Web Services Essentials

Prepared for: New York University, Information Technology Services
Thursday, October 17, 2002

Ethan Cerami
New York University,
Mount Sinai Medical School
Road Map

- **Part I: XML Foundations**
  - Web Services are built on XML
  - Quick Crash Course in XML

- **Part II: Introduction to Web Services**
  - What is a Web Service?
  - Impact / Architectural Overview

- **Part III: Web Service Protocols**
  - XML-RPC and SOAP
  - WSDL and UDDI
Part I: XML Foundations
What is XML?

- **XML**: eXtensible Markup Language
- Two quotes to get things started:
  - “XML, to a certain extent, is HTML done right.” - Simon St. Laurent
  - “XML is HTML on steroids.”
- Two Main XML Keywords:
  - Extensible: can be extended to lots of different applications.
  - Markup language: Language used to mark up or annotate data.
**XML v. HTML**

- The best way to understand XML is to contrast it with HTML.
- **XML is Extensible:**
  - HTML: used to create web pages; restricted set of tags, e.g. `<I>` Italics, `<B>` Bold, etc.
  - XML: you can create your own tags
- **Example:** Put a library catalog on the web.
  - HTML: You are stuck with regular HTML tags that describe formatting.
  - XML: You can create your own set of tags: TITLE, AUTHOR, DATE, PUBLISHER, etc.
All HTML tags are focused on “look and feel” issues.
HTML conveys the “look and feel” of your page.

As a human, it is easy to pick out the publisher.

But, how would a computer pick out the publisher?

Answer: XML
The document now has new XML tags. A Human and a computer can now easily extract the publisher data.
XML v. HTML

- Tag Sets:
  - HTML has a predefined set of tags
  - XML lets you create your own tags.

- General Purposes:
  - HTML focuses on "look and feel"
  - XML focuses on the structure of the data.

- XML is not meant to be a replacement for HTML. In fact, they are usually used together.
Origins of XML
Origins of XML

- XML is based on SGML: Standard Generalized Markup Language
- SGML:
  - Developed in the 1970s
  - Used by big organizations: IRS, IBM, Department of Defense
  - Focuses on content structure, not look and feel
  - Good for creating catalogs, manuals.
  - Very complex
Origins of XML

- **XML**: SGML-Lite: 20% of SGML's complexity, 80% of its capacity.
- **HTML** and **XML** are both based on SGML.
XML and the W3C

- XML is an official standard of the World Wide Web Consortium (W3C)
- The Official Version is 1.0
- Official information is available at:
  - http://www.w3.org/XML/
- The Official XML FAQ:
  - http://www.ucc.ie/xml/
- W3C sponsors many projects which seek to enhance and improve XML.
Basic XML Examples
Scalable Vector Graphics

- SVG is a new standard for creating 2D graphics and animations on the web.
- SVG documents are basically XML documents.
- The XML tags indicate the location of text, graphics, shapes, etc.
- The XML can also indicate animations and transition effects.
- Represents one of the easiest, most compelling ways to learn XML.
SVG Plug-In

- To try out SVG, you first need to download the Adobe SVG Plug-In.
- Just go to:
  - http://www.adobe.com/svg/
  - Download the Plug-In for your Web Browser.
SVG Example #1

<?xml version="1.0" encoding="iso-8859-1"?>
<svg width="200" height="200"
 viewBox="0 0 200 200">
<rect x="10" y="20" width="180" height="100"
 fill="#eeeeff" stroke="red" stroke-width="1" />
<text x="24" y="75" font-family="sans-serif"
 font-size="20pt" fill="blue">Hello, World!</text>
</svg>

Root <svg> element: Create a 200x200 Box

<text> element: Create text at specified location.

<rect> element: Create a rectangle at the specified location.
<?xml version="1.0" encoding="iso-8859-1"?>
<svg width="200" height="200" viewBox="0 0 200 200">
  <rect x="10" y="20" width="180" height="100" fill="#eeeeff" stroke="red" stroke-width="1"/>
  <text id="hello" x="24" y="75" font-family="sans-serif" font-size="20pt" fill="blue">Hello, World!</text>
  <animate xlink:href="#hello" attributeName="dx" values="-15;15;-15" dur="2s" begin="0s" repeatDur="indefinite"/>
</svg>

<animate> element: animate the hello text, over a 2 second interval.
SVG Example #3

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<svg width="700" height="200" viewBox="0 0 700 200">
  <text id="hello" x="5" y="150" font-family="sans-serif"
       font-size="1pt" fill="blue">
    Hello, World!
  </text>
  <animate xlink:href="#hello" attributeName="font-size"
          values="1;196;1"
          dur="3s"
          begin="0s"
          repeatDur="indefinite"/>
</svg>
```

<animate>
  element: animate the font size over a 3 second interval.
Document Type Definitions and XML Schemas
DTDs and Schemas

- **Document Type Definition**: A set of rules for constructing an XML Document.

- **XML Schemas**: Newer version of DTDs that provides some additional features.

- **Valid**: Documents that conform to a DTD or a Schema are said to be valid.
Why are DTDs/Schemas Important?

- In order to communicate, computers need to agree to specific rules.
- By agreeing to XML standards, lots of people/computers/companies/industries can share data.
- Once we can share data, we can create more powerful applications.
Example:  Content Syndication

- What is Meerkat?
  - “Open Wire Service” that collects news from multiple web sites.
  - Created by OReilly.com
- Uses an XML Document Type Definition, called RSS (Rich Site Summary)
- For Meerkat to work, all participating sites must adhere to the RSS DTD.
Example Continued

- RSS Encapsulates specific article information, such as: Article Title, Article Description, Date, etc.
- By agreeing to follow the RSS DTD, hundreds of web sites can easily share data.
Content Syndication

Web Browser → Meerkat → Linux Today
Meerkat → News.com
Meerkat → Security Focus

XML
Challenges

- In order to be useful, DTDs/Schemas need to be widely used.
- Requires that companies/organizations collaborate on the creation of the rules.
- Results can be slow and fractured.
- May result in competing standards.
- Nonetheless, many more XML DTDs/Schemas are likely to emerge within the very near future.
Questions?
Part II: Introduction to Web Services
Introduction

Today, the principal use of the World Wide Web is for interactive access to documents and applications. In almost all cases, such access is by human users, typically working through web browsers…

The Web can grow significantly in power and scope if it is extended to support communication between applications, from one program to another.

-- From the W3C XML Protocol Working Group Charter
What is a Web Service?
What is a Web Service?

- A Web Service is any service that:
  - Is available over the Internet or private (intranet) networks
  - Uses a standardized XML messaging system
  - Is not tied to any one operating system or programming language
A Basic Web Service

Computer A:
Language: Perl
Operating System: Windows 2000

Computer B:
Language: Java
Operating System: Linux
XML Messaging

- There are several alternatives for XML messaging:
  - XML Remote Procedure Calls (XML-RPC)
  - SOAP
  - Regular XML transported over HTTP
- Any of these options are valid.
Web Services Defined

- Although not required, a web service may also have two additional (and desirable) properties:
  - a web service should be *self describing*.
  - a web service should be *discoverable*. 
Web Services: \textit{Self Describing}

- If you publish a new web service, you should also publish a public interface to the service.
- At a minimum, you should include human-readable documentation so that others can easily integrate your service.
- If you have created a SOAP service, you should also include a public interface written in a common XML grammar.
Web Services: Discoverable

- If you create a web service, there should be a relatively simple mechanism to publish this fact.
- Likewise, interested parties should be able to easily discover your service.
- The discovery service could be completely decentralized or completely centralized.
Web Services: Summary

- To summarize, a complete web service is any service that:
  - Is available over the Internet or private (intranet) networks
  - Uses a standardized XML messaging system
  - Is not tied to any one operating system or programming language
  - Is self-describing via a common XML grammar
  - Is discoverable via a simple find mechanism
The Impact of Web Services
To understand the impact of web services, consider basic e-commerce functionality.

For example, Widgets Inc. sells parts through its web site.
- Customers can purchase parts and check on order status.
The Web Today: Human Centric

- This illustrates a *human-centric web*, where humans are the primary actors initiating web requests.
- It also represents the primary model on which most of the web operates today.
Web Services: Application-Centric Web

- With web services, we move from a human-centric web to an application-centric web.
- In other words, conversations between applications occur as easily as conversations between web browsers and servers.
The Application-Centric Web

- The order status is now a web service.
- Applications can therefore connect to the order status service directly.
Application-Centric Web

- There are numerous areas where an application-centric web would be extremely helpful:
  - credit card verification
  - package tracking
  - shopping bots
  - single sign on registration
  - calendar, email, etc.
The Semantic Web

- Tim Berners Lee, the original inventor of the WWW, is now arguing for a **Semantic Web**.
- Berners Lee: The next stage is development of the Web is the Semantic Web.
- The Semantic Web vision is also application centric.
- Berners Lee has also argued that web services are in fact an actualization of the Semantic Web Vision.
Haven’t we seen this before?

- An application-centric web is not a new notion.
- For years, developers have created
  - CGI applications, Java Servlets and CORBA applications
  - designed primarily for use by other applications.
- The crucial difference is that most of these systems consisted of ad-hoc solutions.
- With web services, we have the promise of some standardization.
- Standardization will lower the barrier to application integration.
The Long-term Vision: Automated Web

- In the long-term, web services offer the promise of the automated Web.

- “Just-in-time” integration:
  - If services are easily discoverable, self-describing, and stick to common standards, it is possible to automate application integration.

- For example, consider a company, Mega Electric (ME) that wants to buy parts from Widgets, Inc.
  - ME wants to automatically integrate inventory with Widgets, Inc. order status service.
Just-In-Time Integration

1. Discover Services
2. Retrieve Service Description
3. Invoke Remote Service

ME “Smart” Inventory Application

Service Registry

Widgets Inc. Server
- Service Description
- Order Status Service
Hype v. Reality

- How close are we to creating “Just-in-time” integration?
- Currently, only some processes can be automated:
  – automatic registry query
  – automatic invocation of service
- However,
  – no mechanism exists for automating business relationships.
  – no mechanism exists for evaluating the quality of services.
Industry Landscape

- Many companies are investing heavily in web services.
- Currently many competing frameworks for building web services.
- Three main contenders:
  - Microsoft .NET
  - IBM Web Services
  - Sun Open Net Environment (ONE), Java
- All frameworks share a commitment to the same web services standards.
The World Wide Web Consortium is heavily involved in standardizing web services.
- SOAP is the first candidate for standardization.
- http://www.w3c.org/2002/ws/

Web Services Interoperability Organization (WSI)
- Industry organization
- Designed to speed up the standardization/compliance process.
- http://www.ws-i.org/
Web Services Architecture
Web Service Architecture

- There are two ways to view the web service architectural framework:
  1) Examine individual roles of each web service actor
  2) Examine the emerging web service protocol stack.
Web Service Roles

- Three major roles in web services:
  - Service Provider: provider of the web service.
  - Service Requestor: any consumer of the web service.
  - Service Registry: logically centralized directory of services.
Web Service Roles

1. Discover Services
   - Service Registry
   - Service Requestor

2. Invoke Service
   - Service Provider
Web Service Protocol Stack

- **Service transport**: responsible for transporting messages.  
  Examples: HTTP, BEEP

- **XML messaging**: responsible for encoding messages in common XML format.  
  Examples: XML-RPC, SOAP

- **Service Description**: responsible for describing an interface to a specific web service.  
  Example: WSDL

- **Service Discovery**: responsible for centralizing services into a common search registry.  
  Example: UDDI.
# Web Service Protocol Stack

<table>
<thead>
<tr>
<th>Discovery</th>
<th>UDDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>WSDL</td>
</tr>
<tr>
<td>XML Messaging</td>
<td>XML-RPC, SOAP, Custom XML</td>
</tr>
<tr>
<td>Transport</td>
<td>HTTP, SMTP, FTP, BEEP</td>
</tr>
</tbody>
</table>
Questions?
Part III: Web Service Protocols
XML Messaging
Option 1: XML-RPC

- XML-RPC: protocol that uses XML messages to perform Remote Procedure Calls (RPC.)
- Platform independent; diverse applications can talk to each other.
- XML-RPC is the easiest way to get started with web services.
  - Simpler than SOAP
  - Simpler data structures for transmitting data.
XML-RPC Example

- Here is a sample XML-RPC request to a weather service:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
    <methodName>weather.getWeather</methodName>
    <params>
        <param><value>10016</value></param>
    </params>
</methodCall>
```

“Give me the current weather conditions in zip code: 10016.”
XML-RPC Example

- Here is a sample Weather response:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
  <params>
    <param>
      <value><int>65</int></value>
    </param>
  </params>
</methodResponse>
```

“Current temperature is 65 degrees”
Option 2: SOAP

- SOAP: used to stand for “Simple Object Access Protocol”
- XML-Based protocol for exchanging information between computers.
- Currently being standardized by the World Wide Web Consortium (W3C.)
SOAP Message Format

SOAP Message

Envelope (Required)

Header (Optional)

Body (Required)

Fault (Optional: Response only)
Here is a sample SOAP request to a weather service:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
   <ns1:getWeather
     xmlns:ns1="urn:examples:weatherservice"
     SOAP-ENV:encodingStyle="http://www.w3.org/2001/09/soap-encoding/">
     <zipcode xsi:type="xsd:string">10016</zipcode>
   </ns1:getWeather>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP Example:

- Here is a sample SOAP response:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getWeatherResponse
      xmlns:ns1="urn:examples:weatherservice"
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
WSDL

- **WSDL**: Web Service Description Language.
- **WSDL** is an XML grammar for specifying an interface for a web service.
- **Specifies**
  - location of web service
  - methods that are available by the web service
  - data type information for all XML messages
- **WSDL** is commonly used to describe SOAP services.
**WSDL In a Nutshell**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;definitions&gt;</code></td>
<td>Root WSDL Element</td>
</tr>
<tr>
<td><code>&lt;types&gt;</code></td>
<td>What data types will be transmitted?</td>
</tr>
<tr>
<td><code>&lt;message&gt;</code></td>
<td>What messages will be transmitted?</td>
</tr>
<tr>
<td><code>&lt;portType&gt;</code></td>
<td>What operations (functions) will be supported?</td>
</tr>
<tr>
<td><code>&lt;binding&gt;</code></td>
<td>What SOAP specific details are there?</td>
</tr>
<tr>
<td><code>&lt;service&gt;</code></td>
<td>Where is the service located?</td>
</tr>
</tbody>
</table>
WSDL Excerpt: Weather Service

```xml
<message name="getWeatherRequest">
  <part name="zipcode" type="xsd:string"/>
</message>

<message name="getWeatherResponse">
  <part name="temperature" type="xsd:int"/>
</message>

<portType name="Weather_PortType">
  <operation name="getWeather">
    <input message="tns:getWeatherRequest"/>
    <output message="tns:getWeatherResponse"/>
  </operation>
</portType>
```
...<service name="Weather_Service">
  <documentation>WSDL File for Weather Service</documentation>
  <port binding="tns:Weather_Binding"
       name="Weather_Port">
    <soap:address
       location="http://ecerami.com/soap/servlet/rpcrouter"/>
  </port>
</service>
</definitions>
So What?

- Given a WSDL file, a developer can immediately figure out how to connect to the web service.
- Eases overall integration process.
- Better yet, with WSDL tools, you can *automate* the integration…
WSDL Invocation Tools

- For example, IBM has a tool called Web Services Invocation Framework (WSIF).
- Given a WSDL file, the tool will automatically invoke the service.
- A number of other tools from Microsoft, GLUE, and open source projects also do this.
WSIF: Example

- Invoke WSIF with the Weather WSDL File; specify method name and zip code parameter:

```java
clients.DynamicInvoker
http://localhost:8080/wSDL/WeatherService.wSDL getWeather 10016
```

- generates the following output:

```plaintext
Reading WSDL document from
'http://localhost:8080/wSDL/WeatherService.wSDL'
Preparing WSIF dynamic invocation
Executing operation getWeather
Result:
temperature=65
Done!
```
UDDI

- UDDI: Universal Description, Discovery and Integration.
- Currently represents the discovery layer in the protocol stack.
- Originally created by Microsoft, IBM and Ariba.
- Technical specification for publishing and finding businesses and web services.
UDDI: Two Parts

- **Part I: Technical specification**
  - specification for building a distributed directory of businesses and services.
  - XML format for specifying businesses and services.
  - API for querying/publishing to the registry.

- **Part II: Implementation**
  - UDDI Business Registry, fully operational implementation of the specification.
  - Businesses can publish services here.
  - Businesses can discover services here.
  - Currently maintained by IBM, Microsoft, etc.
UDDI Data

- **White Pages**
  - Information about a specific company; name description, address, etc.

- **Yellow Pages**
  - Classification data for company or service.
  - For example: industry, product or geographic codes.

- **Green Pages**
  - Technical information about specific services.
  - Pointers to WSDL Files.
All Together Now!
Using the Protocols Together

Step 1: Find Services via UDDI

Step 2: Retrieve Service Description File: WSDL or XML-RPC Instructions

Step 3: Create XML-RPC or SOAP Client

Step 4: Invoke Remote Service
Thanks for having me here today!

Questions?
Sources

- Information available at: http://www.oreilly.com/catalog/webservess/
Permission to Copy

Copyright © Ethan Cerami. All rights reserved.

This digital document may be reproduced and distributed to others provided that the above copyright notice and this paragraph are included in all such copies. This content itself may not be modified in any way. The limited permissions granted above are perpetual and will not be revoked by the author or his successors or assigns.
hell02.svg
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN" "http://www.w3.org/TR/2000/C3-SVG-20001102/DTD/svg-20001102.dtd">
<svg width="200" height="200" viewBox="0 0 200 200" xml:space="preserve">
  <title>Hello, World!</title>
  <desc>Hello, World SVG</desc>
  <rect x="10" y="20" width="180" height="100" fill="#eeeeff" stroke="red" stroke-width="1"/>
  <text id="hello" x="24" y="75" font-family="sans-serif" font-size="20pt" fill="blue">Hello, World!</text>
  <animate xlink:href="#hello" attributeName="dx" values="-15;15;-15" dur="2s" begin="0s" repeatDur="indefinite"/>
</svg>
hello4.svg
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN" 
<svg xmlns="http://www.w3.org/2000/svg" width="700" height="500" viewBox="0 0 700 500" xml:space="preserve">
<title>Hello, World!</title>
desc>text transformations, including changing the font size</desc>
<g transform="translate(350,300)">
text id="all" x="200" y="50" text-anchor="middle">
tspan id="s" font-size="36" fill="red">Hello</tspan>
tspan id="v" font-size="36" fill="green">World</tspan>
tspan id="g" font-size="28" fill="blue">!</tspan>
</text>
<animateTransform xlink:href="#all" attributeName="transform" type="rotate" values="0;360" dur="8s" begin="0s" repeatCount="indefinite"/>
<animate xlink:href="#s" attributeName="rotate" values="45;-45;45" dur="2s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#s" attributeName="font-size" values="36;98;36" dur="1.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="rotate" values="-10;35;-10" dur="2s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="font-size" values="36;200;36" dur="3s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="rotate" values="20;-40;20" dur="1.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="font-size" values="28;72;28" dur="2.5s" begin="0s" repeatDur="indefinite"/> 
</svg>

Page 1
Hello, World!

Text transformations, including changing the font size.

<svg width="500" height="500" viewBox="0 0 500 500" xml:space="preserve">
<title>Hello, World!</title>
<desc>text transformations, including changing the font size</desc>

<g transform="translate(100,100)">
<text id="all" x="200" y="50" text-anchor="middle">
<tspan id="s" font-size="36" fill="red">Hello</tspan>
<tspan id="v" font-size="78" font-family="Courier" fill="green">World</tspan>
<tspan id="g" font-size="28" fill="blue">!</tspan>
</text>
</g>

<animateTransform xlink:href="#all" attributeName="transform" type="rotate" values="0;360" dur="8s" begin="0s" repeatCount="indefinite"/>

<animate xlink:href="#s" attributeName="rotate" values="45;-45;45" dur="2s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#s" attributeName="font-size" values="36;18;36" dur="1.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="dy" values="0;10;0" dur="1.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="dx" values="20;-20;20" dur="3.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="rotate" values="-10;35;-10" dur="2s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#v" attributeName="font-size" values="78;36;78" dur="3s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="dy" values="0;-15;0" dur="2.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="dx" values="-20;30;-20" dur="3s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="rotate" values="20;-40;20" dur="1.5s" begin="0s" repeatDur="indefinite"/>
<animate xlink:href="#g" attributeName="font-size" values="28;72;28" dur="2.5s" begin="0s" repeatDur="indefinite"/>
</svg>