Microsoft and Apple Training

.NET 2015 and ASP.NET 5

Peter Himschoot
peter@u2u.be

Agenda

- What and Why
- Understanding .NET 2015: .NET 4.6 versus .NET Core 5
- Supporting multiple runtimes
- Frameworks and Runtimes
- The new Roslyn Compiler
  - New language features in C#
- Looking at ASP.NET 5

What and Why

- Microsoft is becoming more
  - Open (Source)
    - Using open source, for example Docker
    - Building open source, for example Core Foundation
      http://www.dotnetfoundation.org/
  - Cross-platform
    - E.g. Office for iOS and OSX
- Next in line is open sourcing the .NET framework
  - CLR, JIT, GC, Base libraries, …
    https://github.com/dotnet/corefx
  - Bringing .NET Core to Linux and Mac OSX

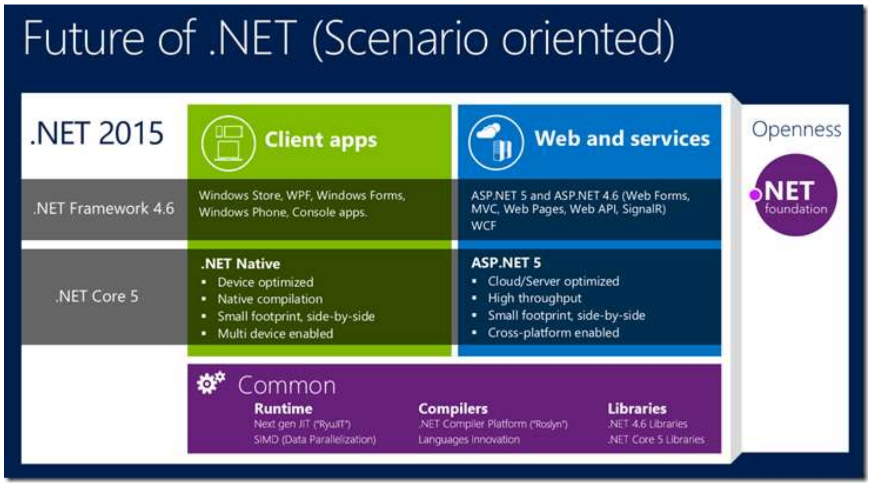
What is .NET Core 5

- **Cross-platform** version of .NET
  - Runs on Windows, Linux and Mac
- Enables **The Internet Of Things** for .NET
  - Runs on Raspberry Pi 2, MinnowBoard Max, Galileo, …
- Smaller, **scenario-specialized**
- Cloud (and server) **optimized**
- Delivered through NuGet packages (modular)

.NET 4.6 versus .NET Core 5

U2U

- .NET 2015 – a collection of .NET releases



Future of .NET (Scenario oriented)

.NET 2015

Client apps

Web and services

Openness

.NET Framework 4.6

Windows Store, WPF, Windows Forms, Windows Phone, Console apps.

ASP.NET 5 and ASP.NET 4.6 (Web Forms, MVC, Web Pages, Web API, SignalR) WCF

.NET foundation

.NET Core 5

.NET Native
- Device optimized
- Native compilation
- Small footprint, side-by-side
- Multi device enabled

ASP.NET 5
- Cloud/Server optimized
- High throughput
- Small footprint, side-by-side
- Cross-platform enabled

Common

Runtime
Next gen JIT ("RyuJIT")
SIMD (Data Parallelization)

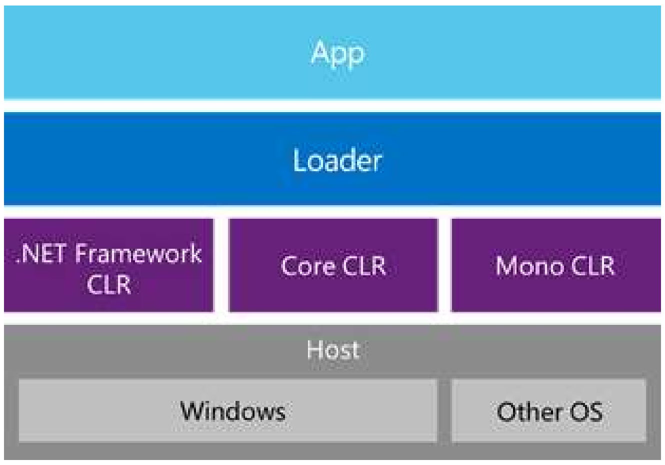
Compilers
.NET Compiler Platform ("Roslyn")
Languages innovation

Libraries
.NET 4.6 Libraries
.NET Core 5 Libraries

Supporting Multiple Runtimes

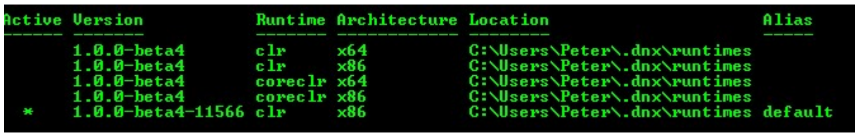
U2U

- .NET Core 5 can run on different runtimes



ASP.NET 5

- ASP.NET 5 can run on .NET 4.6 or .NET Core 5
  - Can run on any version of .NET Core, on the same machine
  - Website A and Website B can run using different versions

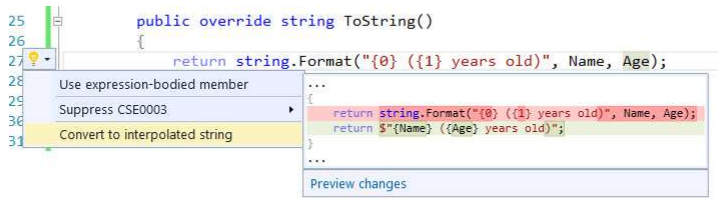


```
Active Version            Runtime Architecture Location                    Alias
------ -------            ------- ------------ --------                    -----
       1.0.0-beta4        clr     x64          C:\Users\Peter\.dnx\runtimes
       1.0.0-beta4        clr     x86          C:\Users\Peter\.dnx\runtimes
       1.0.0-beta4        coreclr x64          C:\Users\Peter\.dnx\runtimes
       1.0.0-beta4        coreclr x86          C:\Users\Peter\.dnx\runtimes
  *    1.0.0-beta4-11566  clr     x86          C:\Users\Peter\.dnx\runtimes default
```

- Running on .NET Core 5 means:
  - Smaller footprint
  - Side-by-side deployment with other versions of .NET Core
  - Develop/Run on Windows, Mac or Linux
- Running on .NET 4.6 means:
  - Highest level of compatibility
  - Windows only

The Roslyn Compiler

- Open Source implementation of the C# compiler
  https://github.com/dotnet/roslyn
- Compiler written in C#
  - Compiles itself ("holy grail")
- Comes with rich code analysis API
  - Compiler becomes platform
    - Intellisense, refactoring, intelligent rename, Go to definition
  - Build your own "light-bulb"



```
        public override string ToString()
        {
            return string.Format("{0} ({1} years old)", Name, Age);
```

```
...
{
    return string.Format("{0} ({1} years old)", Name, Age);
    return $"{Name} ({Age} years old)";
}
...
```

New C# Language Features

- The C# team added the features they wanted
  - But did not have time for until now
- Mainly **syntactic sugar**
  - Make C# more concise…

String Interning

- How many times have you written this kind of code?

```
string.Format("{0} - {1}", Amount, Currency)
```

- This kind of code is very prone to errors…
- String interning to the rescue

```
$"{Amount} ({Currency})"
```

Handy **nameof** operator

- Returns string version of property, method, …
  - Handy for exception handling, INotifyPropertyChanged
  - Using strings is bad for maintenance

```
throw new ArgumentNullException(paramName: "currency");
```

  - Better:

```
throw new ArgumentNullException(paramName: nameof(currency));
```

Readonly Automatic Properties

- Building **immutable** Value Objects (DDD)
  - With readonly properties

```
private readonly decimal amount;

public decimal Amount
{
  get { return amount; }
}
```

- New automatic property syntax

```
public string Currency { get; }
```

Auto property initializers

- Initializing automatic properties
  - Could only be done in constructors
- New auto property initialization syntax
  - Assign value in property declaration – like fields

```
public Money Balance { get; set; } = new Money(0, "EUR");
```

Null-conditional operator

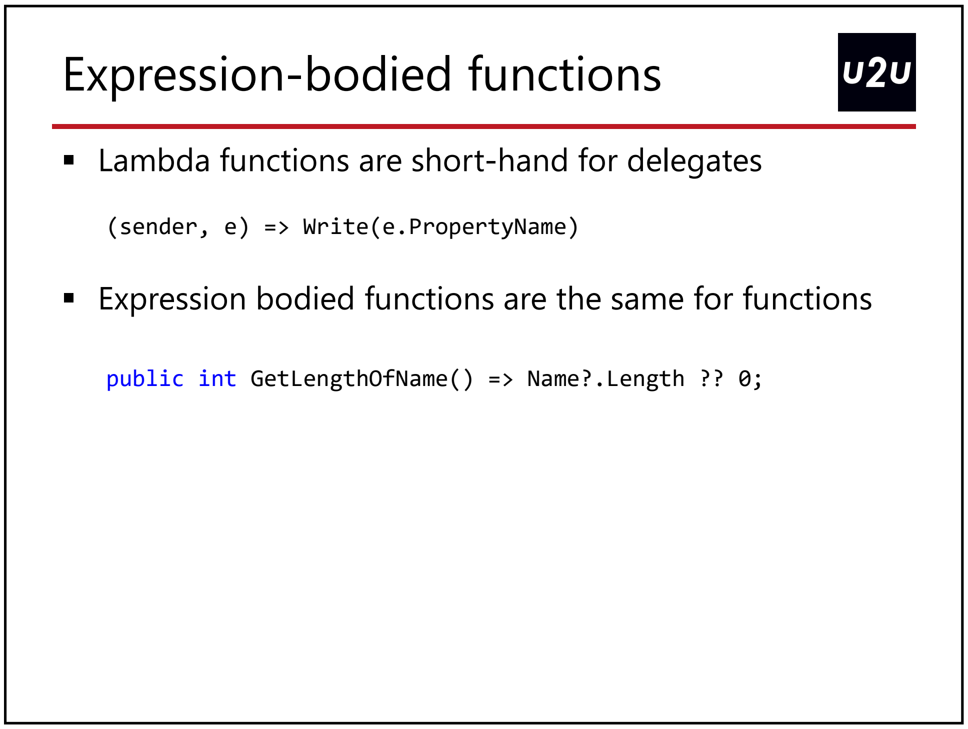
- Handling null references can be very verbose:

```
if (Name != null)           if( PropertyChanged != null )
{                           {
  return Name.Length;         PropertyChanged.Invoke(this, ...
}                           }
else
{
  return 0;
}
```

- Introducing the Null-conditional operator:

```
return Name?.Length ?? 0;


PropertyChanged?.Invoke(this, ...
```



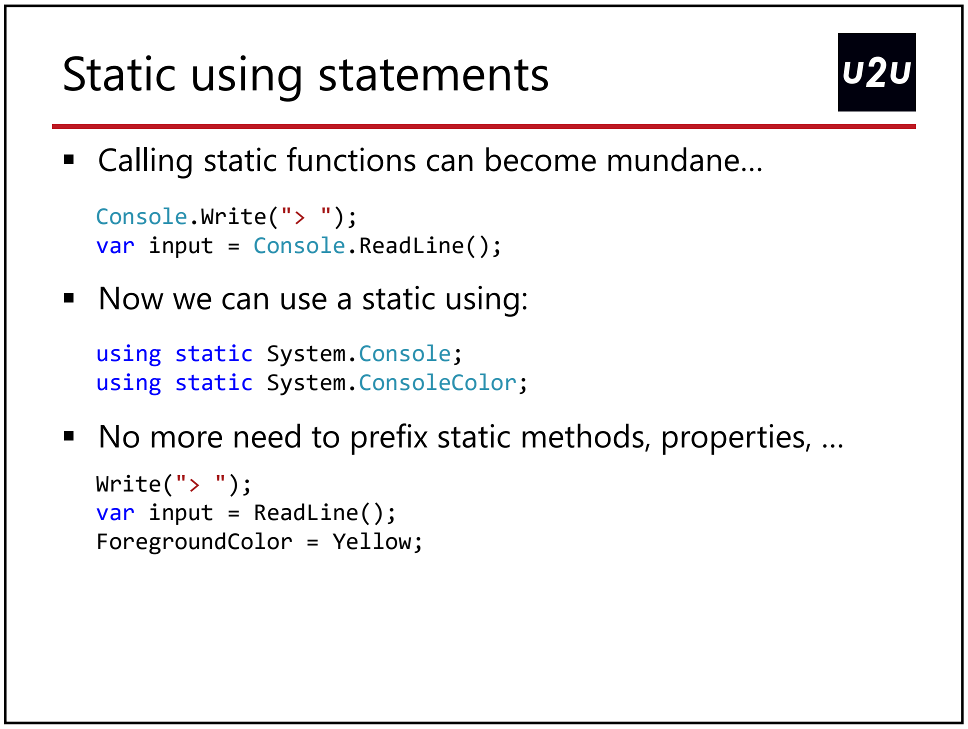
Expression-bodied functions

- Lambda functions are short-hand for delegates

```
(sender, e) => Write(e.PropertyName)
```

- Expression bodied functions are the same for functions

```
public int GetLengthOfName() => Name?.Length ?? 0;
```



Static using statements

- Calling static functions can become mundane…

```
Console.Write("> ");
var input = Console.ReadLine();
```

- Now we can use a static using:

```
using static System.Console;
using static System.ConsoleColor;
```

- No more need to prefix static methods, properties, …

```
Write("> ");
var input = ReadLine();
ForegroundColor = Yellow;
```

Async exception handling

- Using async and await and exception handling is hard
  - E.g. awaiter pattern
- Now C# 6 allows you to use await in the catch/finally

```
public async Task GetMoreInfoAsync()
{
  string s = null;
  try
  {
    HttpClient client = new HttpClient();
    var result = await client.GetAsync("http://www.nobodythere.com");
    s = await result.Content.ReadAsStringAsync();
  }
  catch (ArgumentNullException ex)
  {
    s = await WriteToLog(ex.Message);
  }
}
```

Exception Filters

- To catch exceptions that match some condition
  - Part of VB.NET since the beginning…

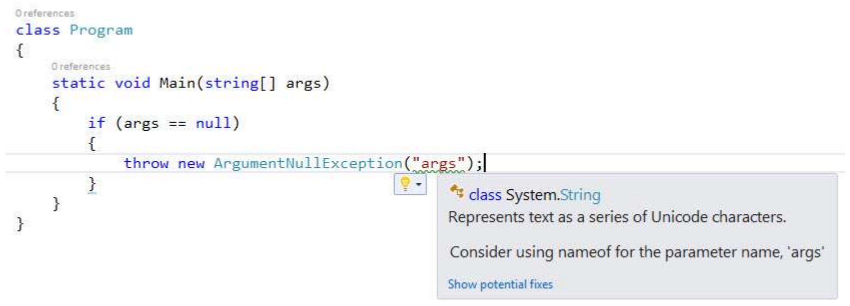
```
catch (ArgumentNullException ex)
  when (ex.ParamName == "requestUri")
```

Starting with C# 6

- Maybe install the Visual Studio extension

C# Essentials Free

- The extension will make suggestions to use C# 6



```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        if (args == null)
        {
            throw new ArgumentNullException("args");
        }
    }
}
```

class System.String
Represents text as a series of Unicode characters.

Consider using nameof for the parameter name, 'args'

Show potential fixes

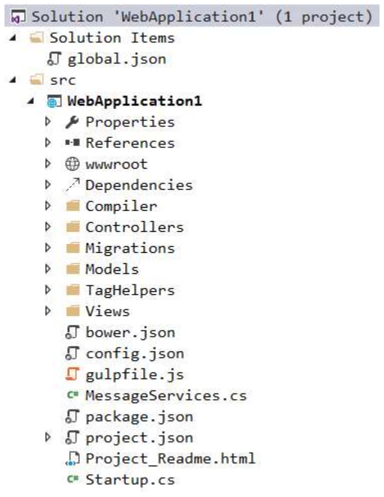
What is ASP.NET 5.0?

- New, from the ground up
  - New light-weight HTTP request pipeline
  - Modular, pay-for-what-you-use
  - Heavily relies on nuget packages
- Open-source, cross-platform
  - GitHub: https://github.com/aspnet/home
  - Windows, Mac, Linux
- Optimized for on premise, or the cloud
  - Seamless transition
  - Unified Web UI and API stack
- Self-host, or host in IIS
- Based on best practices
  - Dependency injection

The new ASP.NET 5 project structure

U2U

- Visual Studio 2015 solution:
  - global.json
    - Contains “sources”
  - project.json
    - Contains target **frameworks**
    - Also has **commands**
    - Tracks **dependencies**
  - wwwroot
    - Contains **static files**
    - Ignored by compiler



```
Solution 'WebApplication1' (1 project)
  Solution Items
    global.json
  src
    WebApplication1
      Properties
      References
      wwwroot
      Dependencies
      Compiler
      Controllers
      Migrations
      Models
      TagHelpers
      Views
      bower.json
      config.json
      gulpfile.js
      MessageServices.cs
      package.json
      project.json
      Project_Readme.html
      Startup.cs
```

Why use JSON files?

U2U

- Easier to merge in source control
- Open for all tooling/editors
  - Editing project.json will update project without VS

Microsoft and Apple Training

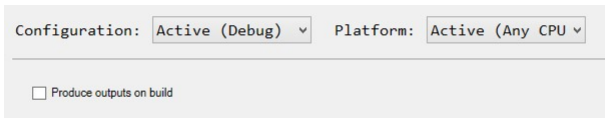
U2U

Demo

Visual Studio Artifacts

U2U

- ASP.NET now compiles to memory
  - Faster
- Does not produce any assemblies on disk
  - You can change this in project properties



Configuration: Active (Debug) Platform: Active (Any CPU

☐ Produce outputs on build

.NET Executing Environment (.DNX)

- Every ASP.NET project is a **DNX** project
  - ASP.NET Application Hosting (package)
- ASP.NET applications are defined in **Startup** class:
  - Replaces global.asax and web.config

```
public class Startup
{
  public Startup(IHostingEnvironment env)
  {}

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

  // Configure is called after ConfigureServices is called.
  public void Configure(IApplicationBuilder app,
    IHostingEnvironment env, ILoggerFactory loggerfactory)
  {}
}
```

The **Startup** method

- Uses Fluent API to configure your web application

```
public Startup(IHostingEnvironment env)
{
  // Setup configuration sources.
  var configuration = new Configuration()
      .AddJsonFile("config.json")
      .AddJsonFile($"config.{env.EnvironmentName}.json", optional: true);

  if (env.IsEnvironment("Development"))
  {
    // This reads the configuration keys from the secret store.
    configuration.AddUserSecrets();
  }
  configuration.AddEnvironmentVariables();
  Configuration = configuration;
}
```

Microsoft and Apple Training

U2U

Demo

Dependency Injection

U2U

- Singleton
  - Created only once, re-used all the time
- Scoped
  - Created if they don't exist yet in current scope
  - Normally a scope is created per request
- Transient
  - Created each time they are requested

?