Fachhochschule Nordwestschweiz
Hochschule für Technik

DYNAMIC AND INTEROP

Martin Kropp, Yves Senn
University of Applied Sciences Northwestern Switzerland

Learning Targets

- You
  - can explain the core concepts of the dynamic language runtime
  - can use the dynamic type in C# programs
  - can integrate dynamic languages into .NET
  - can explain the COM-Interop concepts

Content

- The Dynamic Language Runtime
  - Introduction
  - Using Dynamic in C#
  - Implementing Dynamic Objects in C#
  - Dynamic Language Integration
- COM-Interop (Framework)
  - Using Dynamics for COM-Interop

Fachhochschule Nordwestschweiz
Hochschule für Technik

Dynamic vs. static typing

- Statically typed languages:
  Java, C++, Haskell, …
  - Type checking at compile time
  - Great IDE support (IntelliSense, Refactoring, …)
  - Performance and robustness
- Dynamically typed languages:
  JavaScript, Python, PHP, …
  - Type evaluation at runtime
  - No type checks during compilation
    or no compilation
    → Exceptions at runtime
  - Sub-par IDE support

.NET/C# supports both ways

Dynamically typed objects

```
Calculator calc = GetCalculator();
int sum = calc.Add(10, 20);
```

If `Calculator` is implemented in JavaScript, we can’t know the type of `calc` or return types at compile time.

→ We have to invoke methods *dynamically*:

```
object calc = GetCalculator();
Type calcType = calc.GetType();
object res = calcType.InvokeMember("Add",
    BindingFlags.InvokeMethod, null, calc, new object[] { 10, 20 });
int sum = Convert.ToInt32(res);
```

```
dynamic calc = GetCalculator();
int sum = calc.Add(10, 20);
```

Dynamic conversion

Dynamic method invocation

Fachhochschule Nordwestschweiz
Hochschule für Technik

Dynamically typed objects

- Interoperability with dynamic languages and frameworks
- Keyword: **`dynamic`**

```
dynamic topic = "Dynamic Types";
```

Caveats

- Deactivates type checking
- No IntelliSense
- "Everything" compiles:

```
dynamic speaker = new NonSenseTalker();
speaker.HelloWorld();
speaker.Im().Just().Demonstrating().This().Feature();
speaker.GoodBye();
```

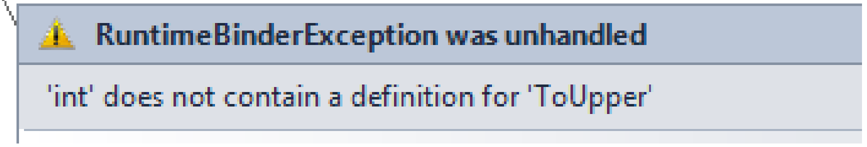
Fachhochschule Nordwestschweiz
Hochschule für Technik

Dynamically typed objects

Example:

```
dynamic d = "Test";
Console.WriteLine(d);    // Output: Test
d = d.ToUpper();
Console.WriteLine(d);    // Output: TEST

d = 3;                   // d changes its runtime type to int
Console.WriteLine(d);    // Output: 3
d += 7;
Console.WriteLine(d);    // Output: 10
d.ToUpper();             // Exception at run time
```



RuntimeBinderException was unhandled

'int' does not contain a definition for 'ToUpper'

Dynamically typed objects

```
public static class Math
{
    public static decimal Abs(decimal value);
    public static double Abs(double value);
    public static float Abs(float value);
    public static int Abs(int value);
    public static long Abs(long value);
    public static sbyte Abs(sbyte value);
    public static short Abs(short value);
```

Method chosen at compile-time:
`double Abs(double x)`

```
double x = 1.75
var y = Math.Abs(x);
```

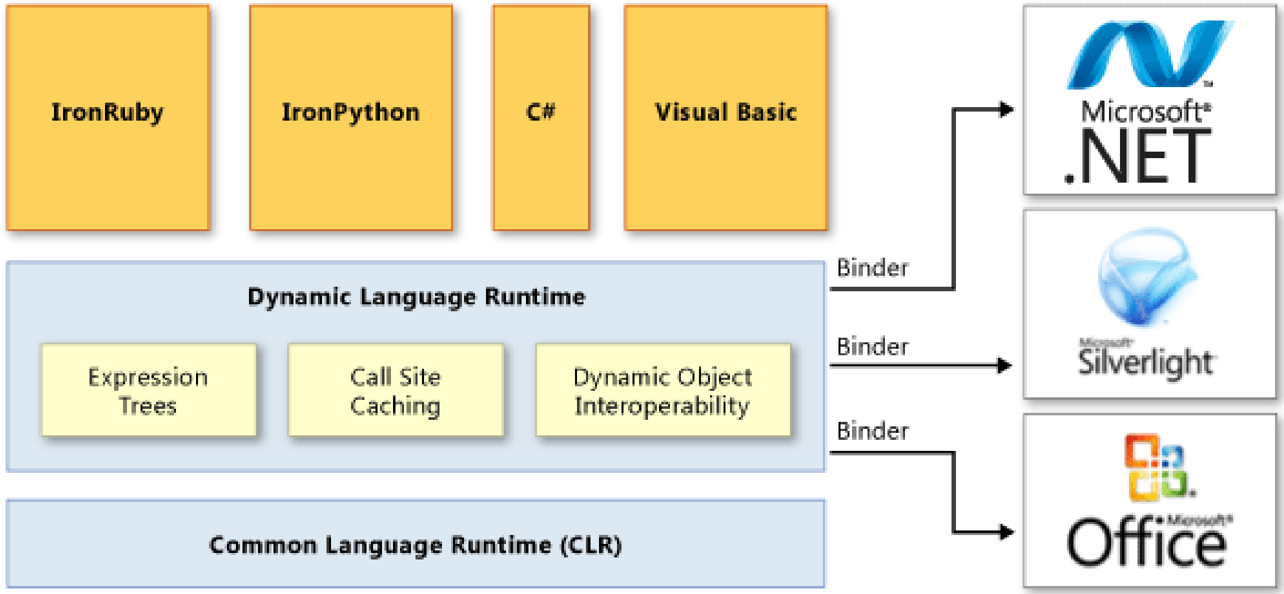
```
dynamic x = 1.75;
var y = Math.Abs(x);
```

Method chosen at run-time:
`double Abs(double x)`

```
dynamic x = 2;
var y = Math.Abs(x);
```

Method chosen at run-time:
`int Abs(int x)`

Dynamic Language Runtime



- DLR adds services to CLR supporting dynamic languages

Running Python in .NET

```
static dynamic Calculate(string expression)
{
    var engine = Python.CreateEngine();
    return engine.Execute(expression);
}


static void Main(string[] args)
{
    dynamic result = Calculate("2 * 3");
    Console.WriteLine("result: {0}", result);
}
```

➔ Enables use of dynamic languages in .NET

Worksheet – Aufgabe 1

Key concepts

- Expression trees

  DLR extends LINQ expression trees with control flow, assignment, other language-modeling nodes for internal use.

- Call-Site caching

  Optimization for fast dynamic binding.

- Dynamic object interop

  Dynamic binding and dispatching objects and method calls

ExpandoObject

ExpandoObjects implements dynamic properties:

```
dynamic o = new ExpandoObject();

o.Test = 123;
o.Foo = "hallo";
o.Bar = new object();
o.F = (Action)(() => Console.WriteLine("Action done"));
o.F();

Console.WriteLine(o.Foo);
```

Implementing dynamic Types

```
class Duck : DynamicObject
{
   public override bool TryInvokeMember(
               InvokeMemberBinder binder, object[] args,
               out object result)
   {
       Console.WriteLine(binder.Name + " method was called");
       result = null;
       return true;
   }
}

static void Main(string[] args)
{
    dynamic duck = new Duck();

    duck.Quack();
    duck.Waddle();
}
```

Dynamic Objects API

- Call dynamic methods
```
public virtual bool TryInvokeMember(InvokeBinder
binder, Object[] args, out Object result)
```

- Get dynamic properties
```
public virtual bool TryGetMember(GetMemberBinder
binder, out Object result)
```

- Set dynamic properties
```
public virtual bool TrySetMember(SetMemberBinder
binder, Object value)
```

Interoperating with unmanaged code

Fachhochschule Nordwestschweiz
Hochschule für Technik

Interoperating with unmanaged code

- You can execute unmanaged/native code from C# with **P/Invoke**
  - e.g. C++ DLLs

Example: Calling C++ MessageBox method

```
class Test
{
    [DllImport("user32.dll")]
    static extern int MessageBox(uint hWnd, string text, string caption, uint type);

    static void Main()
    {
        MessageBox(0, "Calling native code - isn't that cool?", "", 1);
    }
}
```

Challenges

- Method API
- Mapping Data Types
- Memory alignment
- Memory management
- Pointers
- Passing managed allocated memory

Fachhochschule Nordwestschweiz
Hochschule für Technik

Working with P/Invoke

Example: Show dialog box with native Win32-API

1. Find signature on http://pinvoke.net/

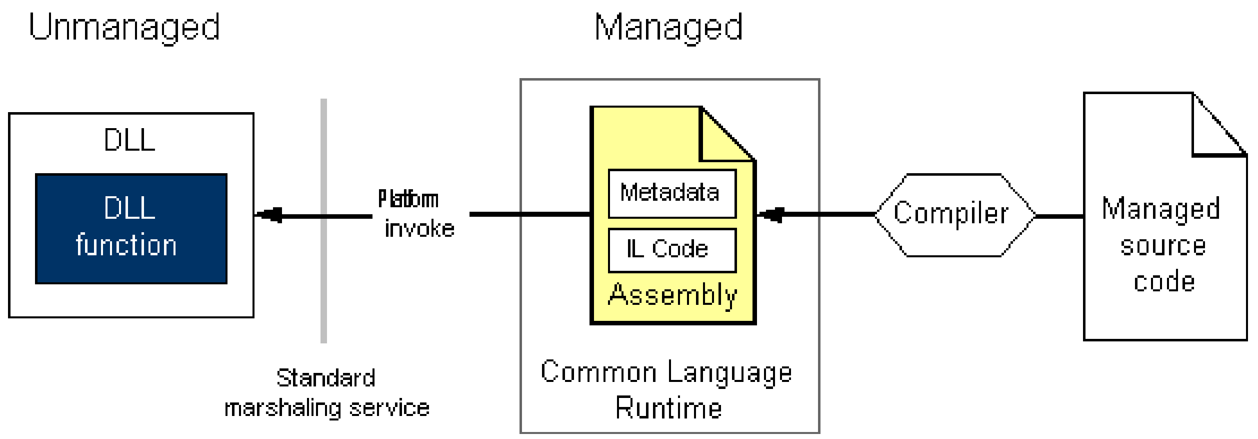
```
int MessageBox(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);
```

2. Invoke it from C#

```
class Test
{
    [DllImport("user32.dll")]
    static extern int MessageBox(uint hWnd, string text, string caption, uint type);

    static void Main()
    {
        MessageBox(0, "Calling native code - isn't that cool?", "", 1);
    }
}
```

P/Invoke behind the scenes



http://msdn.microsoft.com/en-us/library/0h9e9t7d.aspx

1. Locates the DLL containing the function.
2. Loads the DLL into memory.
3. Locates the address of the function in memory and pushes its arguments onto the stack, marshaling data as required.
4. Transfers control to the unmanaged function

COM interoperability (Framework)

- COM stands for «Component Object Model»
- Used internally by Windows and many applications
- Mostly used to control Microsoft Office Apps (Word, Excel, Outlook, …)
  - Controlling Excel from your application
  - To build plugins

Fachhochschule Nordwestschweiz
Hochschule für Technik

COM Interop with MS Office

COM interop with .NET and the DLR:

```
using Word = Microsoft.Office.Interop.Word;

var word = new Word.Application();
word.Documents.Add();
word.ActiveDocument.SaveAs("test.doc");
```

Demo – Excel & Word Interop

Worksheet – Aufgabe 2

Dynamic resources

- Microsoft DLR Home
  http://msdn.microsoft.com/en-us/library/dd233052.aspx
- IronPython Project
  http://ironpython.net/
- IronPython Cookbook
  http://www.ironpython.info
- Mark Gu about DLR
  http://markyourfootsteps.wordpress.com/2010/04/20/dynamic-programming-dlr/