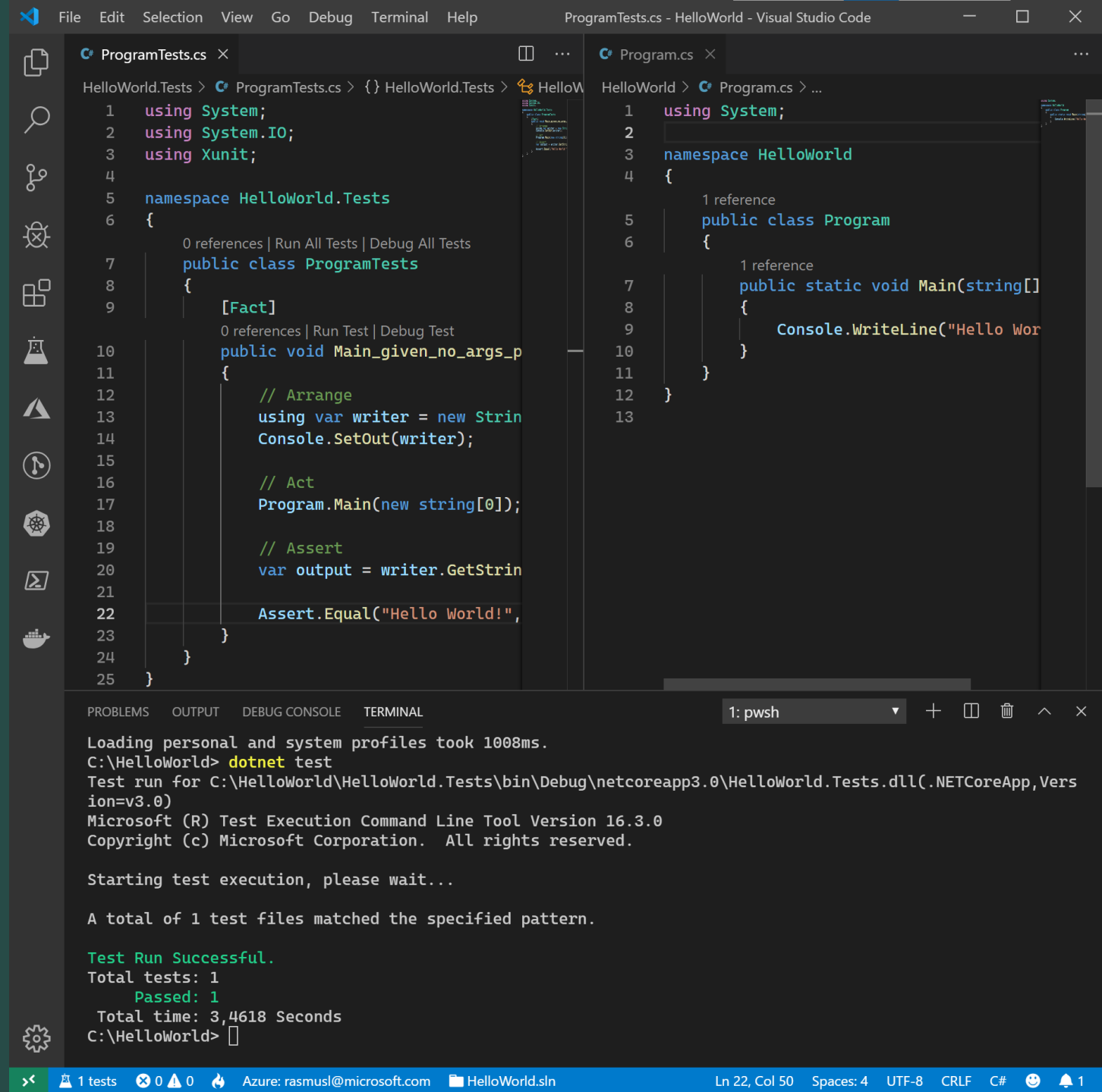


JSON and the REST ASP.NET Core

Rasmus Lystrøm
Associate Professor
ITU
rne@itu.dk



The screenshot shows the Visual Studio Code editor with two files open: `ProgramTests.cs` and `Program.cs`. The `ProgramTests.cs` file contains a test class `ProgramTests` with a single test method `Main_given_no_args_p` that uses `Assert.Equal` to verify the output of `Program.Main`. The `Program.cs` file contains a simple `Program` class with a `Main` method that writes "Hello World!" to the console. The terminal at the bottom shows the output of running the tests, indicating that the test was successful.

```
ProgramTests.cs
1 using System;
2 using System.IO;
3 using Xunit;
4
5 namespace HelloWorld.Tests
6 {
7     0 references | Run All Tests | Debug All Tests
8     public class ProgramTests
9     {
10         [Fact]
11         0 references | Run Test | Debug Test
12         public void Main_given_no_args_p
13         {
14             // Arrange
15             using var writer = new StringWriter();
16             Console.SetOut(writer);
17
18             // Act
19             Program.Main(new string[0]);
20
21             // Assert
22             var output = writer.GetStringBuilder().ToString();
23             Assert.Equal("Hello World!", output);
24         }
25     }
26 }
```

```
Program.cs
1 using System;
2
3 namespace HelloWorld
4 {
5     1 reference
6     public class Program
7     {
8         1 reference
9         public static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }
14 }
```

```
Terminal
1: pwsh
Loading personal and system profiles took 1008ms.
C:\HelloWorld> dotnet test
Test run for C:\HelloWorld\HelloWorld.Tests\bin\Debug\netcoreapp3.0\HelloWorld.Tests.dll (.NETCoreApp, Version=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 1
Passed: 1
Total time: 3,4618 Seconds
C:\HelloWorld>
```

Agenda

XML

SOAP

JSON

YAML

REST

ASP.NET Core

Web API with ASP.NET Core

gRPC

XML

XML

eXtensible Markup Language

Markup language like HTML

Designed to carry data, not to display data

Tags are not predefined – You must define your own tags

Designed to be self-descriptive

XML does not do anything

XML was created to structure, store, and transport information

```
<?xml version="1.0" encoding="UTF-8"?>
<note id="1">
  <to>Kathleen B.</to>
  <from>Tom W.</from>
  <subject>Reminder</subject>
  <body>Don't forget small change!</body>
</note>
```

How Can XML be Used?

Separates data from HTML

Simplifies data sharing

Simplifies data transport

Simplifies platform changes

Used to create new (Internet) languages

- XHTML

- WSDL for describing web services

- RSS and ATOM for news feeds

- XAML

SOAP

SOAP

SOAP request

POST https://store.com/services/PriceService/getPrice

```
<?xml version="1.0"?>
```

```
<soap:Envelope  
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
```

```
  <soap:Body>  
    <m:GetPrice xmlns:m="https://www.w3schools.com/prices">  
      <m:Item>Apples</m:Item>  
    </m:GetPrice>  
  </soap:Body>
```

```
</soap:Envelope>
```

Source: https://www.w3schools.com/xml/xml_soap.asp

SOAP response

```
<?xml version="1.0"?>
```

```
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
```

```
<soap:Body>  
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">  
    <m:Price>1.90</m:Price>  
  </m:GetPriceResponse>  
</soap:Body>
```

```
</soap:Envelope>
```

JSON

JSON

JavaScript Object Notation

Lightweight text-data interchange format

Language independent (uses JavaScript syntax)

"Self-describing" and easy to understand

JSON Syntax

(subset of the JavaScript object notation syntax)

Data is in name/value pairs

Data is separated by commas

Curly braces hold objects

Square brackets hold arrays

JSON Name/Value Pairs

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

"firstName": "John"

This is simple to understand, and equals to the JavaScript statement:

firstName = "John"

JSON Data Types

Number (integer or floating point)

String (in double quotes)

Boolean (true or false)

Array (in square brackets)

Object (in curly brackets)

null

Examples

```
{ "alterEgo": "Batman", "firstAppearance": 1939 }
```

```
[  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Jane", "lastName": "Doe" },  
  { "firstName": "John", "lastName": "Smith" }  
]
```

```
null
```

```
"John Doe"
```


Best of both worlds?

https://www.ibm.com/support/knowledgecenter/en/SS9H2Y_7.7.0/com.ibm.dp.doc/json_jsonxconversionexample.html

YAML

YAML

YAML Ain't Markup Language

Human-readable data-serialization language.

It is commonly used for configuration files and in applications where data is being stored or transmitted.

Superset of JSON

Examples

people:

- first_name: John
last_name: Doe
- first_name: Jane
last_name: Doe
- first_name: John
last_name: Smith

REST

REpresentation

http://

HTTP request

URI: string

(Query): key/value pairs

Method: string

Header: key/value pairs

Body: string/binary

HTTP response

Status-Code: number

Header: key/value pairs

Body: string/binary

REST

Maps your CRUD actions to HTTP verbs

Action	Verb
Create	POST
Read (Retrieve)	GET
Update (Replace)	PUT
Update (Modify)	PATCH
Delete	DELETE

HTTP status codes

Code	Meaning
200	OK
201	Created
202	Accepted
204	No Content
301	Moved Permanently
302	Found (Previously "Moved temporarily")
307	Temporary Redirect
308	Permanent Redirect

Code	Meaning
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
409	Conflict
415	Unsupported Media Type
418	I'm a Teapot
422	Unprocessable Entity
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

RFC2324: Hyper Text Coffee Pot Control Protocol

In practice, most automated coffee pots cannot currently provide additions.

2.3.2 418 I'm a teapot

Any attempt to brew coffee with a teapot should result in the error code "418 I'm a teapot". The resulting entity body MAY be short and stout.

3. The "coffee" URI scheme

Because coffee is international, there are international coffee URI schemes. All coffee URL schemes are written with URL encoding of the UTF-8 encoding of the characters that spell the word for "coffee" in any of 29 languages, following the conventions for internationalization in URIs [[URLI18N](#)].

```
coffee-url = coffee-scheme ":" [ "//" host ]  
             [ "/" pot-designator ] [ "?" additions-list ]
```

Source: <https://tools.ietf.org/html/rfc2324>

HTTP headers

Header Field	Description	Examples
Accept	Content-Types that are acceptable for the response	text/plain application/json application/xml
Content-Type	The media type of the body of the request (POST, PUT, and PATCH)	application/x-www-form-urlencoded application/json; charset=utf-8
Authorization	Authentication credentials for HTTP authentication	Bearer ey...

Why REST?

Simple, both conceptually and programmatically

Simpler and cleaner than SOAP

REST is the new black

REST request

GET `https://store.com/prices/Apples`

REST response

1.9

HTTP request

URI: <https://futurama.com/api/characters>

Method: POST

Header:

Content-Type: application/json; charset=utf-8

Authorization: Bearer ey...

Body:

```
{  
  "name": "Bender",  
  "species": "Robot",  
  "planet": "Earth"  
}
```

HTTP response

Status-Code: 201

Header:

Content-Type: application/json; charset=utf-8

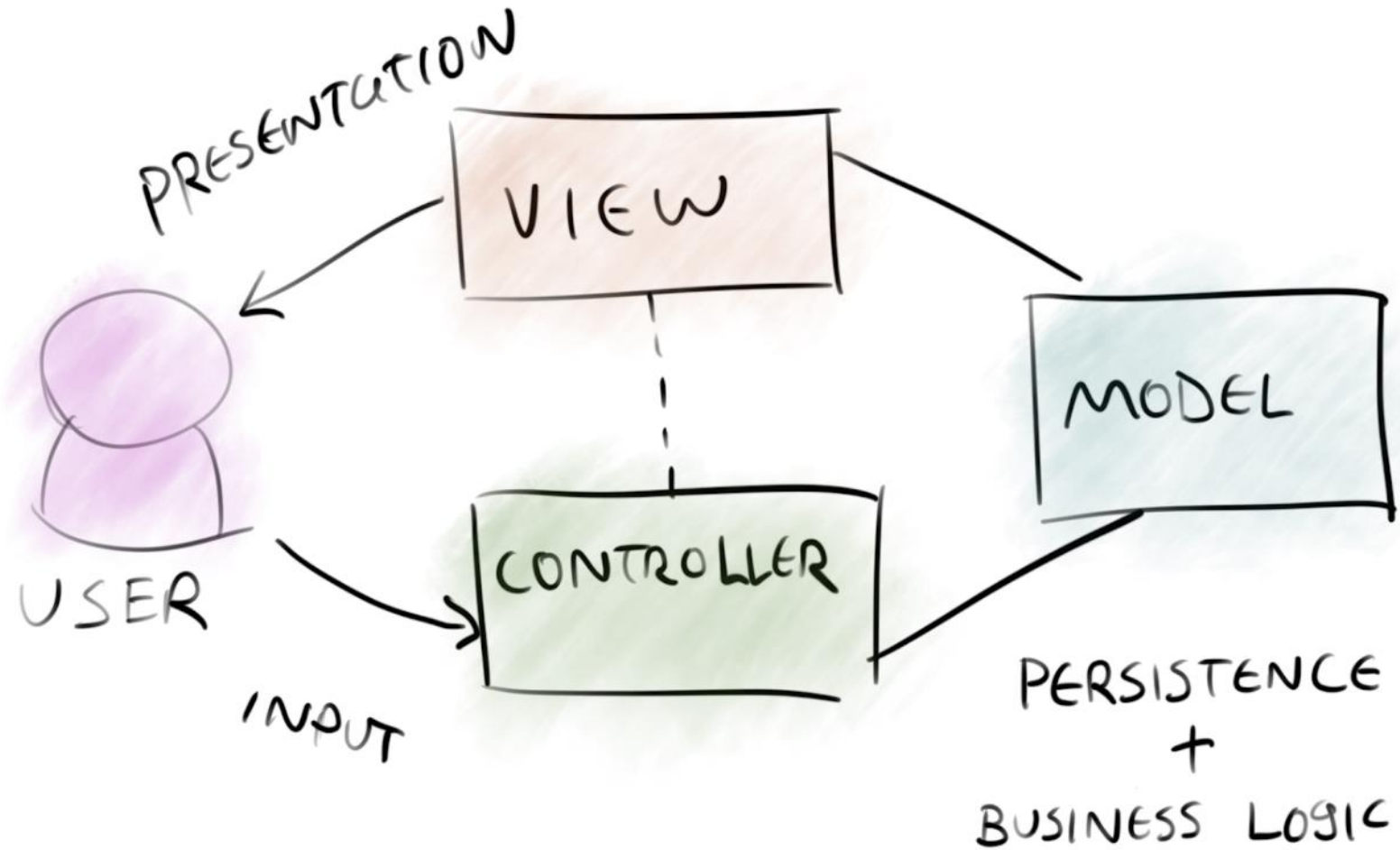
Location: <https://futurama.com/api/characters/42>

Body:

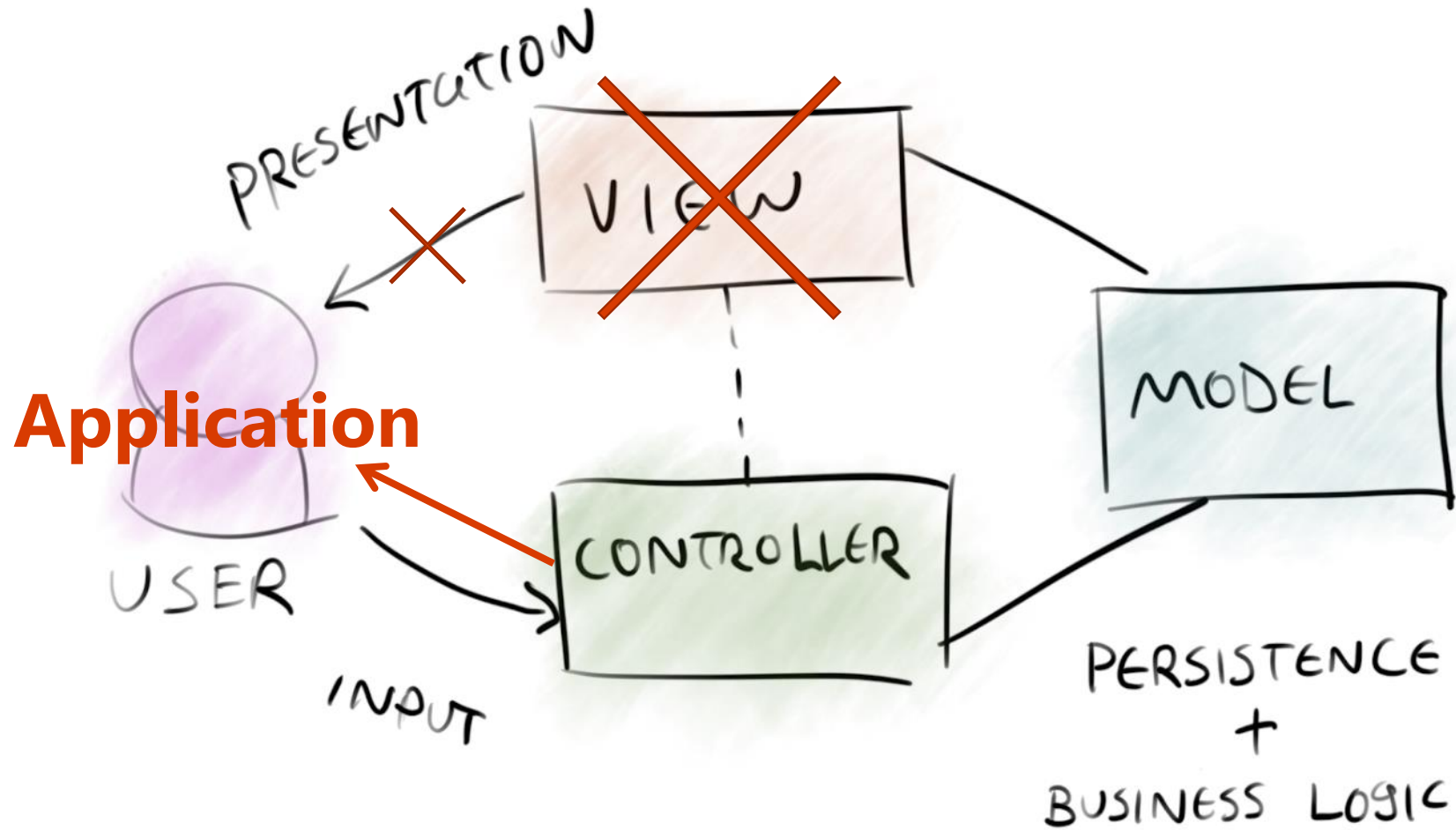
```
{  
  "id": 42,  
  "name": "Bender",  
  "species": "Robot",  
  "planet": "Earth"  
}
```


ASP.NET Core

Model – View – Controller



ASP.NET Web API



ASP.NET Core Web API

Demo

Controller

Class

Derive from ControllerBase

Decorate with [ApiController] and [Route("[controller]")]

Method

Decorate with [HttpGet], [HttpPost], [HttpPut], [HttpDelete]

Return *specific type*, IActionResult, or ActionResult<T> - or async Task<...>.

ASP.NET (Web API) best practices

Be RESTful

Use proper status codes

Use meaningful routes

Use proper HTTP methods

Don't throw exceptions

Use *user secrets* in development

```
dotnet user-secrets init
```

```
dotnet user-secrets set "ConnectionStrings:<connectionstring-name>" "<connection-string>"
```


Use built-in IoC container

```
services.AddDbContext<MyContext>(o =>  
    o.UseSqlServer(Configuration.GetConnectionString("MyConnectionStringName"))  
);  
services.AddScoped<IMyContext, MyContext>();  
services.AddScoped<IMyRepository, MyRepository>();
```

Consider running migrations on load

```
using (var serviceScope = app.ApplicationServices
    .GetRequiredService<IServiceScopeFactory>().CreateScope())
{
    var context = serviceScope.ServiceProvider.GetService<SuperheroContext>();
    context.Database.Migrate();
}
```

Support the OpenAPI Specification (Swagger)

```
dotnet add package Swashbuckle.AspNetCore
```

```
using Microsoft.OpenApi.Models;
```

```
services.AddSwaggerGen(c =>  
{  
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });  
});
```

```
app.UseSwagger();  
app.UseSwaggerUI(c =>  
{  
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");  
    c.RoutePrefix = string.Empty;  
});
```

Support HTTPS *only*

```
dotnet dev-certs https --trust
```

```
app.UseHttpsRedirection();
```

Standardize on lowercase urls

```
services.AddRouting(options => options.LowercaseUrls = true);
```

Secure your Web API

Azure AD (later)

Azure AD B2C

Other 3

Do not write your own security layer!!!

gRPC

gRPC Remote Procedure Calls

gRPC is a modern, open source, high-performance remote procedure call (RPC) framework that can run anywhere. gRPC enables client and server applications to communicate transparently, and simplifies the building of connected systems.

```
$ dotnet new grpc
```