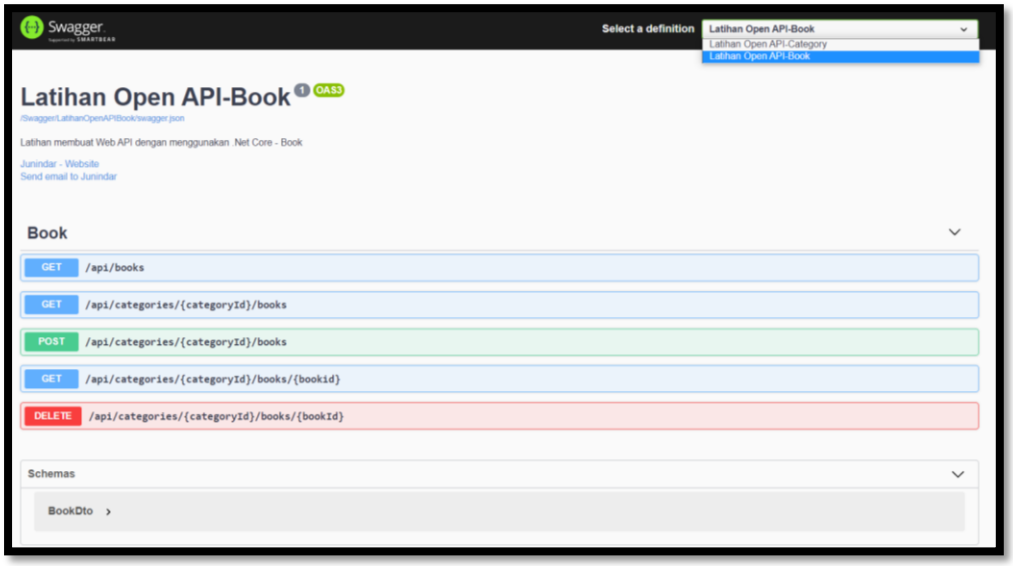
Pada sintaks diatas dapat kita lihat, terdapat dua buah SwaggerDoc, yang pertama **LatihanOpenApiCategory** dan **LatihanOpenApiBook**. Begitu juga dengan sintaks dibawah ini terdapat dua buat SwaggerEndPoint baik untuk Category maupun untuk Book.

```
app.UseSwaggerUI(setupAction =>
{
    setupAction.SwaggerEndpoint("/Swagger/LatihanOpenAPICategory/swagger.json", "Latihan Open API-
Category");
    setupAction.SwaggerEndpoint("/Swagger/LatihanOpenAPIBook/swagger.json", "Latihan Open API-
Book");
     setupAction.RoutePrefix = "";
});
```

Langkah terakhir dengan menambahkan sintak dibawah untuk masing-masing controller

```
//untuk Category Controller
[ApiExplorerSettings(GroupName = "LatihanOpenAPICategory")]


//untuk Book Controller
[ApiExplorerSettings(GroupName = "LatihanOpenAPIBook")]
```

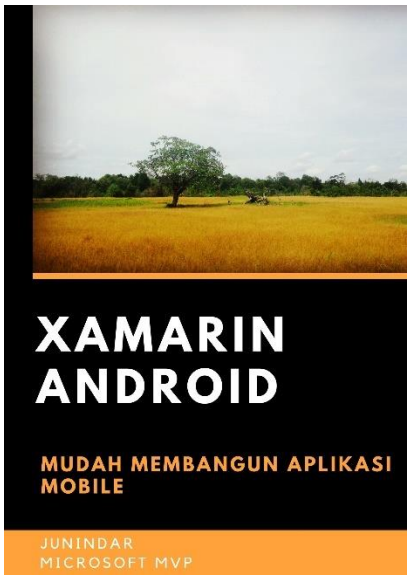
Lalu untuk melihat hasilnya jalankan program, dan pastikan mendapatkan hasil seperti pada gambar dibawa ini.



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/open-apiswagger-pada-web-api-aspnet-core.html

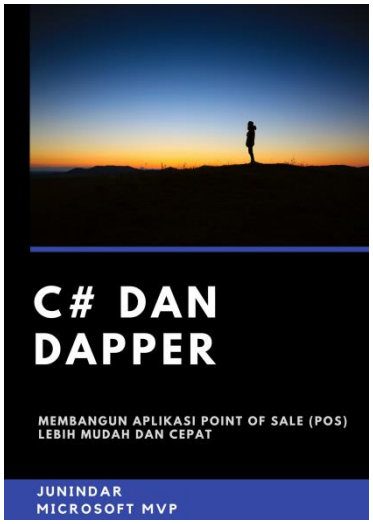
Referensi



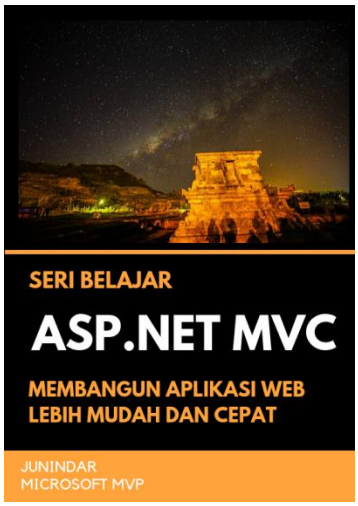
https://play.google.com/store/books/details?id=G4tFDgAAQBAJ

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

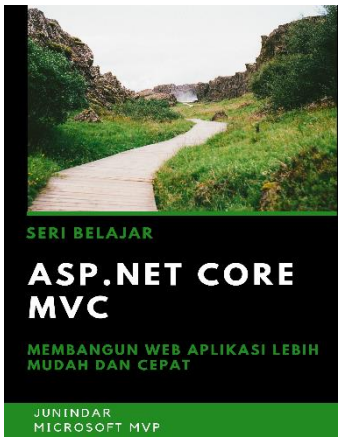
https://play.google.com/store/books/details/Junindar_Xamarin_Forms?id=6Wg-DwAAQBAJ



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".**