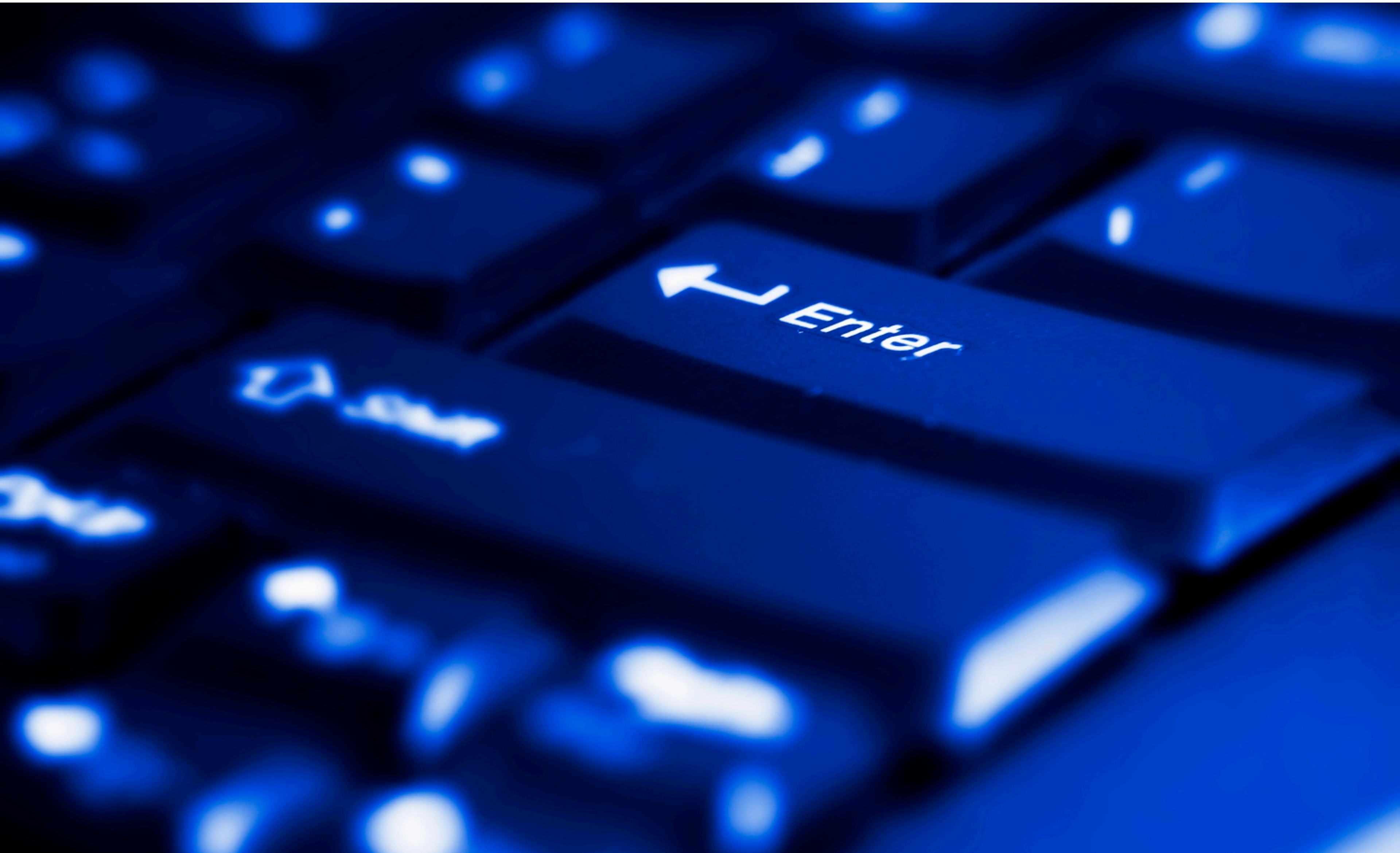
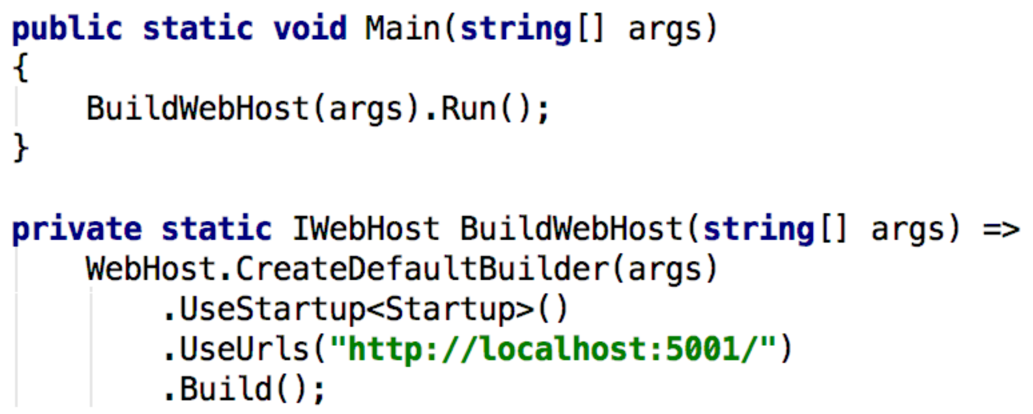
dotNET CORE Web API Fundamentials

Modern Application Architectures



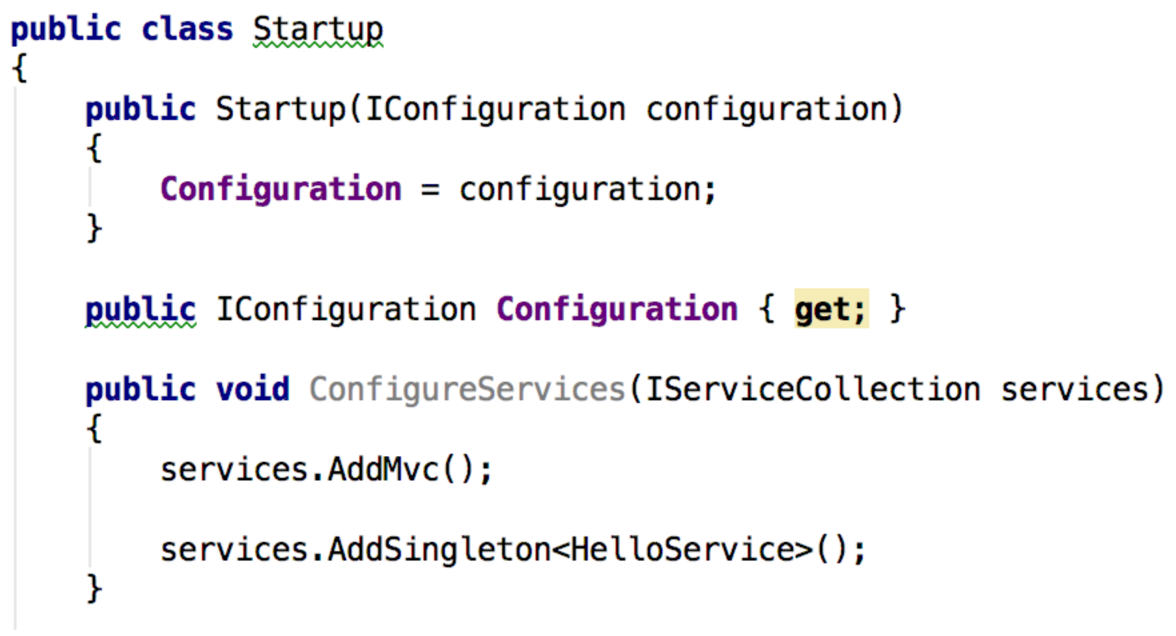
REST Service

Modern Application Architectures



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

private static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>()
        .UseUrls("http://localhost:5001/")
        .Build();
```



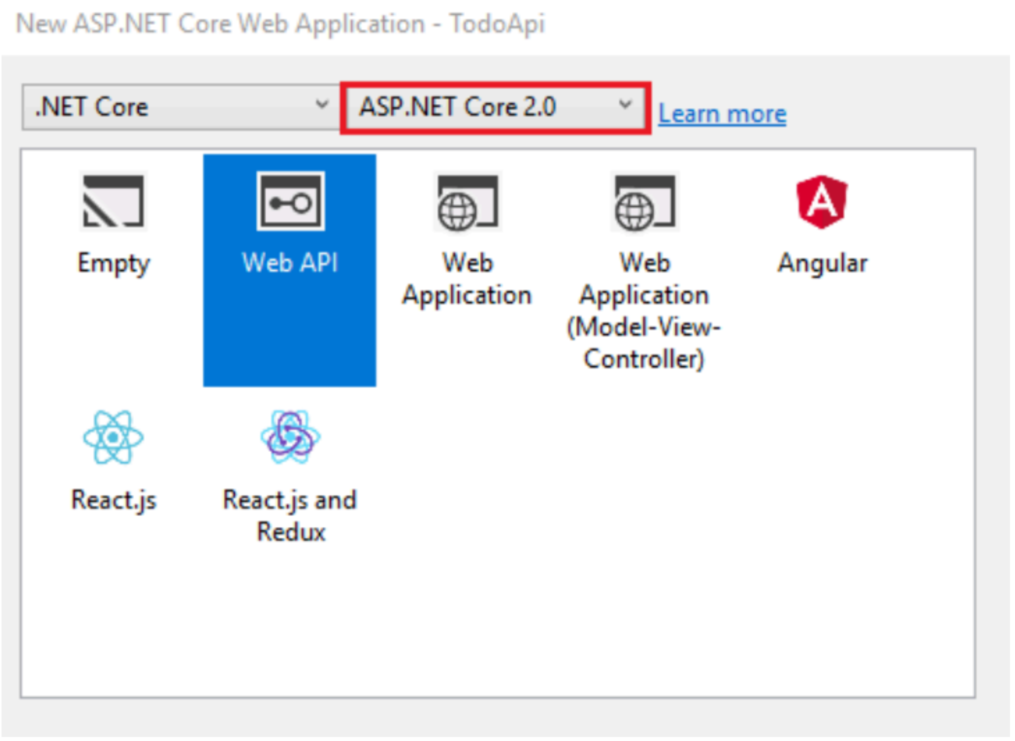
```csharp
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();

        services.AddSingleton<HelloService>();
    }
```

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

```
GET /api/todo
```

```
GET /api/todo/{id}
```

```
public class TodoItem
{
    public long Id { get; set; }
    public string Name { get; set; }
    public bool IsComplete { get; set; }
}
```

```
[Route("api/[controller]")]
public class TodoController : Controller
{
    [HttpGet]
    public IEnumerable<TodoItem> GetAll()
    {
        return _context.TodoItems.ToList();
    }

    [HttpGet("{id}", Name = "GetTodo")]
    public IActionResult GetById(long id)
    {
        var item = _context.TodoItems.FirstOrDefault(t => t.Id == id);
        if (item == null)
        {
            return NotFound();
        }
        return new ObjectResult(item);
    }
```

```
[HttpPost]
public IActionResult Create([FromBody] TodoItem item)
{
    if (item == null)
    {
        return BadRequest();
    }

    _context.TodoItems.Add(item);
    _context.SaveChanges();

    return CreatedAtRoute("GetTodo", new { id = item.Id }, item);
}
```

```
[HttpPut("{id}")]
public IActionResult Update(long id, [FromBody] TodoItem item)
{
    if (item == null || item.Id != id)
    {
        return BadRequest();
    }

    var todo = _context.TodoItems.FirstOrDefault(t => t.Id == id);
    if (todo == null)
    {
        return NotFound();
    }

    todo.IsComplete = item.IsComplete;
    todo.Name = item.Name;

    _context.TodoItems.Update(todo);
    _context.SaveChanges();
    return new NoContentResult();
}
```

```
[HttpDelete("{id}")]
public IActionResult Delete(long id)
{
    var todo = _context.TodoItems.FirstOrDefault(t => t.Id == id);
    if (todo == null)
    {
        return NotFound();
    }

    _context.TodoItems.Remove(todo);
    _context.SaveChanges();
    return new NoContentResult();
}
```

Client

Modern Application Architectures

https://github.com/dotnet/docs/tree/master/samples/csharp/getting-started/console-webapiclient

```
var repositories = ProcessRepositories().Result;

foreach (var repo in repositories)
{
    Console.WriteLine(repo.Name);
    Console.WriteLine(repo.Description);
    Console.WriteLine(repo.GitHubHomeUrl);
    Console.WriteLine(repo.Homepage);
    Console.WriteLine(repo.Watchers);
    Console.WriteLine(repo.LastPush);
    Console.WriteLine();
}

private static async Task<List<Repository>> ProcessRepositories()
{
    var client = new HttpClient();
    var serializer = new DataContractJsonSerializer(typeof(List<Repository>));

    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/vnd.github.v3+json"));
    client.DefaultRequestHeaders.Add("User-Agent", ".NET Foundation Repository Reporter");

    var stringTask = client.GetStringAsync("https://api.github.com/orgs/dotnet/repos");
     var streamTask = client.GetStreamAsync("https://api.github.com/orgs/dotnet/repos");
    var repositories = serializer.ReadObject(await streamTask) as List<Repository>;
    return repositories;
```

```
[DataContract(Name="repo")]
public class Repository
{
    [DataMember(Name="name")]
    public string Name { get; set; }

    [DataMember(Name="description")]
    public string Description { get; set; }

    [DataMember(Name="html_url")]
    public Uri GitHubHomeUrl { get; set; }

    [DataMember(Name="homepage")]
    public Uri Homepage { get; set; }

    [DataMember(Name="watchers")]
    public int Watchers { get; set; }

    [DataMember(Name="pushed_at")]
    private string JsonDate { get; set; }

    [IgnoreDataMember]
    public DateTime LastPush
    {
        get
        {
            return DateTime.ParseExact(JsonDate, "yyyy-MM-ddTHH:mm:ssZ", CultureInfo.InvariantCulture);
        }
    }
}
```

```
<PackageReference Include="System.Runtime.Serialization.Json" Version="4.3.0" />
<PackageReference Include="System.Runtime.Serialization.Primitives" Version="4.3.0" />
```