C# Extension Methods

Module 3: Advanced Extension Methods (part 2)

Elton Stoneman
geekswithblogs.net/eltonstoneman
elton@sixeyed.com

pluralsight
hardcore developer training

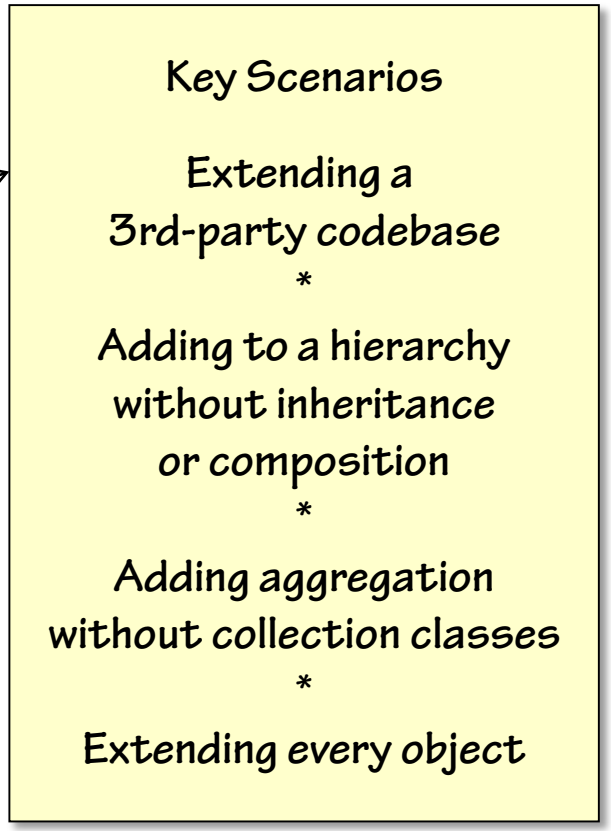
Advanced Extension Methods

- What you can't do
  - Extending state
  - Bypass accessibility
- How extension methods work
  - In the compiler
  - At runtime
- What that means
  - Limitations
  - Portability
- **Resolving extension methods**
  - Compile-time rules
  - Swapping implementations

You are here

What that Means

- **Syntactic sugar**
  - Built into the language
  - Limitations - accessibility and state
    - Original class unchanged
    - External method call
- **At design time**
  - Enabler for key scenarios
- **At run time**
  - Fast, efficient IL call
- **At build time**
  - Portable code
  - Compiler determines call



Demo 4: Portable Extension Methods

Feature

Create a shortcut for string.Format which can be used in any .NET app

Task

Extend string class in a portable library

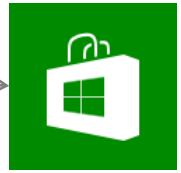
Task

Extend string class in a client-profile library

Demo 4: Portable Extension Methods

Demo 4: Portable Extension Methods

- Class Library **extension methods**
  - Can be used anywhere with a valid reference
    - C# projects, Visual Basic projects, etc.
    - Not portable (Windows Store) projects
- Portable Class Library **extension methods**
  - Can be used anywhere with a valid reference
    - Portable and non-portable C# projects, Visual Basic projects, etc.
- Targeted Class Library **extension methods**
  - Can be used with the same (or higher) .NET framework
    - Except Portable projects
  - Including Client Profile targets
  - Minimum framework version 3.5



Compiler Binding

- **Generates IL for static call**
  - At build time
- **Disambiguation rules**
  - Extension method name == method name
    - Class method called
  - Extension methods for different types
    - Most specific called
  - Multiple extension methods in scope
    - Compiler error
  - No extension methods in scope
    - Compiler error

Demo 5: Compiler Binding

Feature

Return a rounded string from any decimal input

Task

Extend Decimal class with new ToString method

Demo 5: Compiler Binding

Demo 5: Compiler Binding

- **Overriding class methods**
  - Extension method with same signature
  - Compiler ignores extension method
- **Extension methods for matching types**
  - Extension method same name
  - For different types

```
public static string ToStringRounded(this decimal input)
```

```
public static string ToStringRounded(this IComparable input)
```

  - Binds to exact match

```
10.51M.ToStringRounded();
```

```
((IComparable)10.51M).ToStringRounded();
```

Demo 5: Compiler Binding

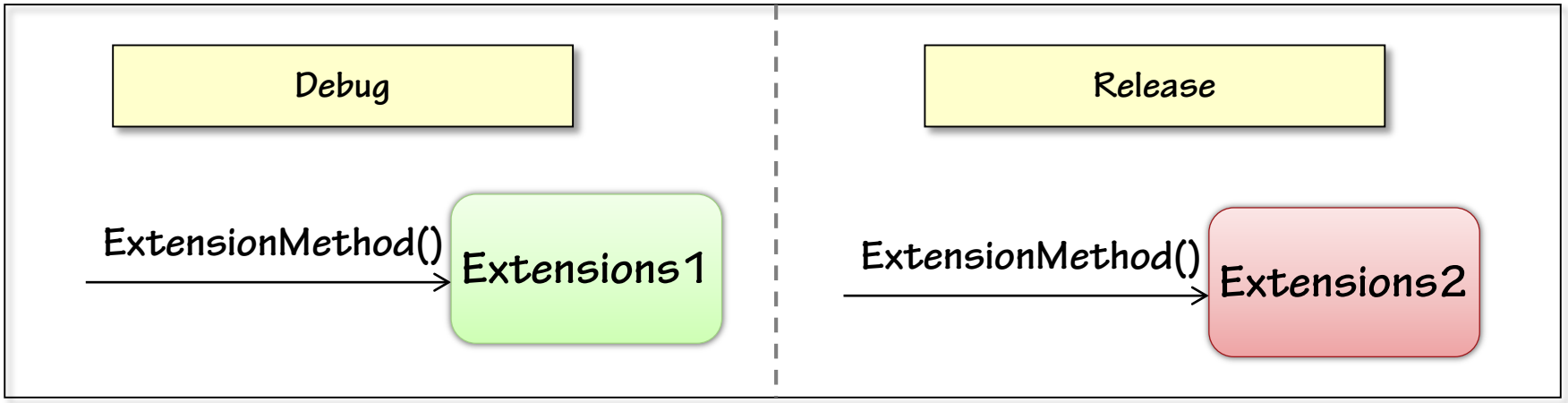
- **Conflicting extension methods**
  - Same signature, same namespace

```
namespace System
{
  public static class DecimalExtensions
  {
     public static string ToStringRounded(this decimal input)
  //etc.
  public static class DecimalExtensions2
  {
     public static string ToStringRounded(this decimal input)
```

  - Build error - remove one from scope
  - Be careful piggybacking namespaces
- **No extension methods**
  - Build error

Configuration Binding

- **Call decided at build time**
  - Governed by build process
  - Can change between configurations
- **Call different extension methods**
  - In Debug and Release configurations
  - Exactly the same codebase



Demo 6: Configuration Binding

Feature

Add reCAPTCHA to website without impacting dev time

Task

Add reCAPTCHA to page with Html Helper extension

Task

Configure build to use stub Html Helper extension in Debug config

Demo 6: Configuration Binding

Demo 6: Configuration Binding

- **Switching between implementations**
  - Use extension method as usual

```
@Html.Raw(Html.GenerateCaptcha("reCAPTCHA", "blackglass"))
```

  - Define alternative - same namespace, same signature

```
public static string GenerateCaptcha(this HtmlHelper helper,
                                     string id, string theme)
```

  - Reference alternative assembly
  - Edit .csproj to condition between implementations

```
Condition="'$(Configuration )' == 'Debug'"
```

  - Works with any project type
    - Visual Studio copies all included assemblies
  - Or use a separate namespace
    - Config transforms & publish profiles

Module Summary

- **What you can't/shouldn't do** ✓
  - Extending state – store additional state
  - Bypass accessibility – except with reflection
- **How extension methods work** ✓
  - Compiler binding to IL
- **What that means** ✓
  - Limitations on state and accessibility
  - Portability
    - .NET languages
    - Portable class libraries
    - Any .NET framework, 3.5+
- **Resolving extension methods** ✓
  - Compiler binding rules
  - Conditional configuration binding