.NET Overview

José M. Vidal

Tue Apr 13 15:12:53 EDT 2004

This talk gives a very quick overview of the .NET architecture for people interested in distributed programming with SOAP.

1 What is .NET?

- "The .NET Framework is an integral Windows component for building and running the next generation of software applications and Web services. The .NET Framework:
  - Supports over 20 different programming languages.
  - Manages much of the plumbing involved in developing software, enabling developers to focus on the core business logic code.
  - Makes it easier than ever before to build, deploy, and administer secure, robust, and high-performing applications." – Microsoft .NET[1]

2 Architecture

<table>
<tr><td colspan="2">Web Services</td></tr>
<tr><td colspan="2">Frameworks and Libraries: ASP.NET, ADO.NET, Window Forms</td></tr>
<tr><td>Interchange standards:
SOAP, WSDL, UDDI</td><td>Common Development Tools:
Visual Studio .NET</td></tr>
<tr><td colspan="2">Component Model</td></tr>
<tr><td colspan="2">Object Model and Common Language specification</td></tr>
<tr><td colspan="2">Common Language Runtime</td></tr>
</table>

- **Web services**: are the applications built with .NET. These services will be provided to users.
- **Frameworks and libraries**: are a bunch of libraries available for developers to use.
- **Interchange standards** are more libraries. These ones support SOAP, et.al.
- **Development Environment**: Everybody use Visual Studio!
- **Component Model**: Like CORBA, COM, and J2EE, but different.
- **Object Model and CLS**: place restrictions on all program languages so that they can run in the
- **Common Language Runtime** which is a (JIT) virtual machine with support for versioning.

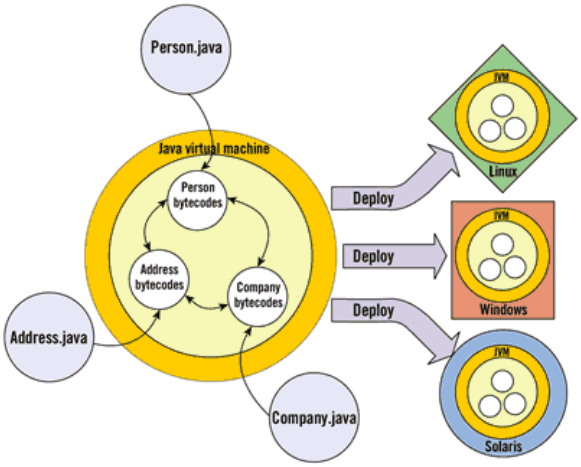
3 ASP.NET

- Makes it very easy to turn a program (with a GUI) into a web application.
- The code downloaded to the browser is dependent on the type of browser.
- It maintains state.

4 Component Model

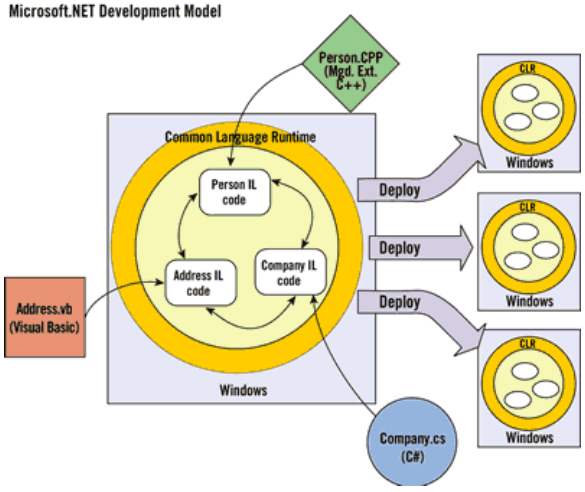
- .NET uses **assemblies** which are compiled and versioned collections of code and metadata that form an atomic functional unit.
- An assembly contains the code, the version number, a list of all the other assemblies (with version number and public keys) it requires to run, and a description of the interface it provides.
- Assemblies are generated for you by VS.
- No IDL is needed. The interface is generated automatically (as with RMI, but supporting many languages).
- The code in these assemblies runs in the common language runtime.

5 Language Interoperability



Java/J2EE Development Model

Microsoft.NET Development Model



6 Building a .NET Web Service

1. Write a class with **attributes** (comments) that identify it as a web service with exposed methods.

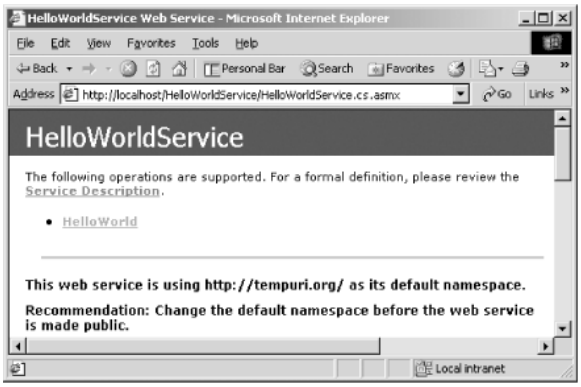
2. .NET uses these to create WSDL document.
3. Client adds service as **web reference** in Visual Studio .NET, or runs WSDL.exe to create stubs.
4. .NET uses WSDL file to generate proxy (stub) classes.
5. Client instantiates proxy class and calls methods on it.
6. Proxy class converts it to a SOAP message and sents it.
7. Proxy class receives SOAP reply, parses it, and returns value to client.

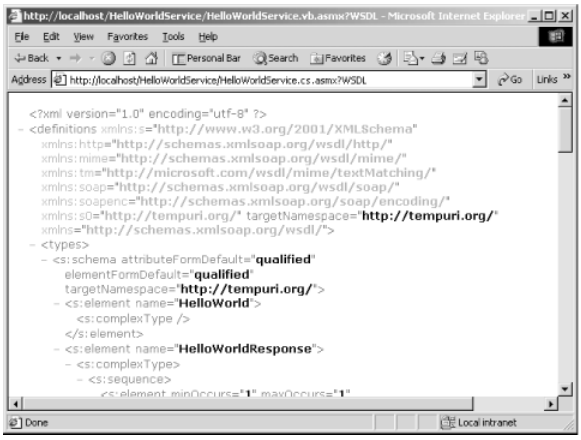
6.1 Placing Attributes

```
<%@ WebService Language="C#"
Class="ProgWS.HelloWorldService" %>
using System.Web.Services;
namespace ProgWS
{
  public class HelloWorldService: WebService
  {
    [WebMethod]
    public string HelloWorld()
    {
      return "Hello World";
    }
  }
}
```

- You must specify the webservice directive and give it a language and class.
- The `WebMethod` can have
  - `Description`: a text description.
  - `MessageName`: the name of the SOAP message.
  - `EnableSession`: HTTP sessions.
  - `CacheDuration`: how long to cache request/response pair.
  - `TransactionOption`: is it a database transaction? (atomic, can be rolled back)

6.2 Accessing





- In VS .NET you can compile the file into an assembly (.asmx).
- If you also have IIS installed, you can then acces the webservice via a browser (to check only),
- or get its WSDL.

6.3 The Client

- First, add a wrb service to VS .NET as a Web Reference by giving the URL of its .asm.
- The stub class (`DNSLookupService`) is added automatically, so you can then use it:

```
using System;
namespace DNSConsumerApp
{
  class Consumer
  {
    static void Main(String[] args)
    {
      localhost.DNSLookupService objDNS= new localhost.DNSLookupService();
      string strIPAddress = "";
      strIPAddress = objDNS.getIPForHostnae(args[0]);
      Console.WriteLine("Hostname: " + args[0] + " IP: " + strIPAddress);
    }
  }
}
```

6.4 Asynchronous Invocation

- The genrated stubs have two methods that allow one to call web services asynchronously.

```
class proxy {
  ...
  public System.IAsyncResult BeginDelay(System.AsyncCallback callback,
                                        object asyncState){
  return this.BeginInvoke("Delay", new object[0], callback, asyncState);
}
  public int EndDelay(System.IAsyncResult asyncResult){
    object[] results = this.EndInvoke(asyncResult);
```

```
    return((int)(results[0]));
  }
}
```

- Start by calling `BeginDelay`. It returns an object that allows us to check if the request has been done.

7 Summary

- .NET Web Services uses SOAP, WSDL, and UDDI.
- VS .NET makes web services appear to be local (like RMI object), but they are not.
- VS .NET and IIS and MS Windows can be used together to make deployment trivial, but everything comes from the same vendor.

Notes

[1] http://msdn.microsoft.com/netframework/technologyinfo/overview/

This talk is available at `http://jmvidal.cse.sc.edu/talks/dotnetoverview`