

ASP.NET Web APIs

ASP.NET Core Web API

פרויקט חדש – סוג הפרוייקט:



ASP.NET Core Web API

A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optio

C#

Linux

macOS

Windows

API

Cloud

Service

Web

Web API

Additional information

ASP.NET Core Web API

C#

Linux

macOS

Windows

API

Cloud

Service

Web

Web API

Framework [i](#)

.NET 8.0 (Long Term Support)

Authentication type [i](#)

None

☒ Configure for HTTPS [i](#)

☐ Enable container support [i](#)

Container OS [i](#)

Linux

Container build type [i](#)

Dockerfile

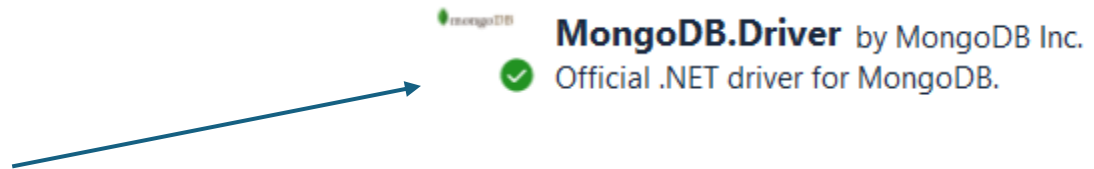
☒ Enable OpenAPI support [i](#)

☒ Do not use top-level statements [i](#)

☒ Use controllers [i](#)

☐ Enlist in .NET Aspire orchestration [i](#)

התקנת חבילות NuGet



אחסון נוח של Connection String

WeatherForecastController.cs appsettings.json ×

Schema: <https://json.schemastore.org/appsettings.json>

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft.AspNetCore": "Warning"
6      }
7    },
8    "ConnectionStrings": {
9      "MongoDBConnectionString": "mongodb://127.0.0.1:27017/iloved
10    },
11    "AllowedHosts": "*"
12  }
13
```

Dependency Injection

DI Container

בנק של אובייקטים שנרצה
להשתמש בהם בתוכנית

אפשר לגשת לאובייקטים
ממחלקות אחרות בתכנית

בקובץ program.cs הוא נקודת הכניסה לתוכנית.

אפשר להוסיף אובייקטים לבנק.

ע"י המתודות builder.Services.add...

את האובייקטים במאגר
אפשר ל"קבל" (מוזרקים) בכל מקום בפרוייקט.

גישה ל Connection String מאיפה שרוצים:
:DI

הבנאי של הController

```
public WeatherForecastController(IConfiguration con)
{
    var str = con.GetConnectionString("MongoDBConnectionString");
    Console.WriteLine(str);
}
```

Service שיחזיק את החיבור לדטה-בייס פתוח

```
namespace WebAppMongo.Service
{
    4 references
    public class MongoService
    {
        private readonly IMongoDatabase _database;

        0 references
        public MongoService(IConfiguration config)
        {
            var connectionString = config.GetConnectionString("MongoDBConnectionString");
            var client = new MongoClient(connectionString);
            _database = client.GetDatabase(config["DatabaseName"]);
        }

        2 references
        public IMongoCollection<T> GetCollection<T>(string name)
        {
            return _database.GetCollection<T>(name);
        }
    }
}
```

הגדרת השירות שלנו כסינגלטון בקונטיינר של DI

```
public class Program
{
    0 references
    public static void Main(string[] args)
    {
        var builder = WebApplication.CreateBuilder(args)

        // Add services to the container.

        builder.Services.AddControllers();

        builder.Services.AddSingleton<MongoService>();
        // Learn more about configuring Swagger/OpenAPI
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();
    }
}
```


מודל לעבודה עם Mongo לטובת סריליזציה אוטומטית לBsonDocument

```
namespace Models
{
    public class Person
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string? Id { get; set; }
        public string Name { get; set; } = string.Empty;
    }
}
```

כך נוכל להשתמש בService שלנו בקונטרולר:

```
public class WeatherForecastController : ControllerBase
{
    public MongoService MongoService { get; set; }

    public WeatherForecastController(MongoService ms)
    {
        MongoService = ms;
    }

    [HttpGet(Name = "GetWeatherForecast")]
    public IActionResult Get()
    {
        MongoService.GetCollection<Person>("People").InsertOne(new Person { Name = "John Doe" });
        return Ok(MongoService.GetCollection<Person>("People").Find(p => true).ToList());
    }
}
```

Constructor injection

בדיקה בקובץ http

```
@WebAppMongo_HostAddress = http://localhost:5187
```

```
GET {{WebAppMongo_HostAddress}}/weatherforecast/  
Accept: application/json
```

```
###
```

בדיקה עם Postman



Weather / GET Weather Forecast

GET



{{base}}/WeatherForecast

Params

Authorization

Headers (6)

Body

Query Params

	Key
	Key

בקשת Post להוספה של אובייקט חדש לאוסף People בדטה-בייס

```
[HttpPost]
```

0 references

```
public IActionResult PostPerson([FromBody] Person person)
{
    people.InsertOne(person);
    return Ok(person);
}
```

בקשת GET להצגה של כל האובייקטים באוסף People בדטה-בייס

```
[HttpGet]
```

```
0 references
```

```
public IActionResult Get()
```

```
{
```

```
    return Ok(people.Find(_ => true).ToList());
```

```
}
```

בקשת GET להצגה של אובייקט אחד לפי ID מהאוסף People בדטה-בייס

```
[HttpGet("{id}")]
```

0 references

```
public IActionResult Get(string id)
{
    var person = people.Find(p => p.Id == id).FirstOrDefault();
    if (person is null)
    {
        return NotFound();
    }
    return Ok(person);
}
```

בקשת DELETE למחיקה של אובייקט אחד לפי ID מהאוסף People בדטה-בייס

```
[HttpDelete("{id}")]
0 references
public IActionResult Delete(string id)
{
    var result = people.DeleteOne(p => p.Id == id);

    if (result.DeletedCount == 0)
    {
        return NotFound();
    }
    return NoContent();
}
```


בקשת PUT לעדכון של אובייקט אחד לפי ID מהאוסף People בדטה-בייס

```
[HttpPut("{id}")]
```

0 references

```
public IActionResult Put(string id, Person p)
{
    var result = people.ReplaceOne(person => person.Id == id, p);

    if (result.ModifiedCount == 0)
    {
        return NotFound();
    }
    return Ok(p);
}
```

בדיקות עם קובץ http.
בדיקות לבקשות GET ו POST

```
@BASE = http://localhost:5066
```

```
Send request | Debug  
GET {{BASE}}/people/
```

```
### Add A Person:  
Send request | Debug  
POST {{BASE}}/people/  
Content-Type: application/json
```

```
{  
  "name": "Mickey Doe"  
}
```

בדיקות עם קובץ http.
בדיקות לבקשות GET BY ID, PUT ו-DELETE

Get A Person By Id:

Send request | Debug

GET {{BASE}}/people/6697f7d53c4080064f5bb6fb

Delete a Person By Id:

Send request | Debug

DELETE {{BASE}}/people/6697f7bf3c4080064f5bb6fa

Update a Person By Id:

Send request | Debug

PUT {{BASE}}/people/6697f7d53c4080064f5bb6fb

Content-Type: application/json

```
✓ {  
  "id": "6697f7d53c4080064f5bb6fb",  
  "name": "Mickey Mouse"  
}
```

תרגיל:

(2) צרו Controller בשם MovieController
הזריקו בבנאי את ה Service של מונגו ושמרו אצלכם בקונטרולר
אוסף לסרטים IMongoCollection<Movie>

(1) צרו מחלקה בשם Movie
תכונות: id, title, description
שימו לב לAttribute של [BsonID]

(4) בדקו עם SwaggerUI
ועם קובץ http.

(3) בקונטרולר יש להגיב לבקשות
POST, GET, Get by id, PUT, DELETE

SOLID Principles

Single responsibility principle


עקרון האחריות הבודדת – למחלקה/מתודה
יש מטרה אחת בלבד – סיבה אחת להשתנות.

אם צריך לבצע שינוי – יש מקום אחד לבצע אותו ולא לסרוק את כל הפרוייקט ולשנות
בהרבה מקומות.

מאוד מקל על בדיקות תוכנה – בדיקות יחידה.
מקל על קריאות של הקוד למתכנתים אחרים בצוות.

שיעורי בית

ממשו את ה Controller הבא תוך שימוש ב MongoDB

 **swagger**

http://localhost:60201/swagger/v1/swagger.json

My API V1 ▾

My API

Todo

Show/Hide | List Operations | Expand Operations

GET	/api/Todo
POST	/api/Todo
DELETE	/api/Todo/{id}
GET	/api/Todo/{id}
PUT	/api/Todo/{id}

[BASE URL: / , API VERSION: v1]

שיעורי בית

ממשו את הController הבא תוך שימוש בMongoDB

PetStore

pet

Show/Hide | List Operations | Expand Operations

GET	/api/pet
POST	/api/pet
DELETE	/api/pet/{id}
PUT	/api/pet/{id}