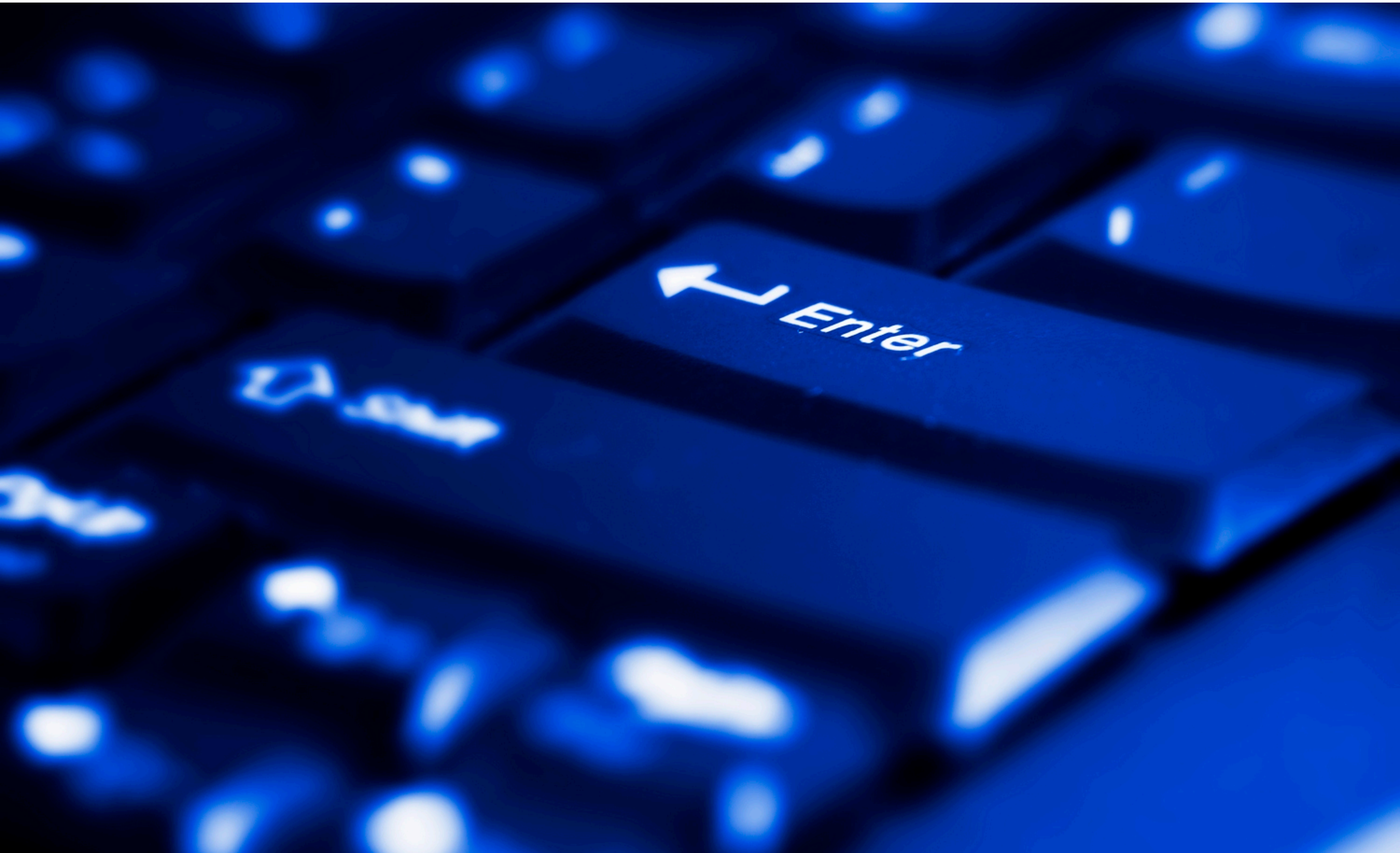
dotNET CORE Web API Fundamentials

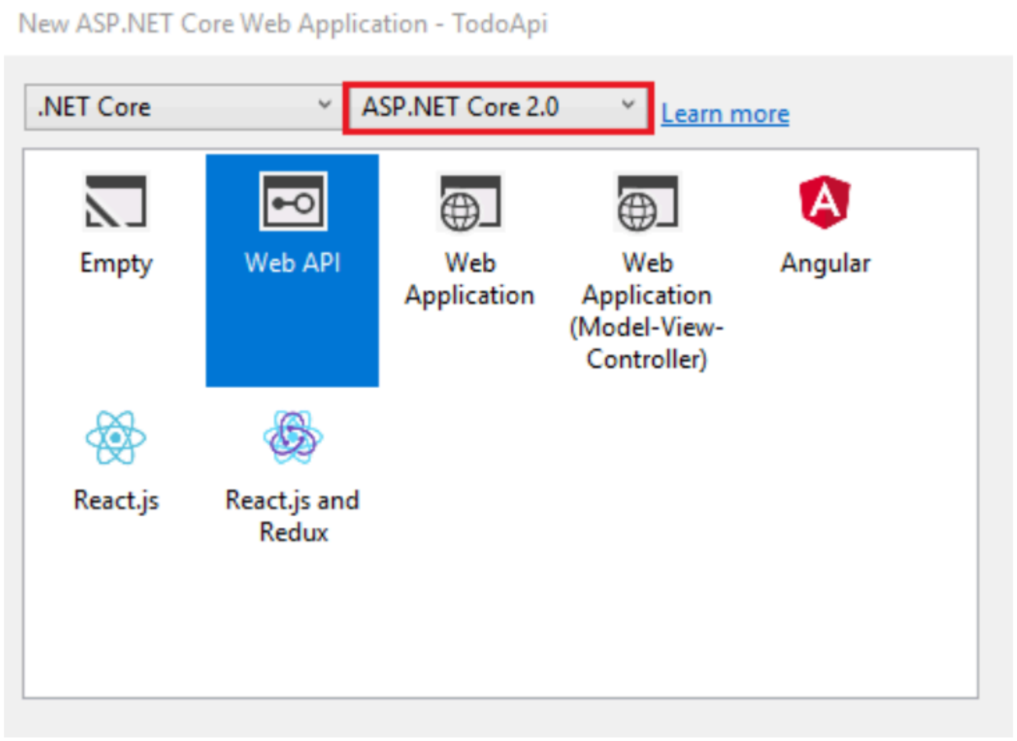
Modern Application Architectures



REST Endpoints

Modern Application Architectures

https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api

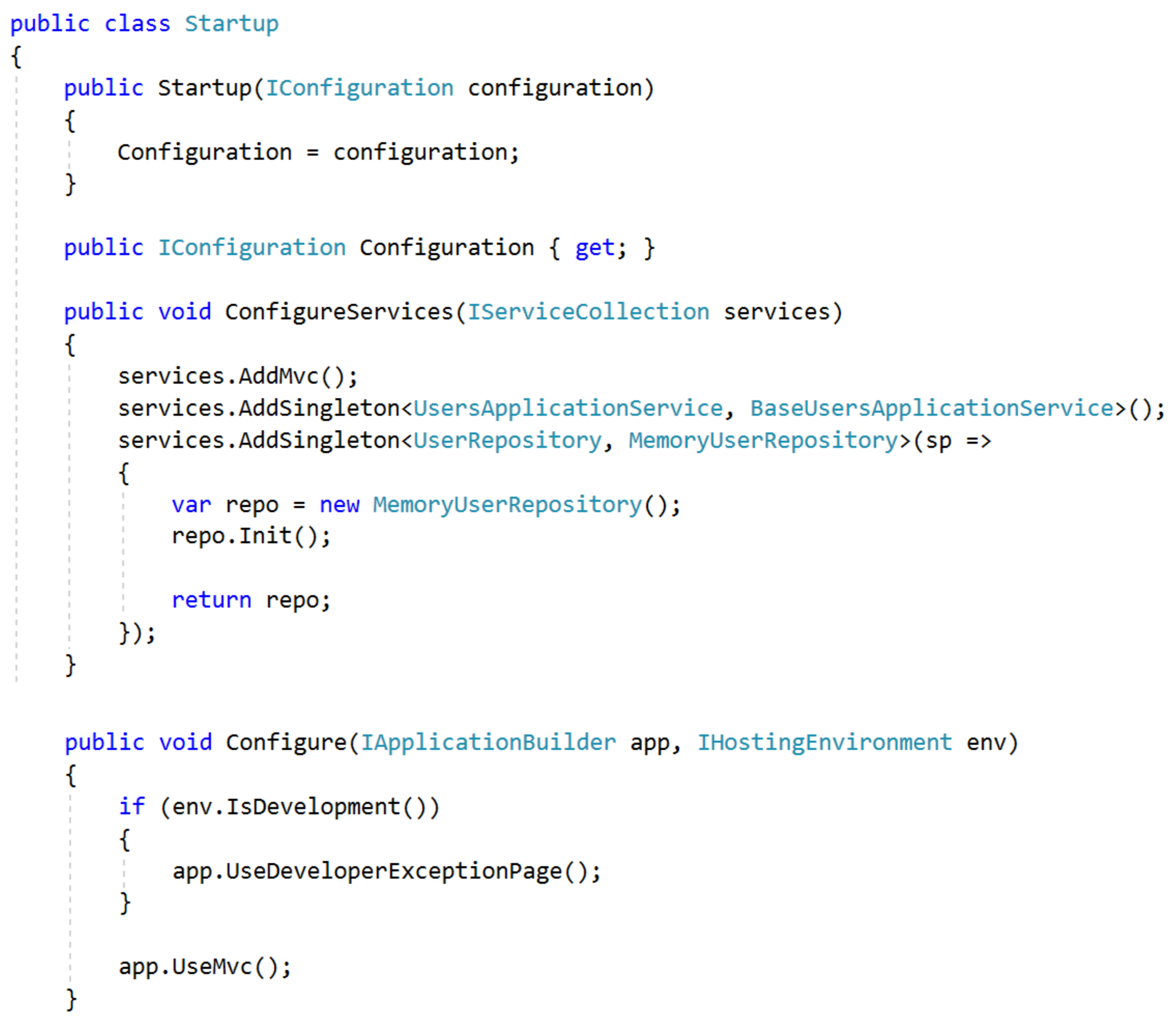

New ASP.NET Core Web Application - TodoApi

.NET Core

ASP.NET Core 2.0

```csharp
public class Program
{
    public static void Main(string[] args)
    {
        BuildWebHost(args).Run();
    }

    public static IWebHost BuildWebHost(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .UseUrls("http://localhost:5000")
                .Build();
}
```



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

    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
        services.AddSingleton<UsersApplicationService, BaseUsersApplicationService>();
        services.AddSingleton<UserRepository, MemoryUserRepository>(sp =>
        {
            var repo = new MemoryUserRepository();
            repo.Init();

            return repo;
        });
    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseMvc();
    }
```

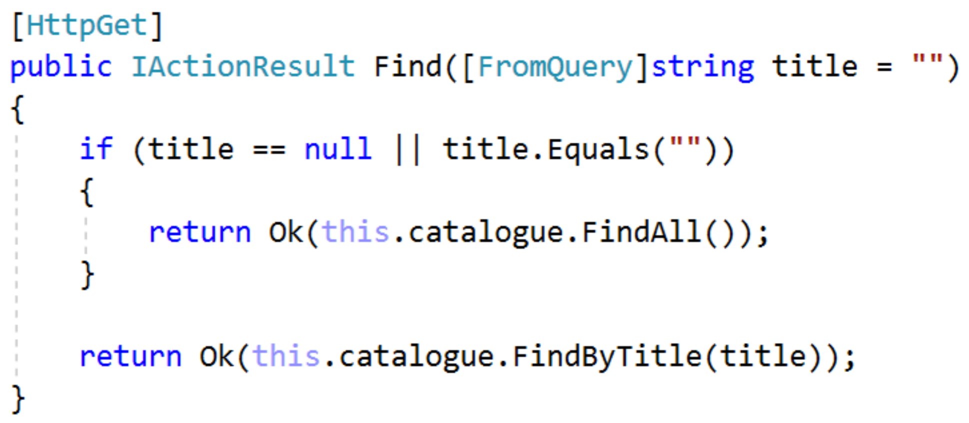
New controller

```
[Route("api/[controller]/books")]
public class CatalogueController : Controller
{
```

# "api/[controller]/books”

- default path for all controller methods
- [controller] will be replaced by name of the controller

Controller Attributes



```
[HttpGet]
public IActionResult Find([FromQuery]string title = "")
{
    if (title == null || title.Equals(""))
    {
        return Ok(this.catalogue.FindAll());
    }

    return Ok(this.catalogue.FindByTitle(title));
}
```

# [HttpGet]
- GET method implementation

# [FromQuery]param
- param will be read from query eg. /books?title=abc

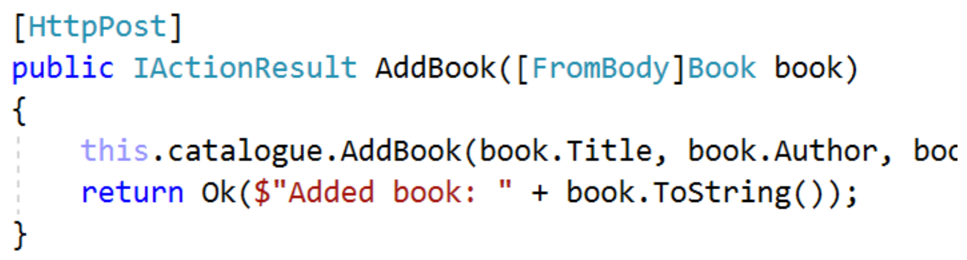
# OK(object)
- return object with OK HTTP status

Controller Attributes

```
[HttpGet]
[Route("{id:long}/description")]
public IActionResult FindByDescription(long id)
{
    try
    {
        return Ok(this.catalogue.FindDescription(id));
    }
    catch (BookNotFoundException e)
    {
        return NotFound(e.Message);
    }
}
```

- [Route(path)]
  - path is appended to the core path of the controller eg. /books/1/description
- {id:long}
  - id will be available as long parameter
- NotFound
  - return result with NOT FOUND HTTP status

Controller Attributes

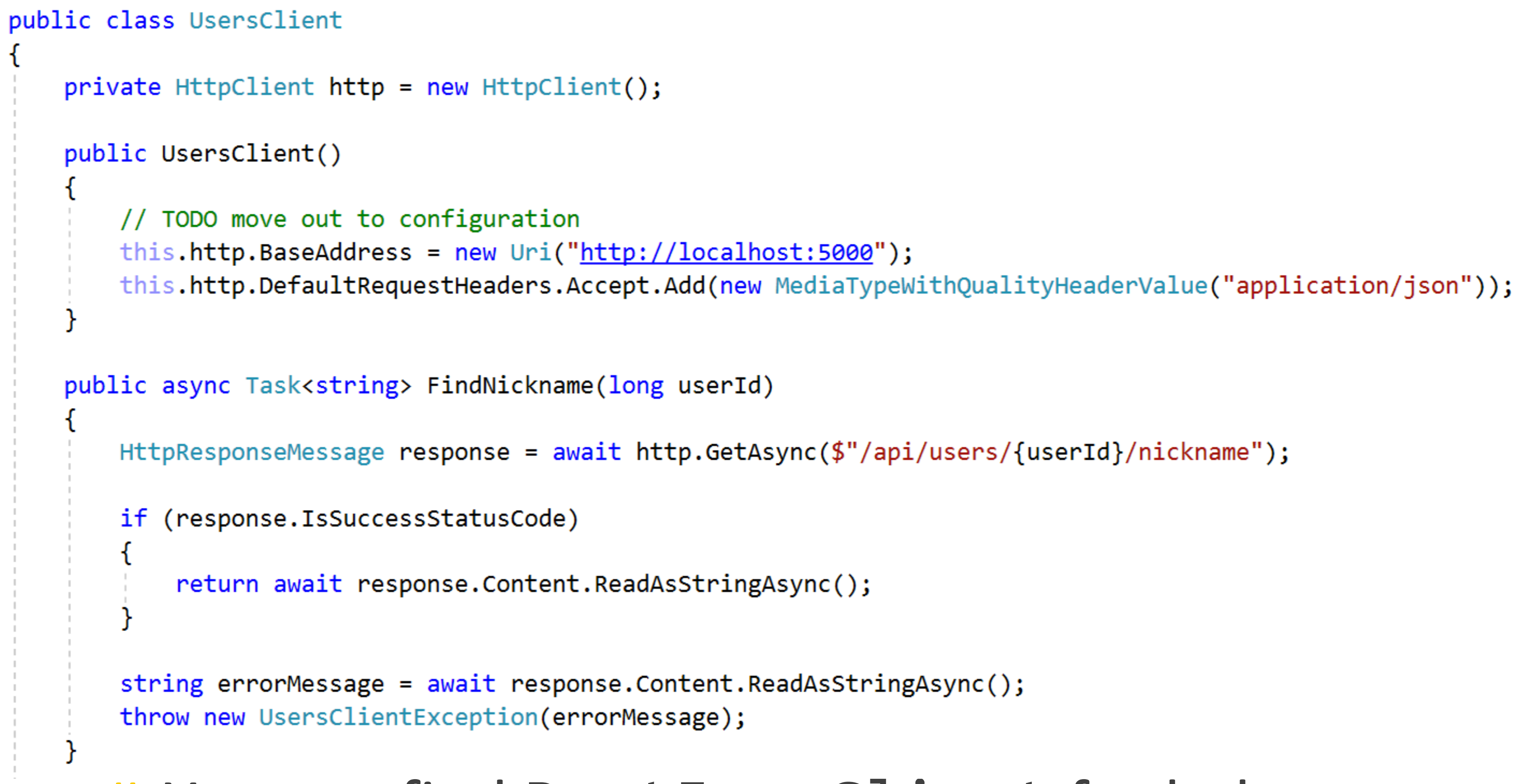


```
[HttpPost]
public IActionResult AddBook([FromBody]Book book)
{
    this.catalogue.AddBook(book.Title, book.Author, boc
    return Ok($"Added book: " + book.ToString());
}
```

- [HttpPost]
  - POST method implementation
- [FromBody]
  - parameter will be created from body contents according to content type
- IActionResult
  - generalized result from Controller method

Client

Modern Application Architectures



```
public class UsersClient
{
    private HttpClient http = new HttpClient();

    public UsersClient()
    {
        // TODO move out to configuration
        this.http.BaseAddress = new Uri("http://localhost:5000");
        this.http.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    }

    public async Task<string> FindNickname(long userId)
    {
        HttpResponseMessage response = await http.GetAsync($"/api/users/{userId}/nickname");

        if (response.IsSuccessStatusCode)
        {
            return await response.Content.ReadAsStringAsync();
        }

        string errorMessage = await response.Content.ReadAsStringAsync();
        throw new UsersClientException(errorMessage);
    }
```

# You can find `RestJsonClient` for help