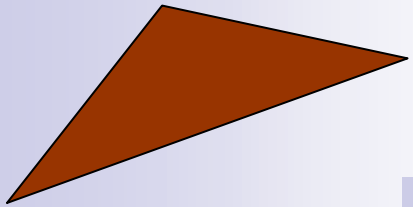

XSLT, .NET, and Web Services

Chris Dix

Dart Communications

Introductions

- Chris Dix
  - Lead Developer for PowerTCP Components at Dart Communications
  - http://www.dart.com/
  - XML Schema Complete Reference (Addison-Wesley, 2002)
  - Professional XML Web Services (Wrox, 2000)

XSLT

- eXtensible Stylesheet Language: Transformations
- XML syntax for transforming XML documents

XSLT Basics

- Templates
  - Basic unit of execution for a transformation
  - Can be associated with an XPath pattern or can be named and executed explicitly
  - <xsl:template >
  - <xsl:apply-templates >
  - <xsl:call-template >

XSLT Basics

- Functional Programming
- XPath Expressions
- Variables
- Conditional Logic

XSLT and Web Services

- XSLT is a great tool for manipulating XML
- Don Box once described SOAP as *“XSLT with a longer wire”*
- The execution of a stylesheet in response to a SOAP request can generate a SOAP response

XSLT and Web Services

- When using XSLT, there is no “language barrier”
- XML content of the payload is directly accessible via XPath expressions
- There is a good and bad side to this

Is this a good idea?

- It boils down to language-preference
- There are a lot of ways this could be done
  - .NET Remoting Extension?
- There are a lot of improvements that could be made that will not discussed today

XSLT As Endpoint

- What is missing?
  - XSLT can’t listen for an HTTP request
    - Transport protocol support must be separate
  - XSLT is “typeless” (ok, it has 5 types)
    - Type information must be added somehow

Example: Simple XSLT Web Service

- This needs:
  - A stylesheet to validate the SOAP message and perform the transformation of the request payload into the response
  - Code to trigger the transformation

Example Stylesheet Basics

- SOAP Envelope processing
  - Request
  - Response
- Payload manipulation
  - Request
  - Response

Example Listener Basics

- .NET Code to Handle HTTP Request
- .NET Code to Trigger Transformation

- This code could be written in a lot of different ways; I chose .NET, but the principle is widely applicable

IHttpHandler

```
using System;
using System.Web;
using System.Xml;
using System.Xml.Xsl;
using System.Xml.XPath;

namespace kafka
{
    public class SoapHttpRequestHandler : IHttpHandler
    {
          public SoapHttpRequestHandler()            {}

          public void ProcessRequest(HttpContext context)
          {…}

          public bool IsReusable
          {
                  get { return false; }
          }
    }
}
```

.NET and XSLT

- **System.Xml**
  - **System.Xml.Xsl**
- These contain the classes for using XSLT in .NET
  - **XslTransform**
  - **XmlTextWriter**
  - **XPathDocument**

.NET Transformations of an HTTP Request

```
XslTransform xsltExec = new XslTransform();
xsltExec.Load( context.Request.MapPath( “code.xslt" ) );

XmlTextWriter refResponse = new XmlTextWriter(
   context.Response.OutputStream,
   new System.Text.UTF8Encoding() );
refResponse.Formatting = Formatting.Indented;

// Process message
XPathDocument docRequest = new XPathDocument(
   context.Request.InputStream );
context.Response.ContentType = "text/xml";
xsltExec.Transform( xpathdocument, null, refResponse );
```

Starting From There

- A combination of stylesheets with minimal “engine” code can function as a framework for service development
- The simple Web Service stylesheet provides a foundation for such a framework

Building the Framework

- Goal:
  - Make the developer unaware of the nature of Web Services
- In XSLT, templates are the unit of work, so they map to operations of a Web Service

XSLT Templates as Operations

- It would be difficult to match XSLT templates to Web Service operations unless we use named templates
- Named templates are, by definition, named, and they have parameter lists as well

Named Template for echoString

```
<xsl:template name="echoString" >
  <xsl:param name="inputString" />
      <xsl:value-of select="$inputString"/>
</xsl:template>
```

- This is an XSLT named template that only echoes the value passed into the inputString parameter
- The goal is to expose this as a Web Service operation

Extensibility

- Open nature of XSLT allows for attributes to be safely added to XSLT elements from foreign namespaces; processors are to ignore them if they don’t recognize them
- “**Foreign Namespace Attributes**” allow developers to extend XSLT with additional metadata without breaking the language
- Comparable to .NET Attributes that tag object methods with additional metadata
- Kafka Namespace URI
  - http://www.thoughtpost.com/2002/kafka

Before Extensibility

```
<xsl:template name="echoString" >
  <xsl:param name="inputString" />
      <xsl:value-of select="$inputString"/>
</xsl:template>
```

- As it exists here, this template does not give us enough information to expose it as a Web Service in a friendly manner

After Extensibility

```
<xsl:template name="echoString"
  k:type="xsd:string">
  <xsl:param name="inputString"
  k:type="xsd:string"/>
      <xsl:value-of select="$inputString"/>
</xsl:template>
```

- Metadata is provided for the parameters and return type of this operation, legal to XSLT and enough to generate WSDL

Example Attributes

- type
- operation
  - Equivalent to .NET WebMethod() attribute
- targetNamespace
- schemaNamespace
- documentation
- array

XSLT Transformed

- It is not enough to just xsl:include support for the framework
- Because XSLT is XML, it can be transformed using a stylesheet
- In order to make the XSLT function as a Web Service, it first must be transformed using the framework

XSLT Transformed

- “Source” XSLT is transformed
  - Named templates tagged as operations are identified
  - The framework generates a stylesheet that includes the original, plus SOAP message processing and an xsl:choose to trigger the appropriate template based on the payload

Transport Protocols

- .NET provides **IHttpHandler** interface, a better ISAPI
- “Listener” code written as an IHttpHandler implementation provides just enough code to trigger the transformation in response to an HTTP request
- Additional transports could be supported in other ways

Listener Logic

- An HTTP POST (or GET) can represent the request to the Web Service, but many toolkits support additional requests
  - Description via WSDL
    - ?wsdl query string on a GET can provide WSDL for the service
  - Documentation
    - Browse to Web Service, get smart documentation

WSDL As Output

- XSLT needs additional type information in order to generate useful WSDL
- Foreign namespace attributes provide enough type information to generate WSDL
- Listener maps ?wsdl requests to a WSDL-generating transformation

Example

- XSLT Endpoint
- .NET HttpHandler for Listener
- Supports WSDL Generation

What Is Difficult?

- Focus on named templates
- XSLT support for Section 5 encoding is difficult; it does better with document-style
- Multi-ref accessors
  - This is the kind of detail that a good framework would hide, so it involves expanding the payload references
- Designing for SOAP Header support

Summary

- XSLT can function as the code behind a Web Service
- .NET makes it easy to write the non-XSLT code needed to support this framework
- The extensible nature of XSLT allows this type of framework to be possible