

The .NET Framework

as a platform for Domain-Specific- and Scripting Languages

Henrik Stuart

9th March 2006

Overview

- Motivation
- The .NET framework
- Embedding scripting languages
- Web-based domain-specific languages

Motivation

Python in C

Steps to make functions available:

- Write function
- Write wrapper function
- Declare functions
- Load functions into a Python module

Motivation

Python in .NET

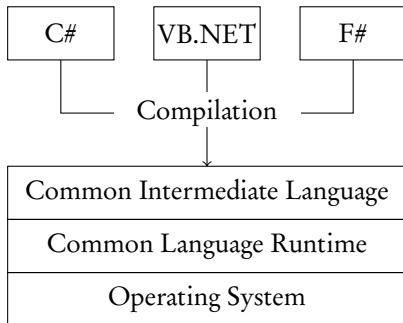
Steps to make functions available:

- Write function

- Load functions into a Python module

The .NET framework

Architecture overview



The .NET framework

Functionality highlights

- Typed intermediate language (CIL)
- Language interoperability (CTS/CLS)
- Extensive base class library (BCL)
- Good overall performance through JIT compilation
- Over 40 languages for .NET including: Scheme, Lisp, SML, O'Caml, Haskell, Java and Fortran

Embedding Scripting Languages

IronPython

- Created by Jim Hugunin of Jython fame
- Now made by Microsoft
- Compiles and integrates seamlessly with the CLR

Embedding Scripting Languages

IronPython — Making a C# class available

Example (A C# Class)

```
namespace Hello {  
    using System;  
  
    public class Test {  
        public void Hello(int times) {  
            while (times-- != 0) Console.WriteLine("Hello World");  
        }  
    }  
}
```

Embedding Scripting Languages

IronPython — Using the C# class library

Example

```
from Hello import *
```

```
t = Test()
```

```
t.Hello(3)
```

Embedding Scripting Languages

IronPython — Executing the Python file from a C# program

Example

```
using IronPython.Hosting;
using System.Reflection;
using Hello;

static class Program {
    static void Main(string[] args) {
        PythonEngine engine = new PythonEngine();
        engine.LoadAssembly(Assembly.GetAssembly(typeof(Test)));
        engine.ExecuteFile("python-example.py");
    }
}
```

Embedding Scripting Languages

IronPython — Running the example

Output

```
> example  
Hello World  
Hello World  
Hello World
```

Embedding Scripting Languages

IronPython — Passing variables between C# and IronPython

Example

```
using IronPython.Hosting;
using System;
using Hello;

static class Program {
    static void Main(string[] args) {
        PythonEngine engine = new PythonEngine();
        engine.SetVariable("t", new Test());
        engine.ExecuteFile("python-example2.py");
        Console.WriteLine(engine.GetVariable("q"));
    }
}
```

Embedding Scripting Languages

IronPython — Passing variables between C# and IronPython

Example

```
t.Hello(3)  
q = "Goodbye World"
```

Embedding Scripting Languages

IronPython — Running the example

Output

```
> example  
Hello World  
Hello World  
Hello World  
Goodbye World
```

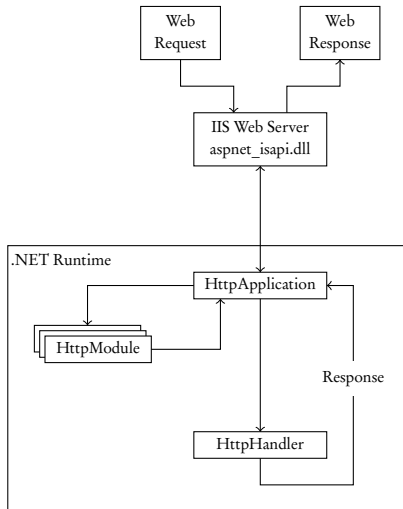
Embedding Scripting Languages

IronPython — Conclusions

- Moving between IronPython and the host language is easy — *very easy!*
- Provides almost all functionality for Python
- Can use existing code written in Python

Domain Specific Languages

The web pipeline in .NET



Domain Specific Languages

Overview

- ASP.NET
- A custom Python handler
- .ida

Domain Specific Languages

ASP.NET — Overview

- Server-side XML-based language
- Embedded in HTML/XHTML
- Includes scripting in C# and VB.NET

Domain Specific Languages

ASP.NET — A simple dynamic page

Example (Simple Page)

```
<% @Page Language="C#" %>
<html>
  <head>
    <title>Test</title>
  </head>
  <body>
    <form runat="server">
      <asp:Button Text="Hello!" runat="server"
        OnCommand="Hello" />
      <asp:Label ID="HelloLabel" runat="server" />
    </form>
```

Domain Specific Languages

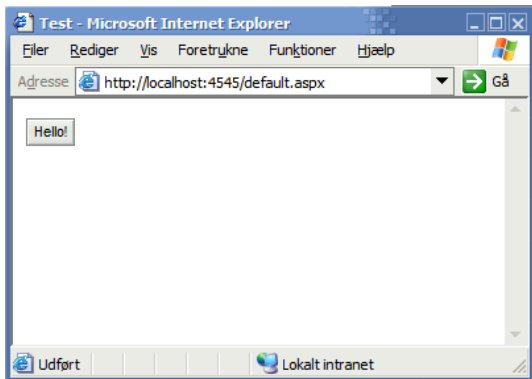
ASP.NET — A simple dynamic page

Example (Simple Page)

```
<script runat="server">  
    void Hello(object sender, System.EventArgs e) {  
        HelloLabel.Text = "Hello!";  
    }  
</script>  
</body>  
</html>
```

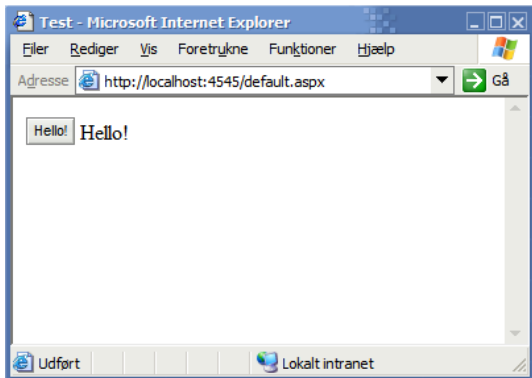
Domain Specific Languages

ASP.NET — A simple dynamic page



Domain Specific Languages

ASP.NET — A simple dynamic page



Domain Specific Languages

ASP.NET — The basic tag

Example (Basic tag)

```
<tag:element attributes>...</tag:element>
```

Domain Specific Languages

ASP.NET — A custom tag

Example (Custom tag)

```
<hs:Hello />
```

Domain Specific Languages

ASP.NET — Creating a custom control

Example (Hello custom control)

```
<% @Control Language="C#" %>
<asp:Button Text="Hello!" runat="server" OnCommand="Hello" />
<asp:Label runat="server" ID="HelloLabel" />
<script runat="server">
    void Hello(object sender, System.EventArgs e) {
        HelloLabel.Text = "Hello!";
    }
</script>
```

Domain Specific Languages

ASP.NET — Consuming the custom control

Example (Hello consumer)

```
<% @Page Language="C#" %>
<% @Register TagPrefix="hs" TagName="Hello" Src="hello.ascx" %>
<html>
  <head><title>Test</title></head>
  <body>
    <form runat="server">
      <hs:Hello runat="server" />
    </form>
  </body>
</html>
```

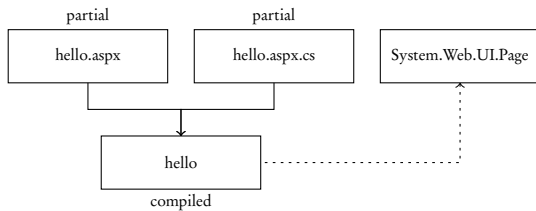
Domain Specific Languages

ASP.NET — The in-page model

- Changes in pages are picked up dynamically
- We can modularise page content with custom controls
- Code inside the page

Domain Specific Languages

ASP.NET — The partial model



Domain Specific Languages

ASP.NET — The partial page

Example

```
<% @Page Language="C#" codefile="hellopage.aspx.cs"
    inherits="HelloPage" %>
<html>
  <head><title>Test</title></head>
  <body>
    <form runat="server">
      <asp:Button runat="server" Text="Hello!"
        OnCommand="Hello" />
      <asp:Label ID="Test" runat="server" />
    </form>
  </body>
</html>
```

Domain Specific Languages

ASP.NET — The partial code

Example

```
public partial class HelloPage : System.Web.UI.Page {  
    protected void Hello(object sender, System.EventArgs e) {  
        Test.Text = "Hello!";  
    }  
}
```

Domain Specific Languages

ASP.NET — Conclusion

- The main, prepackaged `HttpHandler`
- Modularity through controls and code-behind
- Entire page-model is object-oriented
- Good security support (although not covered here)
- **Everything must be reentrant**

Domain Specific Languages

Python – The HTTP handler

Example

```
namespace PyHandler {  
    using System.Web;  
    using IronPython.Hosting;  
  
    public class PythonHandler : IHttpHandler {  
        public bool IsReusable { get { return true; } }  
  
        public void ProcessRequest(HttpContext context) {  
            PythonEngine engine = new PythonEngine();  
            engine.SetVariable("Response", context.Response);  
            engine.ExecuteFile(context.Request.PhysicalPath);  
        }  
    }  
}
```

Domain Specific Languages

Python — Enabling the handler

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.web>
    <httpHandlers>
      <add verb="*" path="*.py"
          type="PyHandler.PythonHandler, PyHandler" />
    </httpHandlers>
  </system.web>

</configuration>
```

Domain Specific Languages

Python — Using the handler

Example (page.py)

```
Response.Write("Hello World")
```

Output

```
Hello World
```

Domain Specific Languages

.ida — Overview

- ASP.NET requires a programmer
- Reduce repeated markup
- XML-based due to laziness
- Integrates XHTML
- Supports internationalisation
- Handler is based on XSLT

Domain Specific Languages

.ida — Example


Example (index.ida)

```
<?xml version="1.0" encoding="utf-8" ?>
<page title="Ida's Homepage">
  <info>
    
  </info>
  <body>
    <h2>Heading</h2>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Curabitur nec lorem. Nam sed lectus. Aenean non purus.
      Curabitur ante odio, iaculis et, suscipit egestas.
    </p>
  </body>
</page>
```

Domain Specific Languages

.ida — Example

Danish · English †



Ida's Homepage

[Hjem](#) · [Vægmaleries](#) · [Malerier](#) · [Akvarel](#) · [Tegninger](#) · [Julepynt](#) · [Bestillingsarbejde](#)

Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec lorem. Nam sed lectus. Aenean non purus. Curabitur ante odio, iaculis et, suscipit egestas.



Copyright 2005 by Ida Hamburger

[Hjem](#) · [Vægmaleries](#) · [Malerier](#) · [Akvarel](#) · [Tegninger](#) · [Julepynt](#) · [Bestillingsarbejde](#)

† Language selection requires cookies.

Copyrighted 2003-2005 by Ida Hamburger

Domain Specific Languages

Conclusion

- The web pipeline allows for easy modification
- Easy to add new HTTP handlers — *very easy*
- Solid platform for functionality

Conclusion

- Type system makes interoperability easy
- Modular web pipeline allows new functionality easily
- Extensive base library makes programming easy
- Targetting .NET is easy

It's just easy

It's just easy

(No, I'm not on Microsoft's payroll)