Setting Up Your C# Pit of Success

with AL Rodriguez

Me (AL)

- @ProgrammerAL
- ProgrammerAL.com
- Senior Azure Cloud Engineer at Microsoft

Why are we here?

- Discuss features of C#/.NET to enforce code quality
    - Setup a Pit of Success
- Present ***Recommendations***
- Please limit your yelling

Pit of Success

"The Pit of Success: in stark contrast to a summit, a peak, or a journey across a desert to find victory through many trials and surprises, we want our customers to simply fall into winning practices by using our platform and frameworks. To the extent that we make it easy to get into trouble we fail."

-Rico Mariani, MS Research MindSwap Oct 2003.

- https://blog.codinghorror.com/falling-into-the-pit-of-success/

Why a Pit of Success?

- Shift-Left
- More work now, less work later

Recommendations

- Some build on each other
- Your call to use or not
- Don't yell at me

Recommendation:

Treat Warnings as Errors

- Don't compile if warning found
  - Ex: Not using method return
- Single Line in `*.csproj` file
- Can disable individual warnings
- Later recommendations built on this

How to Enable Warnings as Error in CsProj

```
<PropertyGroup>
  <TreatWarningsAsErrors>true</TreatWarningsAsErrors>
  <NoWarn>$(NoWarn);NU5104</NoWarn>
<PropertyGroup>
```

Recommendation:

Nullable References Types (NRTs)

- Stop yelling at me!
    - They're good! I swear.
- Not as hard as Rust!
- A lof of code hassle can be mitigated

What are NRTs?

- Nullable Reference Types
- Compiler check for null value at Compile Time
- As Developer, we say code can-be or will-not-be null
- If something can be null when it's used
    - Compiler warning is made

```
<PropertyGroup>
  <Nullable>enable</Nullable>
<PropertyGroup>
```

Sample NRT Usage

```
public void SomeMethod()
{
  var userId = LoadUserId(_context);
  Console.WriteLine(userId);//Generates Compiler Warning
}

private string? LoadUserId(HttpContext context)
{
  ...
}
```

Mitigating Extra Code for NRTs

- Attributes
  - `[NotNullWhen]`, `[MemberNotNullWhen]`, etc
  - Full List: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/attributes/nullable-analysis
- `required` keyword on properties can reduce attributes

Demo: NRT Attribute Example

```
public void SomeMethod()
{
  //No Compiler Warning because if() check on the variable
  if(TryLoadUserId(_context, out string? userId))
  {
    Console.WriteLine(userId);
  }
}

public bool TryLoadUserId(HttpContext context, [NotNullWhen(true)]out string? userId)
{
  userId = "abc";
  return true;
}
```

Demo: DataAnnotations POCO Keyword Example

```
public class MyWebRequest
{
  [Required(AllowEmptyStrings = false)]
  public string? Name { get; set; }
}

public void SomeMethod(MyWebRequest myReq)
{
  Console.WriteLine(myReq!.Name);
}
```

Demo: Required DataAnnotations POCO Keyword Example

```
public class MyWebRequest
{
  [Required(AllowEmptyStrings = false)]
  public required string Name { get; set; }
}

public void SomeMethod(MyWebRequest myReq)
{
  Console.WriteLine(myReq.Name);
}
```

Check for Null at "Ingress of the App"

- Add to entities coming into your code
  - HTTP Requests
  - External Service Calls
  - Database Calls*
- Require data should be default
  - Not always possible, more thinking now (shift-left)

Nullable References Types Finds Bugs

- See Title

Recommendation:

Static Code Analysis

- Checks for common code issues
  - Sometimes fixes
- Enforce styling rules

External Tools

- SonarSource / SonarQube / SonarCloud
- CodeMaid
- FxCop
- Rider / Resharper
- .NET

Roslyn Analyzers

- Scan code at compile time
  - Built-in ones
  - Add with NuGets
  - Build your own
- May include code fixes

Built-In Roslyn Analyzers

```xml
<PropertyGroup>
  <EnableNETAnalyzers>true</EnableNETAnalyzers>
  <IncludeOpenAPIAnalyzers>true</IncludeOpenAPIAnalyzers>
  <EnforceCodeStyleInBuild>true</EnforceCodeStyleInBuild>
  <EnableRequestDelegateGenerator>true</EnableRequestDelegateGenerator>
  <EnableConfigurationBindingGenerator>true</EnableConfigurationBindingGenerator>
</PropertyGroup>
```

- Note: `Microsoft.CodeAnalysis.NetAnalyzers` Package or `<EnableNETAnalyzers>` replaced FxCop

Add Analyzers with NuGet Packages

- `*.Analyzers` package
  - `XUnit.Analyzers`
  - `Roslynator.Analyzers`
  - `Microsoft.Azure.Functions.Worker.Sdk.Analyzers`
  - `SonarAnalyzer.CSharp`

`.editorconfig` File

- Extensible/ Open Standard / Configurable / etc
- Single file checked into source control
- Different languages have different levels of support for it
  - C# support is really good
- https://editorconfig.org

```
# C# files
[*.cs]

#### Core EditorConfig Options ####

# Indentation and spacing
indent_size = 4
indent_style = space
tab_width = 4

# New line preferences
end_of_line = crlf
insert_final_newline = true

# Language keywords vs BCL types preferences
dotnet_style_predefined_type_for_locals_parameters_members = true:error
dotnet_style_predefined_type_for_member_access = true:error

# Parentheses preferences
dotnet_style_parentheses_in_arithmetic_binary_operators = always_for_clarity:error
dotnet_style_parentheses_in_other_binary_operators = always_for_clarity:error
dotnet_style_parentheses_in_other_operators = never_if_unnecessary:silent
#dotnet_style_parentheses_in_relational_binary_operators = always_for_clarity:error

# Built-In Error Codes
csharp_style_deconstructed_variable_declaration = true:suggestion
dotnet_style_coalesce_expression = true:suggestion

# Specific Error Codes
dotnet_diagnostic.S3267.severity = suggestion
csharp_style_namespace_declarations=file_scoped:error
dotnet_diagnostic.S3903.severity=error
dotnet_diagnostic.S4041.severity=error
dotnet_diagnostic.SA1403.severity=error
```

Recommendation:

Codify Code Patterns

- Don't rely on humans to always do things right
- PRs only go so far

Custom Roslyn Analyzers

- Make your own
  - Specific to your projects
- Example:
  - All ASP.NET Controller Endpoints must include `[Authorize]` or `[AllowAnonymous]` attribute

C# Code Generators

- "Next level Roslyn"
- Already used by the .NET Team heavily for AoT stuff

Common Interface Example

```csharp
public interface IUserManager
{
  ValueTask UpdateNameAsync(string name);
}

public class UserManager : IUserManager
{
  public async ValueTask UpdateNameAsync(string name)
  {
    ...
  }
}

//Register with IoC Container
builder.Services.AddScoped<IUserManager, UserManager>();
```

Generated Interface Example

```
[GenerateSimpleInterface]
public class UserManager : IUserManager
{
  public async ValueTask UpdateNameAsync(string name)
  {
    ...
  }
}

//Register with IoC Container
builder.Services.AddScoped<IUserManager, UserManager>();
```

Recommendation:

Secure your Software Supply Chain

- Update Dependencies
  - Update nugets each sprint

Protect Against NuGet CVEs

```xml
<PropertyGroup>
  <NuGetAuditMode>all</NuGetAuditMode>
  <NuGetAuditLevel>low</NuGetAuditLevel>
</PropertyGroup>
```

- Adds build warnings
  - https://learn.microsoft.com/en-us/nuget/concepts/auditing-packages

Recommendation:

Use .NET Aspire

- The New Hotness
- Handy for local feedback
- Also tries to setup a Pit of Success for devs
    - Set config with service discovery instead of hard coding
    - Good defaults for Observability (OpenTelemetry!)

Recommendation:

Continuous Testing

- Continuous Testing Tools
  - NCrunch
  - Resharper*/Rider*
  - Visual Studio*
  - `dotnet watch`

* = certain licenses of that product

Non-Recommendation:

Ahead of Time (AoT) Compilation

- Should probably skip it for your apps
- Consider it for NuGets

Review

- Warnings as Errors
- Nullable Reference Types
- Static Code Analysis
- Codify Patterns
- Secure Supply Chain
- Continuous Testing

Online Info

- @ProgrammerAL
- programmerAL.com