Practical Reflection

Presented by Jeremy Clark
www.jeremybytes.com
Just for Experts?
Explore the Practical Parts of Reflection

Goal

Safety

Performance

Flexibility
What is Reflection?

Inspecting the metadata and compiled code in an assembly.

- What is an assembly?
- What is metadata?
- How is the code compiled?
.NET Assemblies

Assembly
(exe or dll)

Module

Assembly Manifest

Metadata + IL

Resources (optional)

@jeremybytes
Type Definitions

```csharp
public class CachingClass
{
    private DateTime dataDate;
    private List<string> cachedItems;
}```
Assembly Information

Token: 0x20000001
Name: HackingAssemblies
Public Key:
Hash Algorithm: 0x00000004
Version: 1.0.0.0
Major Version: 0x00000001
Minor Version: 0x00000000
Build Number: 0x00000000
Revision Number: 0x00000000
Locale: <null>
Flags: [none] (00000000)
Referenced Assemblies

AssemblyRef #1 (23000001)

- Token: 0x23000001
- Public Key or Token: b7 7a 5c 56 19 34 e0 89
- Name: mscorlib
- Version: 4.0.0.0
- Major Version: 0x00000004
- Minor Version: 0x00000000
- Build Number: 0x00000000
- Revision Number: 0x00000000
- Locale: <null>
- HashValue Blob:
- Flags: [none] (00000000)
IL (Intermediate Language)

```csharp
public string DateTime
{
    get { return dataDate.ToString("HH:mm:ss"); }
}
```

```
.method public hidebysig specialname instance string get_DataTime() cil managed
{
    // Code size 17 (0x11)
    .maxstack 8
    IL_0000: ldarg.0
    IL_0001: ldflda valuetype [mscorlib]System.DateTime JeremyBytes.Library.CachingClass::dataDate
    IL_0006: ldstr "HH:mm:ss"
    IL_000b: call instance string [mscorlib]System.DateTime::ToString(string)
    IL_0010: ret
} // end of method CachingClass::get_DataTime
```
System.Reflection

- GetType
- GetMemberInfo
- GetMethodInfo
- Get PropertyInfo
- GetFieldInfo
  + many more

Activator

- CreateInstance

Assembly

- Load
- LoadFrom
- GetTypes
- GetName
- GetFiles
  + many more

Type

ILGenerator

- Emit
  + others
Things You Can Do

- Reflecting on a Property

```csharp
Type sampleType = typeof(SampleClass);
PropertyInfo cacheProperty = sampleType.GetProperty("CachedItems");
List<string> cacheValue = cacheProperty.GetValue(privateSample) as List<string>;
```

- Useful for interacting with COM objects (pre-.NET 4.0)
- “dynamic” is a better choice for interacting with COM
Things You Can Do

- Reflecting on a Method

```csharp
var list = new List<int>();
Type listType = typeof(List<int>);
Type[] parameterTypes = { typeof(int) };
MethodInfo addMethod = listType.GetMethod("Add", parameterTypes);
addMethod.Invoke(list, new object[] { 7 });
```

- Useful for interacting with COM objects (pre-.NET 4.0)
- “dynamic” is a better choice for interacting with COM
Things You Can Do

- Reflecting on a Private Field

```csharp
Type sampleType = typeof(SampleClass);
FieldInfo cacheField = sampleType.GetField("internalCache",
    BindingFlags.NonPublic | BindingFlags.Instance);
List<string> cacheValue = cacheField.GetValue(privateSample)
    as List<string>;
```

- BindingFlags give us access to non-public members
- DANGER DANGER DANGER DANGER
- Use the exposed interfaces
- Don’t peek inside the box
DEMO

Performance Concerns
Program to an abstraction rather than a concrete type
Practical Reflection Strategy

- **Dynamically Load Assemblies**
  - Happens one time (at start up)

- **Dynamically Load Types**
  - Happens one time (at start up)

- **Cast Types to a Known Interface**
  - All method calls go through the interface
  - No dynamic method calls – no MethodInfo.Invoke
  - Avoid interacting with private members

@jeremybytes
Various Data Sources

- Microsoft SQL Server
- MongoDB
- CSV
- SOAP Service
- Oracle
- WebAPI
- Amazon AWS
- JSON
- Microsoft Azure
- Hadoop

@jeremybytes
Pluggable Repositories

Application

- WCF Service Repository
- CSV File Repository
- SQL Database Repository

@jeremybytes
DEMO

Run-Time Binding
Benefits of Dynamic Loading

• Only ship 1 repository assembly

• Remove dependency on concrete repositories

• New repositories can be added without modifying exiting code
Assembly-Qualified Type Name

PersonRepository.SQL.SQLRepository,
PersonRepository.SQL,
Version=1.0.0.0,
Culture=neutral,
PublicKeyToken=b77a5c561934e089

• Fully-qualified type name (namespace and type)
• Assembly Name
• Assembly Version
• Assembly Culture
• Assembly Public Key (for strongly-named assemblies)
private void FetchButton_Click(object sender, EventArgs e)
{
    ClearListBox();

    var people = repository.GetPeople();
    foreach (var person in people)
    {
        PersonListBox.Items.Add(person);
    }

    ShowRepositoryType(repository);
}

- No Reflection Here
- Method calls through IPersonRepository
Scenario

Order Entry Application

Client #1
Business Rule

Client #2
Business Rule

Client #3
Business Rule

@jeremybytes
Customer: John Crichton

3/19/1999

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal Translator</td>
<td>1</td>
<td>$199.99</td>
</tr>
<tr>
<td>Starship</td>
<td>1</td>
<td>$5,999,999.99</td>
</tr>
<tr>
<td>Space Suit</td>
<td>4</td>
<td>$2,399.96</td>
</tr>
<tr>
<td>Laser Beacon</td>
<td>1</td>
<td>$99.95</td>
</tr>
</tbody>
</table>
public interface IOrderRule
{
    string RuleName { get; }
    OrderRuleResult CheckRule(Order order);
}

public class OrderRuleResult
{
    public bool Result { get; set; }
    public string Message { get; set; }

    public OrderRuleResult(bool result,
        string message) {...}
}
Business Rules

- Maximum Discount based on Customer Rating
- Only 1 Captain’s Chair Allowed
- Maximum of 1 Starship per Order
- Name Badge must match Customer Name
Discovery Process

• Locate all assemblies in the “Rules” folder

• Load each assembly

• Enumerate the types in the assembly

• Check each type to see if it implements our Rule interface

• Create an instance of each Rule and add it to the Rule Catalog
Thank You!

Jeremy Clark

• http://www.jeremybytes.com
• jeremy@jeremybytes.com
• @jeremybytes