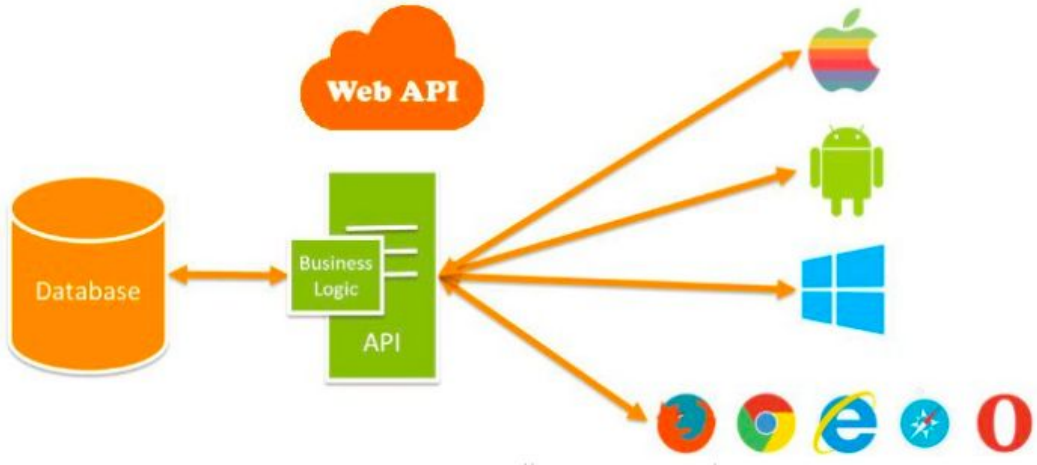
Back End Development 1
sesi 9

REST V
RESTFUL API

Introduction API, REST API & RESTFUL API

Sesi kemarin kita sudah mengenal WEB Services dan bagaimana cara meng-aplikasi-kan nya dengan Asp.NET .Nah kali ini kita akan masuk lebih dalam terkait API. Dimana Ada REST dan RESTFUL API.

Apa itu WEB API ?



Sources : https://tutorialshelper.com/

API

API = Application Programming Interface ;
yaitu sebuah software yang memungkinkan para developer untuk mengintegrasikan dan mengizinkan dua aplikasi yang berbeda secara bersamaan untuk saling terhubung satu sama lain.

Tujuan penggunaan dari API adalah
A. untuk saling berbagi data antar aplikasi yang berbeda tersebut
B. untuk mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah function yang terpisah sehingga para developer tidak perlu lagi membuat fitur yang serupa.

Istilah “API” sebetulnya mirip dengan web service, yang tentunya sudah kita pelajari sebelumnya.

WEB API v WEB SERVICES

Web API adalah aplikasi atau antarmuka untuk menghubungkan aplikasi satu dengan aplikasi yang lain pada sebuah sistem berbasis website. Sedangkan web service adalah bentuk layanan yang diberikan melalui platform berbasis website untuk menghubungkan aplikasi yang berbeda. Berikut merupakan pemaparan dari perbedaan kedua istilah tersebut.

- Setiap jenis web service menggunakan API, akan tetapi tidak semua jenis API menggunakan web service.
- Web service berperan untuk memfasilitasi untuk proses interaksi antara dua perangkat atau aplikasi melalui jaringan. Sedangkan API berperan untuk menghubungkan komunikasi antara dua aplikasi yang berbeda platform baik dengan membutuhkan jaringan maupun tidak.
- API dapat menggunakan arsitektur jenis apapun, sedangkan web service hanya menggunakan tiga arsitektur (REST, SOAP, XML-RPC).
- API tidak membutuhkan jaringan dalam proses pengoperasiannya, sedangkan web service sangat bergantung pada sebuah jaringan.

REST v RESTFUL API

RESTful API / REST API merupakan penerapan dari API (Application Programming Interface).

Sedangkan

REST (Representional State Transfer) adalah sebuah arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dimana metode ini sering diterapkan dalam pengembangan aplikasi.

Dengan tujuannya untuk menjadikan sistem memiliki performa yang baik, cepat dan mudah untuk di kembangkan (scale) terutama dalam pertukaran dan komunikasi data.

Sama seperti Web Services API, RESTFUL API Memiliki 4 komponen penting yaitu :
A. URL Design
B. HTTP Verbs
C. HTTP Response Code
D. Format Response.

Penjelasan

A. URL Design
Menggunakan Protokol HTTP sehingga Penamaan dan Struktur URL yang konsisten akan menghasilkan API yang mudah dimengerti oleh developer. URL API disebut dengan ENDPOINT dalam pemanggilannya.

B. HTTP Verbs
Sudah Dibahas di sesi sebelumnya, Metode yang dipakai agar server mengerti apa yang sedang di-Request client : GET, POST, PUT, DELETE.

C. HTTP Response Code
Standarisasi Kode untuk memberikan informasi hasil request kepada client.
- 2XX = Request Berhasil
- 4XX = Ada Kesalahan Request dari sisi Client
- 5XX = Kesalahan Request pada sisi Server.

D. Format Response
Setiap request yang dilakukan client akan menerima data response dari server, response tersebut biasanya berupa data XML ataupun JSON. Setelah mendapatkan data response tersebut barulah client bisa menggunakannya dengan cara memparsing data tersebut dan diolah sesuai kebutuhan.

API Dapat dikatakan RESTFUL jika memiliki fitur :

1. Client-server = client handle front-end dan server handle back-end sehingga kedua nya dapat diganti secara independen.

2. Stateless = Tidak ada data Client yang disimpan dalam server baik itu ketika ada request dan session disimpan di klien.

3. Cacheable = Client dapat men-cache respons untuk meningkatkan performa aplikasi.

WEB API - ASP.NET Core

REST v RESTFUL API - Sesi 09

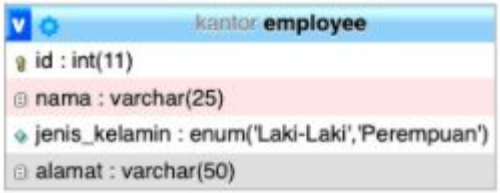
BUILDING WEB API with ASP.NET Core

Sesi ini kita akan Practice buat Web API yang akan tersambung dengan Database MySQL, dan

Persiapkan Tools :
1. Visual Studio
2. Local Web Server via Azure Data Studio

Pertama-tama buat database terlebih dahulu :

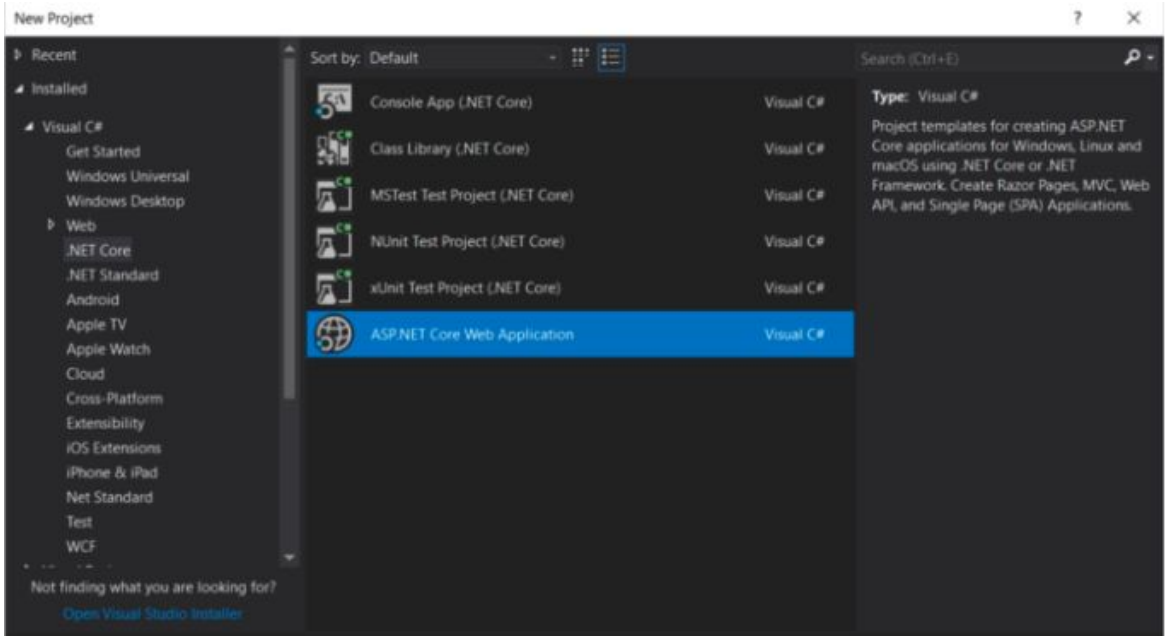
Nama database : kantor
Nama table : employee



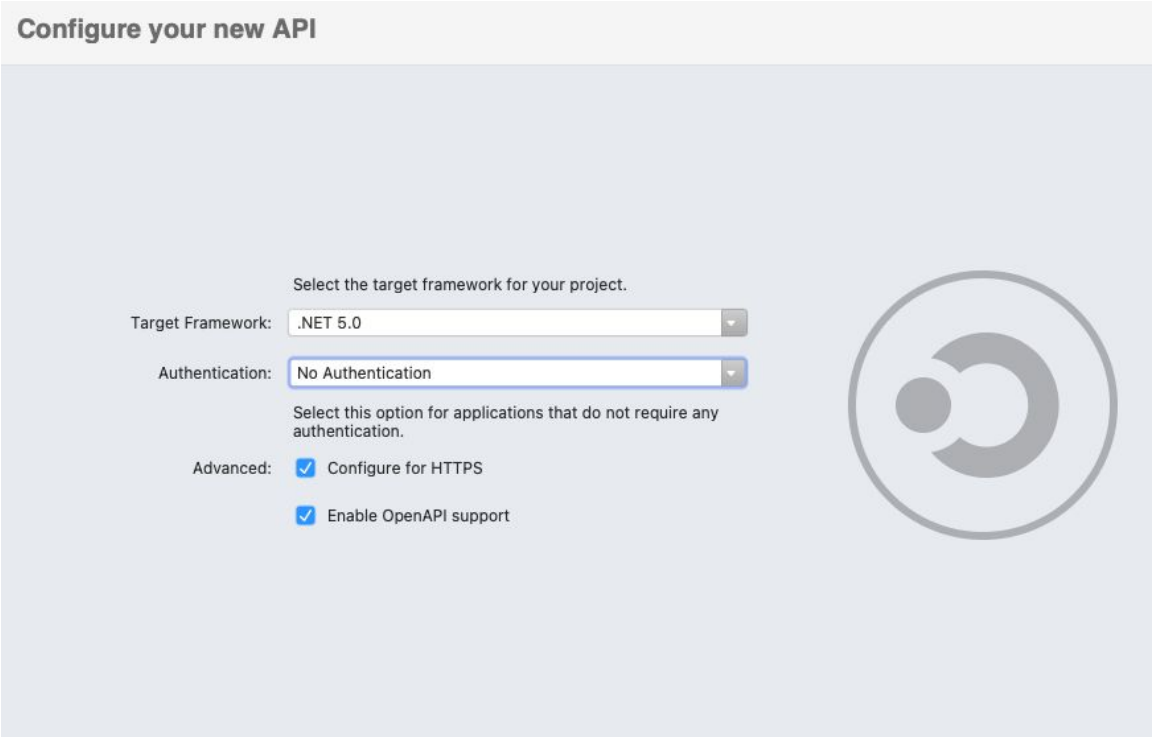
Dan jangan lupa data record ke dalam Table employee yang nantinya bakal Di Request oleh WEB API.

id	nama	jenis_kelamin	alamat
1	John Doe	Laki-Laki	Pagedangan
2	Alicia Jordan	Perempuan	Surabaya
3	Smith	Laki-Laki	Manado
4	James Dron	Laki-Laki	Jakarta

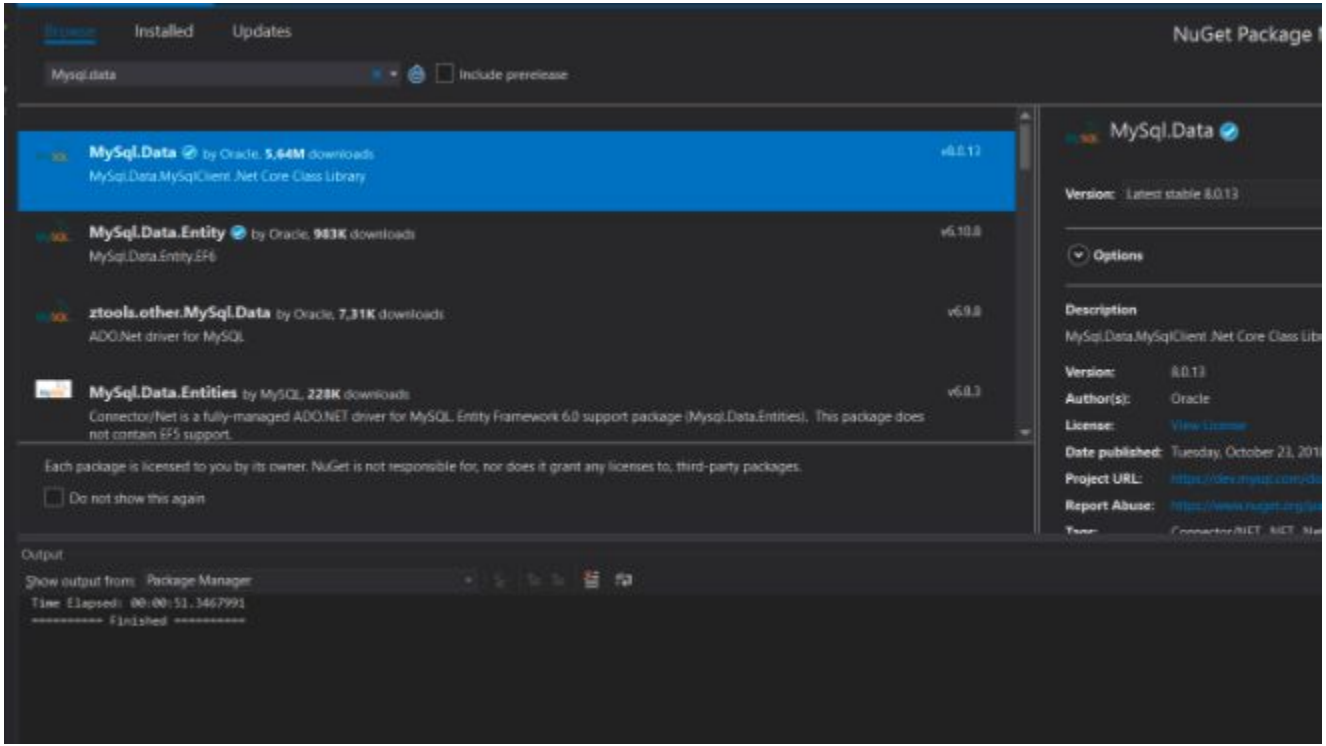
HACKTIV8



Buat project baru di Visual Studio.
Klik File → New → Project..,
lalu pada menu di sebelah kiri pilih Visual C# → Web → .NET Core.
Selanjutnya pilih ASP.NET Core Web Application.
Project ini kita beri nama Kantor_WebAPI.



Selanjutnya bakal muncul jendela baru berisikan template Web Application yang akan kita bangun, pilih API, dan pastikan aplikasi yang dibangun itu merupakan aplikasi .NET Core dan menggunakan ASP.Net Core 5.0



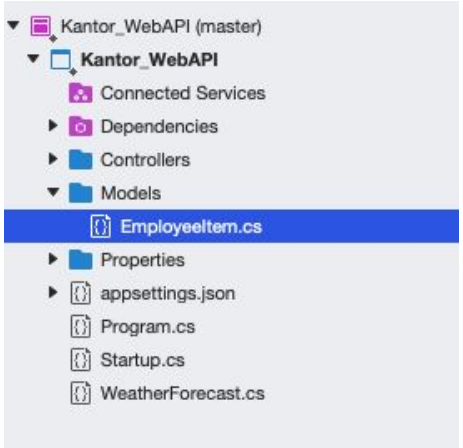
Project baru saja kita create, selanjutnya kita akan menginstall package MySQL.Data menggunakan NuGet.
Klik kanan project → Manage NuGet Packages.
Kita bisa mencari paket MySql.Data dan dapat langsung menginstallnya.

HACKTIV8

Setelah membuat project baru, struktur project masih berupa satu folder Controllers, maka dari itu kita perlu membuat satu folder Models yang berfungsi sebagai penghubung antara database dengan web service yang kita bangun.

Maka dari itu klik kanan project→Add→New Folder dan beri nama folder Models.

Selanjutnya kita perlu membuat sebuah class yang berguna untuk mendeklarasikan entitas yang akan diterima dari database.



Dalam hal ini entitas yang akan diterima oleh aplikasi yaitu Entitas Employee.
Entitas employee akan ditampung didalam Class EmployeeItem.cs.

Caranya dengan Klik kanan Folder Models → Add → Class.. Lalu beri nama Class EmployeeItem

EmployeeItem.cs

No selection

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5
6   namespace Kantor_WebAPI.Models
7   {
8       public class EmployeeItem
9       {
10          private Models.EmployeeContext context;
11
12          public int id { get; set; }
13          public string nama { get; set; }
14          public string jenisKelamin { get; set; }
15          public string alamat { get; set; }
16      }
17  }
```

EmployeeItem.cs

Class EmployeeItem terdapat field-field yang akan menampung atribut-atribut yang diterima dari database, namun kita belum membuat sebuah class yang berperan dalam melakukan komunikasi dan request kepada database.

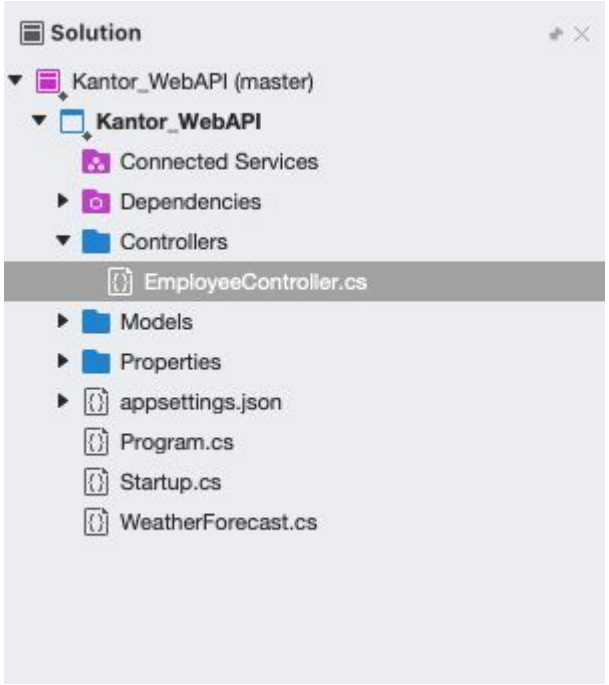
Oleh karena itu selanjutnya kita akan membuat sebuah class dengan nama EmployeeContext yang nantinya akan melakukan request dengan database. Klik kanan pada folder Models → Add → Class lalu class diberi nama EmployeeContext.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Kantor2_WebAPI.Models
{
    public class EmployeeContext
    {
        public string ConnectionString { get; set; }

        public EmployeeContext(string connectionString)
        {
            this.ConnectionString = connectionString;
        }

        private MySqlConnection GetConnection()
        {
            return new MySqlConnection(ConnectionString);
        }

        public List<EmployeeItem> GetAllEmployee()
        {
            List<EmployeeItem> list = new List<EmployeeItem>();

            using (MySqlConnection conn = GetConnection())
            {
                conn.Open();
                MySqlCommand cmd = new MySqlCommand("SELECT * FROM employee", conn);
                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        list.Add(new EmployeeItem()
                        {
                            id = reader.GetInt32("id"),
                            nama = reader.GetString("nama"),
                            jenisKelamin = reader.GetString("jenis_kelamin"),
                            alamat = reader.GetString("alamat")
                        });
                    }
                }
            }
            return list;
        }
```

```
47
48          public List<EmployeeItem> GetEmployee(string id)
49          {
50              List<EmployeeItem> list = new List<EmployeeItem>();
51
52              using (MySqlConnection conn = GetConnection())
53              {
54                  conn.Open();
55                  MySqlCommand cmd = new MySqlCommand("SELECT * FROM employee WHERE id =@id", conn);
56                  cmd.Parameters.AddWithValue("@id", id);
57
58                  using (MySqlDataReader reader = cmd.ExecuteReader())
59                  {
60                      while (reader.Read())
61                      {
62                          list.Add(new EmployeeItem()
63                          {
64                              id = reader.GetInt32("id"),
65                              nama = reader.GetString("nama"),
66                              jenisKelamin = reader.GetString("jenis_kelamin"),
67                              alamat = reader.GetString("alamat")
68                          });
69                      }
70                  }
71              }
72              return list;
73          }
74      }
75  }
```

Fungsi GetAllEmployee() merupakan fungsi yang digunakan untuk mendapatkan seluruh data employee sedangkan GetEmployee() merupakan fungsi yang digunakan untuk mendapatkan data employee berdasarkan id employee di database.

Selanjutnya kita akan membuat Sebuah Class Controller yang digunakan untuk mengolah data yang masuk melalui EmployeeItem dan EmployeeContext.



Pada file launchSettings.json kita harus ngerubah atribut launchUrl dari api/values jadi api/employee. Ini dilakukan supaya ketika aplikasi dijalankan nantinya akan langsung menjalankan EmployeeController.cs.

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:31936",
      "sslPort": 44396
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "api/employee",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "Kantor_WebAPI": {
      "commandName": "Project",
      "dotnetRunMessages": "true",
      "launchBrowser": true,
      "launchUrl": "api/employee",
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using Kantor2_WebAPI.Models;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace Kantor2_WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmployeeController : ControllerBase
    {
        private EmployeeContext _context;

        public EmployeeController(EmployeeContext context)
        {
            this._context = context;
        }

        // GET: api/User
        public ActionResult<IEnumerable<EmployeeItem>> GetEmployeeItems()
        {
            _context = HttpContext.RequestServices.GetService(typeof(EmployeeContext)) as EmployeeContext;
            //return new string[]
            return _context.GetAllEmployee();
        }

        // Get : api/user/{id}
        [HttpGet("{id}", Name = "Get")]
        public ActionResult<IEnumerable<EmployeeItem>> GetEmployeeItem(String id)
        {
            _context = HttpContext.RequestServices.GetService(typeof(EmployeeContext)) as EmployeeContext;
            return _context.GetEmployee(id);
        }
    }
}
```

EmployeeController.cs

HACKTIV8

Atribut [Route("api/[controller]")] memiliki fungsi untuk menjadi indeks atau acuan Web API dalam membuat URI dalam mengakses controller.
Sedangkan atribut [ApiController] memiliki fungsi sebagai penanda kalo class EmployeeController itu merupakan sebuah Class API Controller. Atribut [ApiController] mulai ada sejak ASP .NET Core versi 2.1.

Atribut [HttpGet] itu merupakan sebuah atribut yang akan memerintahkan sebuah method atau subrutin yang nantinya akan dieksekusi sebagai method Get.

GetEmployeeItems() dan GetEmployeeItem() merupakan dua buah method dengan jenis GET yang akan merequest lalu menerima data employee kepada database, namun bedanya GetEmployeeItems() akan menerima seluruh data employee yang ada dalam database sedangkan GetEmployeeItem() hanya akan menerima satu buah data employee yang dipilih berdasarkan Id nya saja.

Sekarang kita waktunya untuk menambahkan konfigurasi database ConnectionStrings yang akan ditulis di dalam file appsettings.json

```
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "Server=localhost;Port=3306;Database=kantor;Uid=root;Pwd="
4    },
5
```

appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Port=3306;Database=kantor;Uid=root;Pwd="
  },

  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

appsettings.json

Terakhir, kita tambahkan service pada file startup.cs di method ConfigureServices() :

```
26      public void ConfigureServices(IServiceCollection services)
27      {
28
29          services.AddControllers();
30          services.Add(new ServiceDescriptor(typeof(Models.EmployeeContext), new Models.EmployeeContext(Configuration.GetConnectionString("DefaultConnection"))));
31      }
```

Nah silahkan klik run project kita.