ASP.NET 5 WebAPIs

Introduction to REST APIs with .NET 5 and C#

Introduction

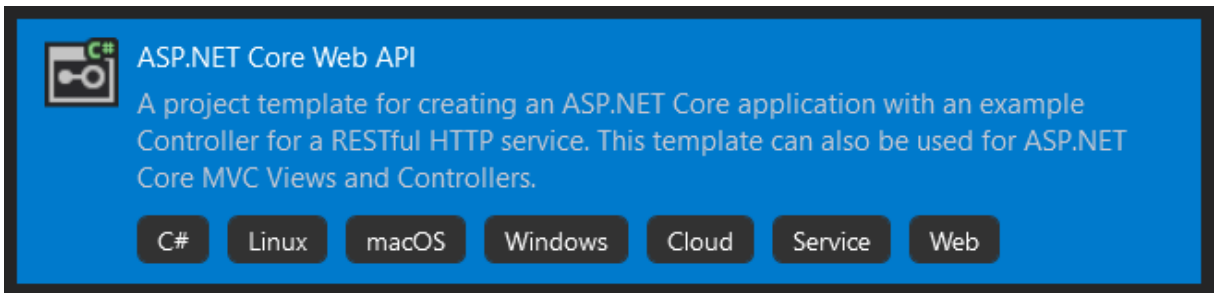
- Develop server-side logic for web apps
  - Serve static content
  - Dynamic content (e.g. HTML, CSS) creation on the server (*ASP.NET MVC*)
  - *HTTP Web APIs* called by other servers or *single page apps* (SPAs)
- Complete, *a la carte* framework
  - Covers all important aspects of web development
  - Can be reduced to specific needs
  - Can be extended with external libraries (NuGet)
- Open source, driven by Microsoft

ASP.NET 5 Fundamentals

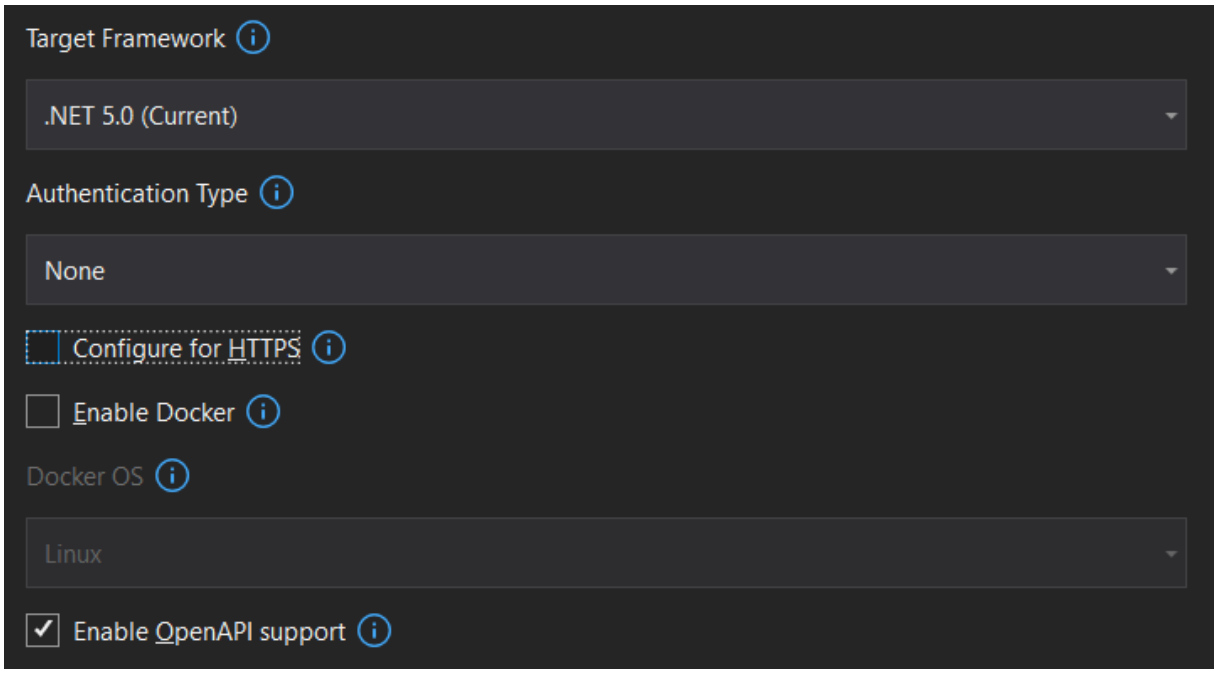
- ASP.NET 5 app is a *console app*
- Embedded web server (*Kestrel*)
  - Comes via NuGet
  - No external web server (e.g. *Tomcat*, *IIS*) necessary
  - *Reverse Proxy* possible for production use (e.g. *API Gateway*)

Create new ASP.NET 5 Web API project

- Create using *ASP.NET Core Web API* template



- Select *.NET 5.0* and *Enable OpenAPI support*



Properties/`launchSettings.json`

- Remove *IIS Express*

```
"profiles": {
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "launchUrl": "swagger",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "ToDoWebApi": {
    "commandName": "Project"
```

Main Program: ***Program.cs***

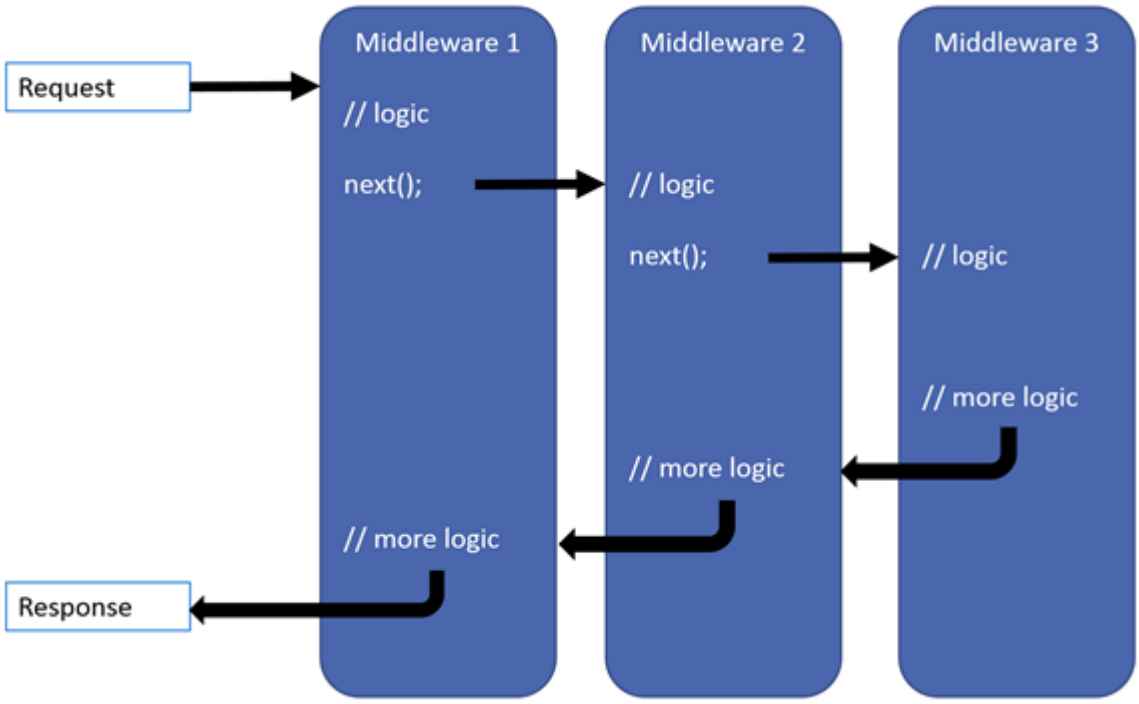
```
namespace ToDoWebApi
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Web APP Pipeline: `Startup.cs`

```
namespace ToDoWebApi
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
        ↪  services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

            services.AddControllers();
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo { Title = "ToDoWebApi",
↪   Version = "v1" });
            });
        }

        // This method gets called by the runtime. Use this method to
        ↪  configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment
        ↪  env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c =>
↪   c.SwaggerEndpoint("/swagger/v1/swagger.json", "ToDoWebApi v1"));
            }

            app.UseRouting();

            app.UseAuthorization();
```

```
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

Web App Pipeline: *Startup.cs*

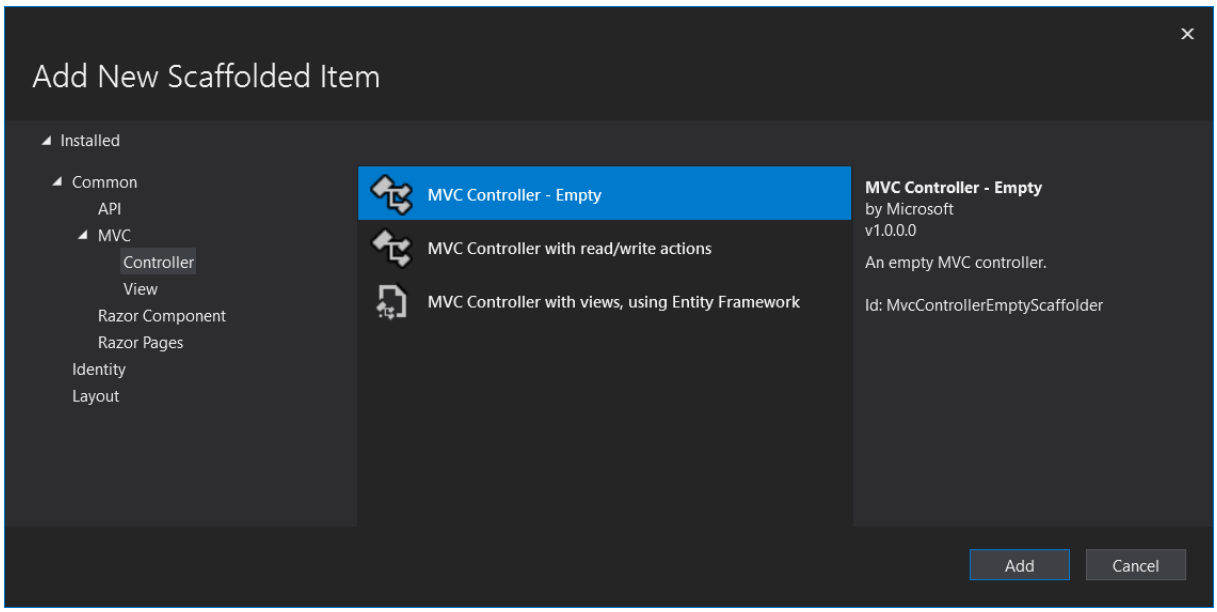


Web App Pipeline: *Startup.cs*

- Create list of *Middlewares* (=*Pipeline*) using functions `Use`, `Map` and `Run`
  - `Use`: Perform some logic and optionally call `next` to invoke next middleware
  - `Map`: Build sub-middleware for specific URL prefix
  - `Run`: Last element in pipeline, no `next`
- **Everything is asynchronous**
- Read more about *ASP.NET Core Startup*...

Add new controller: *ToDoController.cs*

- Click with *right mouse button* on *Controllers* and select *Add... / Controllers*
- Select *MVC Controller - Empty* and name it *ToDoController.cs*

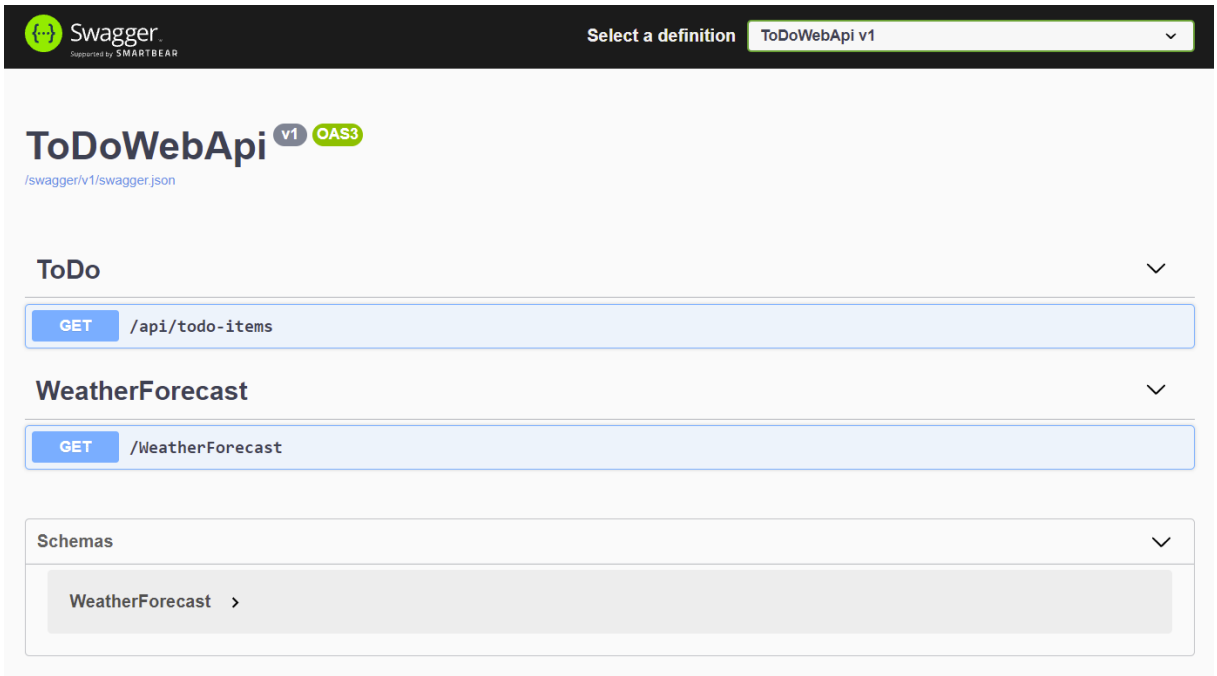


Web API: Simple *Get All*

```
namespace ToDoWebApi.Controllers
{
    [ApiController]
    [Route("api/todo-items")]
    public class ToDoController : Controller
    {
        private static readonly List<string> items =
            new List<string> { "Clean my room", "Feed the cat" };

        [HttpGet]
        public IActionResult GetAllItems()
        {
            return Ok(items);
```

```
        }
    }
}
```

Test endpoint using Swagger



Web API: Simple *Get by ID*

```
[HttpGet]
[Route("{index}", Name = "GetSpecificItem")]
public IActionResult GetItem(int index)
{
    if (index >= 0 && index < items.Count)
    {
        return Ok(items[index]);
    }

    return BadRequest("Invalid index");
}
```

- Get todo item at index 1 with `GET http://localhost:<port>/api/todo-items/1`

Web API: Simple *Add*

```
[HttpPost]
public IActionResult AddItem([FromBody] string newItem)
{
    items.Add(newItem);
    return CreatedAtRoute("GetSpecificItem", new { index =
    ↪  items.IndexOf(newItem) }, newItem);
}
```

Web API: Simple *Update*

```
[HttpPut]
[Route("{index}")]
public IActionResult UpdateItem(int index, [FromBody] string newItem)
{
    if (index >= 0 && index < items.Count)
    {
        items[index] = newItem;
        return Ok();
    }

    return BadRequest("Invalid index");
}
```

Web API: Simple *Delete*

```
[HttpDelete]
[Route("{index}")]
public IActionResult DeleteItem(int index)
{
    if (index >= 0 && index < items.Count)
```

```
    {
        items.RemoveAt(index);
        return NoContent();
    }

    return BadRequest("Invalid index");
}
```

Web API: Accessing Query Parameters

```
[HttpGet]
[Route("sorted")]
public IActionResult GetAllItemsSorted([FromQuery] string sortOrder)
{
    return sortOrder switch
    {
        "desc" => Ok(items.OrderByDescending(item => item)),
        "asc" => Ok(items.OrderBy(item => item)),
        _ => BadRequest("Invalid or missing sortOrder query parameter")
    };
}
```

- Get sorted todo items with `GET http://localhost:<port>/api/todo-items/sorted?sortOrde`

Web API: DTO Classes

```
namespace ToDoWebApi.DTOs
{
    public class ToDoItem
    {
        [MinLength(5)]
        [MaxLength(50)]
        [Required]
        public string Description { get; set; }

        [MaxLength(50)]
        public string AssignedTo { get; set; }
```

```
    }
}
```

- *Data Transfer Objects*
- Note *Data Annotations* like `Required`

Web API: Using DTO Classes

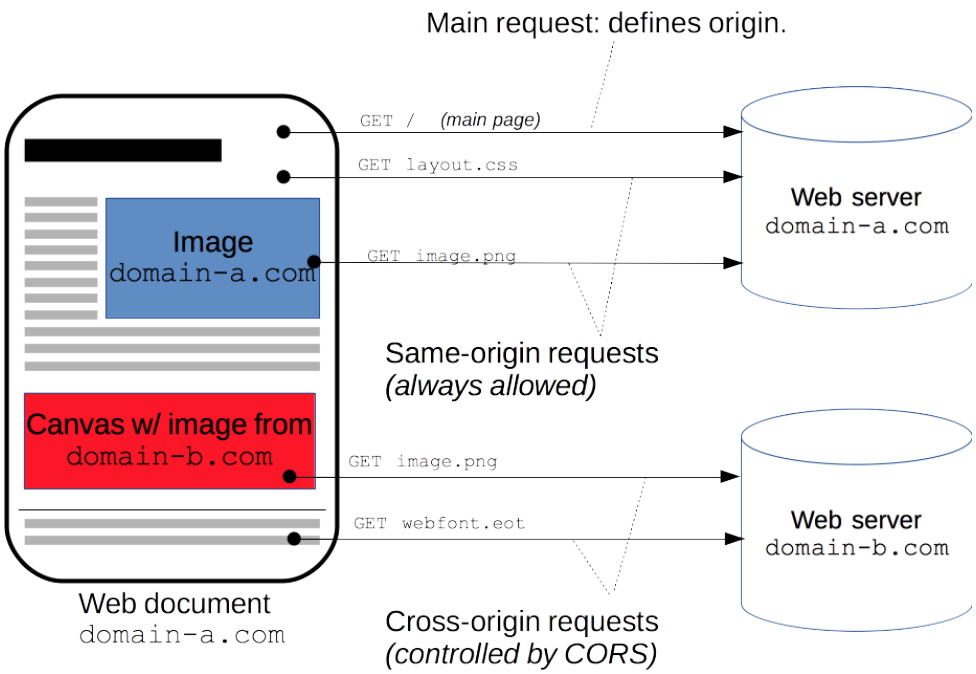
```
namespace ToDoWebApi.Controllers
{
    [ApiController]
    [Route("api/advanced-todo-items")]
    public class AdvancedToDoController : Controller
    {
        private static readonly List<ToDoItem> items =
            new List<ToDoItem> {
                new ToDoItem { Description = "Clean my room", AssignedTo =
                ↪  "me" },
                new ToDoItem { Description = "Feed the cat", AssignedTo =
                ↪  "somebody else" }
            };

        [HttpGet]
        public IActionResult GetAllItems()
        {
            return Ok(items);
        }

        [HttpGet]
        [Route("{index}", Name = "GetSpecificToDoItem")]
        public IActionResult GetItem(int index)
        {
            if (index >= 0 && index < items.Count)
            {
                return Ok(items[index]);
            }

            return BadRequest("Invalid index");
        }
```

```
        [HttpPost]
        public IActionResult AddItem([FromBody] ToDoItem newItem)
        {
            items.Add(newItem);
            return CreatedAtRoute("GetSpecificTodoItem",
                new { index = items.IndexOf(newItem) }, newItem);
        }
    }
}
```

- Try sending invalid data to API
  - ASP.NET Core handles invalid data automatically
  - Error format: RFC 7807

Don't Forget *CORS*!

- Image Source MDN
- CORS in ASP.NET Core...



Enable *CORS* in *Startup.cs*

```
namespace ToDoWebApi
{
    public class Startup
    {
        private readonly string myAllowSpecificOrigins =
        ↪  "_myAllowSpecificOrigins";

        ...

        public void ConfigureServices(IServiceCollection services)
        {
            ...

            services.AddCors(options =>
            {
                options.AddPolicy(myAllowSpecificOrigins,
                    x => x.AllowAnyOrigin()
                          .AllowAnyMethod()
                          .AllowAnyHeader()
                  );
            });
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment
        ↪  env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c =>
↪   c.SwaggerEndpoint("/swagger/v1/swagger.json", "ToDoWebApi v1"));
            }

            app.UseCors(myAllowSpecificOrigins);

            ...
        }
    }
}
```