Advanced C#

Dynamic

The problem (Silverlight)

Silverlight can communicate with the host page using javascript objects:

```
public static void ChangeMenu(ScriptObject menuObject)
        {
            menuObject.Invoke("setMenuColor", Colors.Green);
            menuObject.Invoke("setMenuWidth", menuObject.Invoke("getMenuWidth") +
                                        10);
            menuObject.Invoke("setMenuHeight", menuObject.Invoke("getMenuHeight“) +
                                        10);
        }
```

The problem (COM Interop)

Reading a Word document's custom properties in C#:

```
public string GetCustomPropertyValue(Document doc, string propertyName)
{
    object oDocCustomProps = doc.CustomDocumentProperties;
    Type typeDocCustomProps = oDocCustomProps.GetType();
    object oCustomProp = typeDocCustomProps.InvokeMember("Item",
                                            BindingFlags.Default |
                                            BindingFlags.GetProperty,
                                            null, oDocCustomProps,
                                            new object[] { propertyName });
    Type typePropertyValue = oCustomProp.GetType();
    string propertyValue = typePropertyValue.InvokeMember("Value",
                                            BindingFlags.Default |
                                            BindingFlags.GetProperty,
                                            null, oCustomProp,
                                            new object[] { }).ToString();
     return propertyValue;
}
```

The solution (Silverlight)

With dynamic:

```
public static void ChangeMenu(ScriptObject menuObject)
{
        menuObject.Invoke("setMenuColor", Colors.Green);
        menuObject.Invoke("setMenuWidth",
                          menuObject.Invoke("getMenuWidth")  + 10);
        menuObject.Invoke("setMenuHeight",
                          menuObject.Invoke("getMenuHeight")  + 10);
}
```

The problem (COM Interop)

Without dynamic:

```
public string GetCustomPropertyValue(Document doc, string propertyName)
{
    dynamic oDocCustomProps = doc.CustomDocumentProperties;
    dynamic oCustomProp = oDocCustomProps[propertyName];
    return oCustomProp.Value;
}
```

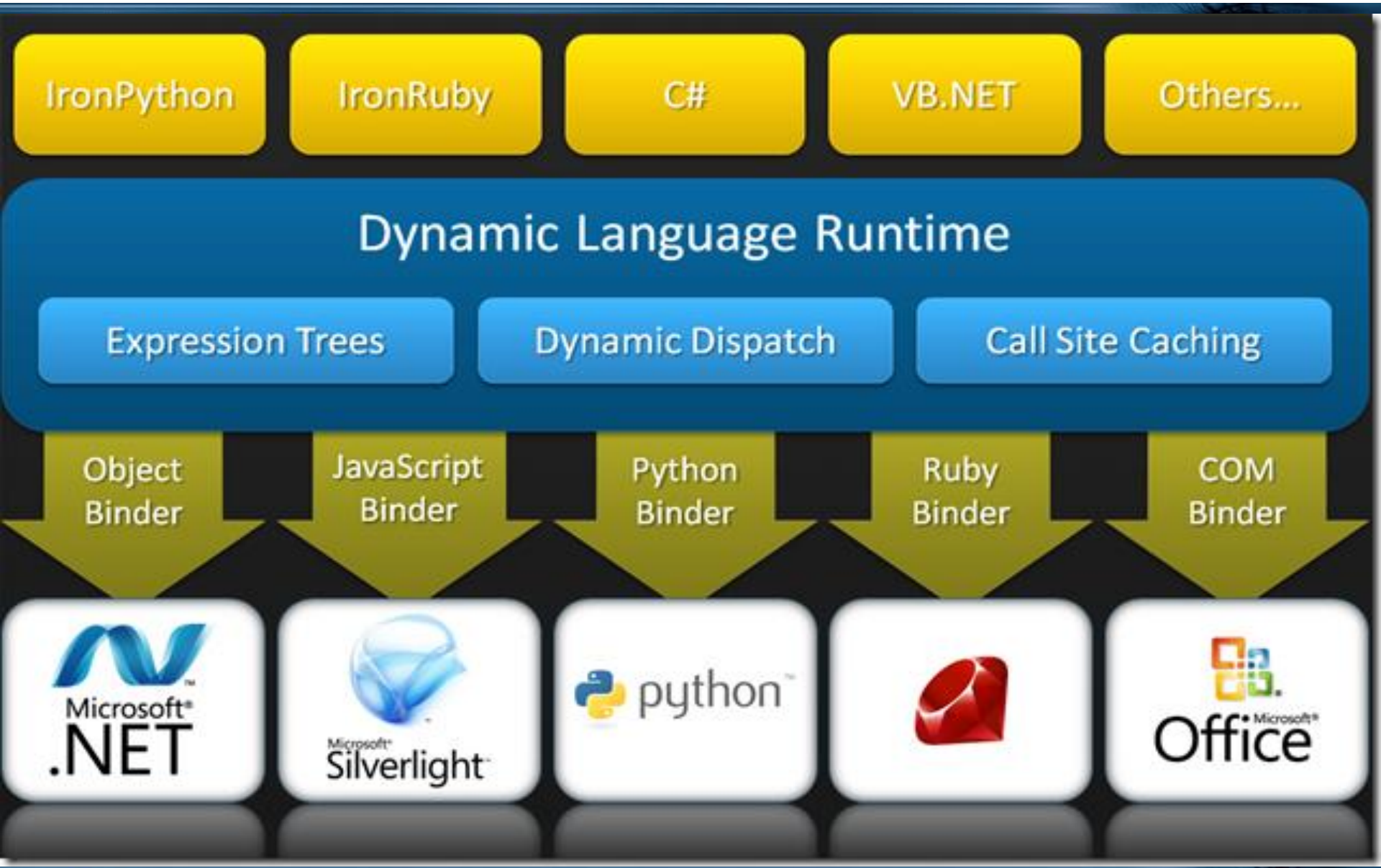
```
                                        BindingFlags.GetProperty,
                                        null, oDocCustomProps,
                                        new object[] { propertyName });
```

Return types are typed as dynamic:

```
public string GetCustomPropertyValue(Document doc, string propertyName)
{
    return doc.CustomDocumentProperties[propertyName].Value;
}
```

```
                                        null, oCustomProp,
                                        new object[] { }).ToString();
     return propertyValue;
}
```

Dynamic overview



Dynamic consequences

```
dynamic d = 10;
C c = new C();
// (1) Dynamic receivers:
d.Foo(); // Call.
d.PropOrField = 10; // Property.
d[10] = 10; // Indexer.
// (2) Statically typed receivers (or static methods)  with dynamic arguments.
c.Foo(d); // Instance method call.
C.StaticMethod(d); // Static method call.
c.PropOrField = d; // Property.
c[d] = 10; // Indexer.
d++; // Think of this as op_increment(d).
var x = d + 10; // Think of this as op_add(d, 10).
int x = d; // Think of this as op_implicit(d).
int y = (int)d; // Think of this as op_explicit(d). }
```

When operand(s) are ***dynamic:***

- Member selection deferred to run-time.
- At run-time, actual type(s) substituted for ***dynamic.***
- At compile-time, result type of operation will be ***dynamic.***

Dynamic overloading

What will be the output?

```
public class C
{
    public void Foo(decimal x)
    {
        Console.WriteLine("Decimal");
    }
    public void Foo(int x)
    {
        Console.WriteLine("Integer");
    }
    static void Main(string[] args)
    {
        C c = new C();
        dynamic d = 10;
        c.Foo(d);
    }
}
```

The output will be “Integer”.

Dynamic overloading continued

What will be the output?

```
public class C
{
    public void Foo(decimal x)  {  Console.WriteLine("Decimal"); }

    static void Main(string[] args)
    {
        C c = new D();
        dynamic d = 10;
        c.Foo(d);
    }
}

public class D : C
{
     public void Foo(int x) { Console.WriteLine(“Integer");  }
}
```

The output will be “Decimal”.

Dynamic overloading concluded

What will be the output?

```
public class C
{
    public void Foo(decimal x)  {  Console.WriteLine("Decimal"); }

    static void Main(string[] args)
    {
        C c = new D();
        dynamic d = "Hello";
        c.Foo(d);
    }
}

public class D : C
{
     public void Foo(string s) { Console.WriteLine(“String");  }
}
```

The output will be a “RuntimeBinderException”.

Runtime lookup order

1. Checks whether dynamic receiver is a COM object. If so, IDispatch is used to complete the operation.
2. If dynamic receiver implements IDynamicObject, the receiver itself is asked to complete the operation.
3. If both of the above are not true, the dynamic receiver is a plain .Net object and reflection is used to complete the operation.

What dynamic can’t do

1. Dynamic lookup will not be able to find extension methods.
2. You can’t pass a lambda expression as an argument in a dynamic method call.
3. When inheriting, you can’t call a base method overload with dynamic arguments.
4. Dynamic lookup will not be able to find explicitly implemented interface methods.