Workshop 3 – Building a RESTful API

Agenda

- Create a RESTful API backend

Platforms

- .NET - ASP.NET WebAPI
- Or .NET Core – ASP.NET Core WebAPI
- Couchbase (document database)

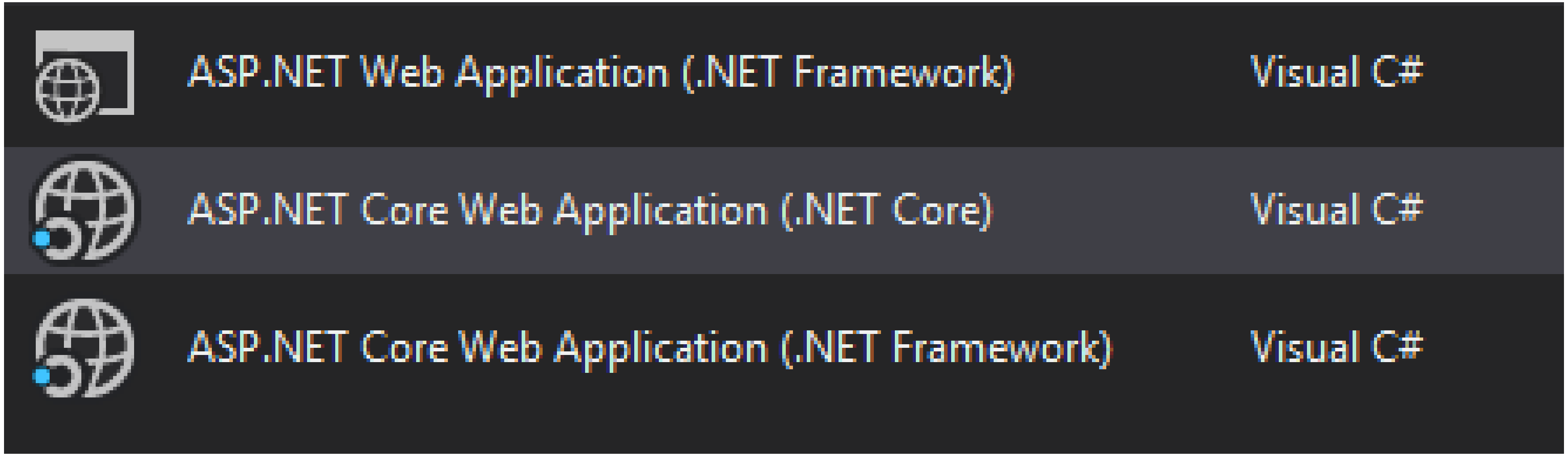
ASP.NET

- Relies on IIS
- Controllers contain methods
- Global.asax.cs

ASP.NET Core

- Runs as a "command line" app, uses Kestrel server
- Controllers contain methods
- Can run on .NET or .NET Core

ASP.NET vs ASP.NET Core



ASP.NET Web Application (.NET Framework)	Visual C#
ASP.NET Core Web Application (.NET Core)	Visual C#
ASP.NET Core Web Application (.NET Framework)	Visual C#

ASP.NET WebAPI

```
[RoutePrefix("api")]
[EnableCors(origins: "*", headers: "*", methods: "*")]
public class PersonController : ApiController
{
    [HttpGet]
    [Route("get/{id?}")]
    public async Task<IHttpActionResult> Get(string id = null)
    {
        if (string.IsNullOrEmpty(id))
            return BadRequest("Missing or empty 'id' query string parameter");

       // … get someObject from database by id …

        return Ok(someObject);
    }
}
```

WebAPI

- There are a lot of similarities between ASP.NET and ASP.NET Core
- Similar to MVC:
    - Controllers: classes that inherit from a special base class
    - Methods: endpoints
    - Optional async/await

ASP.NET WebAPI vs ASP.NET Core WebAPI

- WebAPI/MVC convergence
- CORS
- Configuration
- Startup
- Static files
- Casing / JsonProperty

Core WebAPI Convergence

- In ASP.NET Core, MVC and WebAPI converge
- Instead of Controller and ApiController, there is just Controller
- Instead of IHttpActionResult (WebAPI) there is just IActionResult

CORS

- Cross-Origin Resource Sharing

- ASP.NET
  - `[EnableCors(origins: "*", headers: "*", methods: "*")]`

- ASP.NET Core

  -
    ```
    app.UseCors(builder => builder
                   .AllowAnyHeader()
                   .AllowAnyMethod()
                   .AllowAnyOrigin());
    ```

Startup

- ASP.NET
  - Global.asax.cs
  - `System.Web.HttpApplication`
  - `protected void Application_Start()`

- ASP.NET Core
  - Program.cs – console application
  - Startup.cs – used by Program.cs
  - `public void ConfigureServices(IServiceCollection services)`
  - `public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)`

Configuration

- ASP.NET
  - Web.config (XML)
  - `ConfigurationManager.AppSettings`

- `ASP.NET Core`
  - appsettings.json (JSON)

```
IConfigurationSection settingsSection = Configuration.GetSection("MySettings");
MySettings settings = settingsSection.Get<MySettings>();
services.Configure<MySettings>(settingsSection);
```

Static Files

- ASP.NET
  - Put static files wherever

- ASP.NET Core
  - Put static files in wwwroot folder

Casing / JsonProperty

- ASP.NET
  - `return Ok(result.Value);`
  - Result object serialized
  - Resultant JSON is PascalCased

- ASP.NET Core
  - `return Ok(result.Value);`
  - Result object serialized
  - Resultant JSON is camelCased
  - Use `[JsonProperty("FirstName")]` to PascalCase

Questions

Exercise

- Create REST API
- Checkout the code from Github
  - https://github.com/couchbaselabs/aspnet-nosql-workshop/tree/master/03
- The source code is also available on USB sticks
- You can test with Postman / Fiddler / curl

How to execute

Execute RESTful API backend:

- Visual Studio: F5 (or ctrl+f5)
- `dotnet run` from command line

Exercise: Getting Started

At the end of the lab, your app should be able to <u>list, add, edit, and delete.</u>

If you have questions or are running into a problem, I'll be walking around helping you individually.