

C# Extension Methods

Module 2: Advanced Extension Methods (part 1)

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



pluralsight 
hardcore developer training

Advanced Extension Methods

- Looking closer at Extension Methods

The image illustrates the difference between a debug build and a release build of an application, specifically focusing on how extension methods are handled.

Debug build: The left side shows a Visual Studio window displaying the IL (Intermediate Language) code for a method named `ToXmlDateTime`. The code is annotated with various attributes and instructions, including `.custom instance void [Microsoft.VisualStudio.TestTools.UnitTesting.Framework]Microsoft.VisualStudio.TestTools.UnitTesting.Framework::TestClassAttribute` and `.locals init ([0] string xmlDateTime, [1] string xmlDateTime2)`. The instructions include `IL_0000: nop`, `IL_0001: ldc.i4 0x7dd`, `IL_0006: ldc.i4.s 10`, `IL_0008: ldc.i4.s 24`, `IL_000a: newobj instance void`, `IL_000f: call string [Sixeyed.Extensions]Sixeyed.Extensions::ToXmlDateTime(string)`, `IL_0014: stloc.0`, `IL_0015: ldstr "2013-10-24T00:00:00"`, `IL_001a: ldloc.0`, `IL_001b: call void [Microsoft.VisualStudio.TestTools.UnitTesting.Framework]Microsoft.VisualStudio.TestTools.UnitTesting.Framework::Assert.AreEqual(string, string)`, `IL_0020: nop`, `IL_0021: ldc.i4 0x7dd`, `IL_0026: ldc.i4.s 10`, `IL_0028: ldc.i4.s 24`, `IL_002a: newobj instance void`, and `IL_002f: call string [Sixeyed.Extensions]Sixeyed.Extensions::ToXmlDateTime(string)`. A green box labeled "Debug build" points to this window.

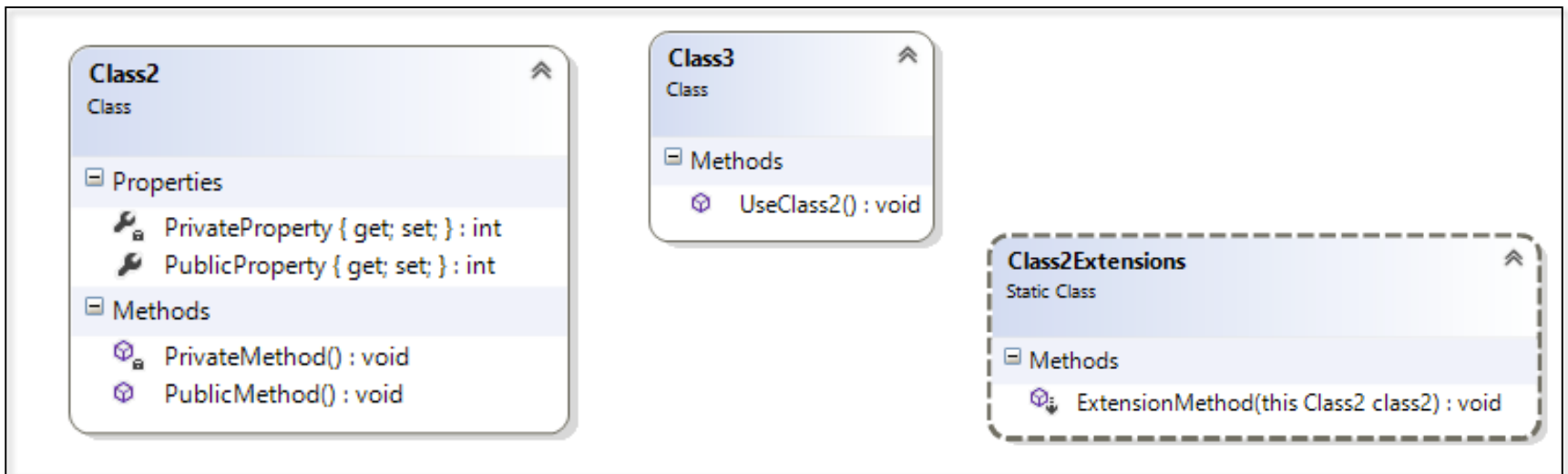
Release build: The right side shows a screenshot of the application's user interface. It features a form titled "Submit a form" with a text input field labeled "Field 1:". Below the input field is a "Prove you're human" section with a reCAPTCHA widget. The reCAPTCHA widget displays a distorted image of the number "177" and the number "5888458". Below the reCAPTCHA widget is a "Type the text" input field and a "Register" button. A green box labeled "Release build" points to this screenshot.

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
-
- ```
graph LR; A[accessibility] --> B[public, internal, private]; A --> C[abstract, sealed]; D[Portability] --> E[Other languages
Other frameworks];
```
- public, internal, private
- abstract, sealed
- Other languages  
Other frameworks

# What you can't do

- Access inaccessible members
  - Using the extended class externally



- Extend state
  - Static method, static class
  - Can add to class behaviour - but not state

# Demo 1: Extending State

## Feature

Extend system instrumentation to increase precision

## Task

Extend the Instrumentation class to use a Stopwatch

# **Demo 1: Extending State**

# Demo 1: Extending State

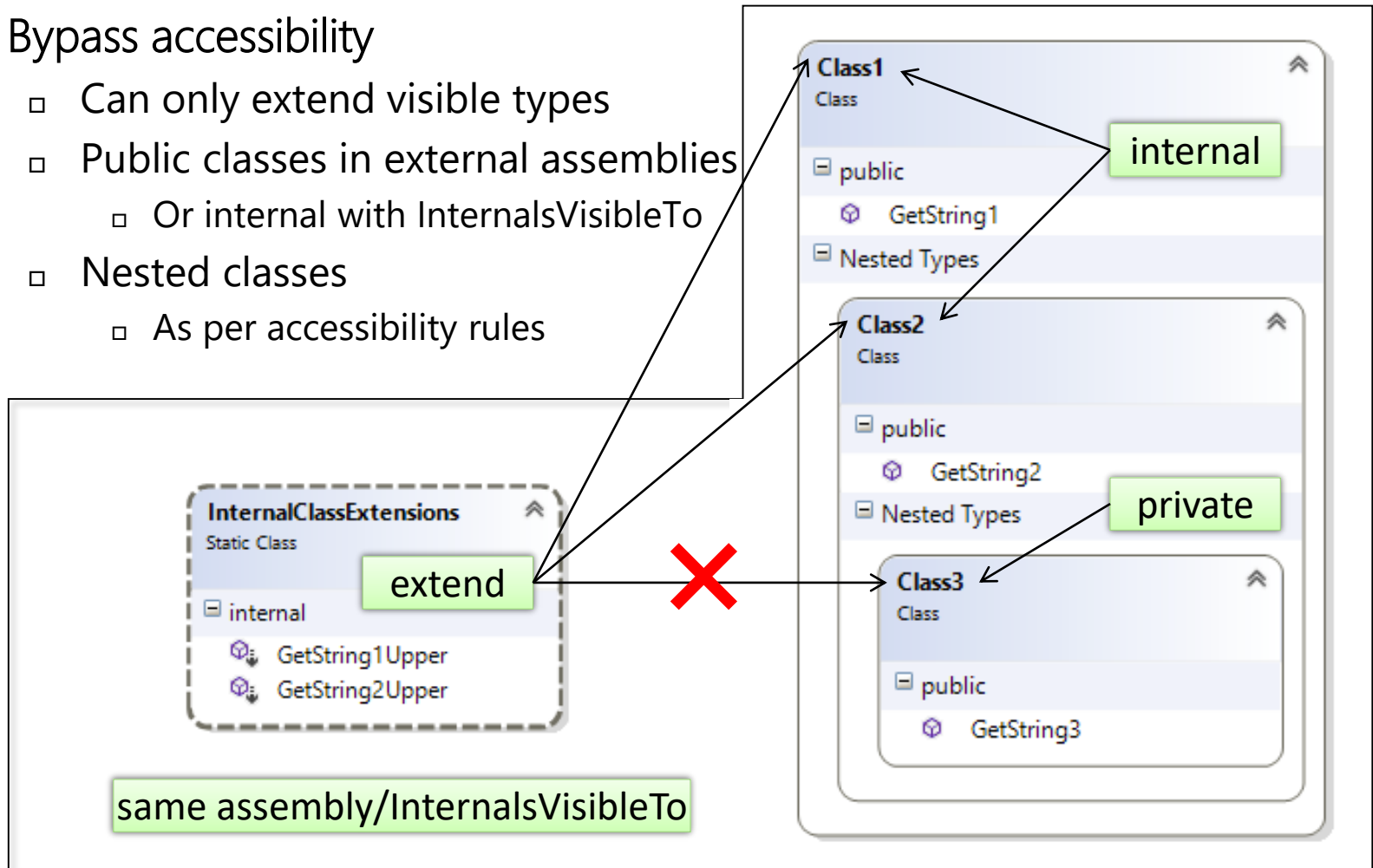
- Can access visible members
  - Public/internal methods, properties & fields
- Can't access others
  - Private/protected
  - But can use reflection
- Can't add state
  - But you can keep additional state
  - Dictionary in extension class, keyed by ID

```
private static Dictionary<Guid, Stopwatch> _Stopwatches;
```

- Not recommended
  - Reflection breaks if type changes
  - No GC for additional state

# What you can't do

- Bypass accessibility
  - Can only extend visible types
  - Public classes in external assemblies
    - Or internal with InternalsVisibleTo
  - Nested classes
    - As per accessibility rules





# Demo 2: Bypassing Accessibility

## Feature

Extend  
inaccessible  
classes

## Task

Try to extend  
inaccessible  
classes...

## **Demo 2: Bypassing Accessibility**

# Demo 2: Bypassing Accessibility

- Cannot bypass accessibility
  - Extend public classes from any assembly
  - Extend internal classes in same assembly
    - Or any assembly named with InternalsVisibleTo
  - Extend nested classes – if accessible
- Can extend inheritance
  - Extend sealed classes – if accessible
  - Extend abstract classes – if accessible

# Demo 2: Bypassing Accessibility

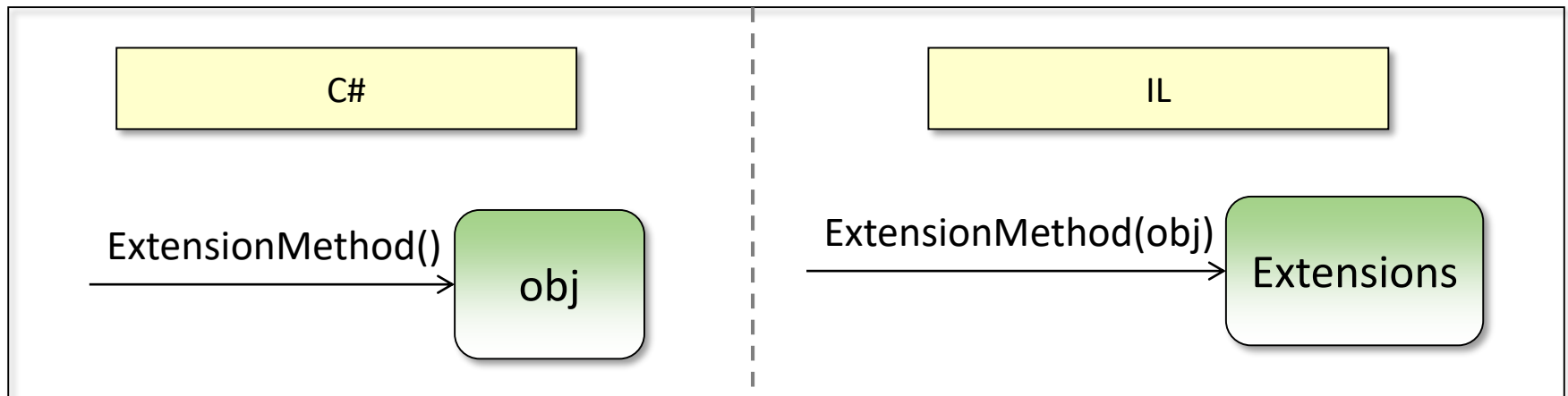
- Cannot extend invisible classes
  - Private/internal
  - Unless you use reflection

```
internal static string GetString3Upper(this object obj)
{
 var type3 = typeof(Class1.Class2)
 .GetNestedType("Class3", BindingFlags.NonPublic);
 if (obj.GetType() == type3) //etc.
```

- Not recommended
  - Reflection breaks if type changes
  - Or is removed

# How Extension Methods Work

- Unique to .NET
  - At least, among major languages
  - Requested feature in Java
- Compile-time trick
  - Compiles extension method call
  - As a direct call to the static method



# Demo 3: How Extension Methods Work

## Feature

Compare  
extension method  
invocations

## Task

Inspect IL output  
for extension  
method call &  
direct call

# **Demo 3: How Extension Methods Work**

# Demo 3: How Extension Methods Work

- Using extension methods
  - Calling the extension method on an instance

```
string xmlDateTime = new DateTime(2013,10,24).ToXmlDateTime();
```

- Generates IL

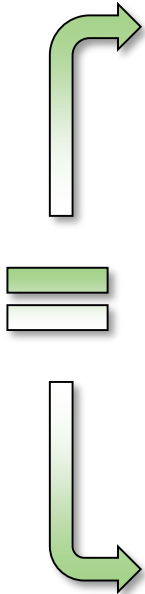
```
call string [Sixeyed.Extensions.Advanced]
 Sixeyed.Extensions.Advanced.Demo3.DateTimeExtensions::
 ToXmlDateTime(valuetype [mscorlib]System.DateTime)
```

- Calling the static method directly

```
string xmlDateTime = DateTimeExtensions.ToXmlDateTime(
 new DateTime(2013,10,24));
```

- Generates IL

```
call string [Sixeyed.Extensions.Advanced]
 Sixeyed.Extensions.Advanced.Demo3.DateTimeExtensions::
 ToXmlDateTime(valuetype [mscorlib]System.DateTime)
```





# Demo 3: How Extension Methods Work

- Writing extension methods
  - Method definition with this keyword in C#

```
public static string ToXmlDateTime(this DateTime dateTime)
```

- Generates Extension attribute in IL for method

```
...void [mscorlib]System.Runtime.CompilerServices.ExtensionAttribute
```

- And class

```
...void [mscorlib]System.Runtime.CompilerServices.ExtensionAttribute
```

- And assembly

```
.assembly Sixeyed.Extensions.Advanced
{
 ...void [mscorlib]System.Runtime.CompilerServices.ExtensionAttribute
```

# 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

