Web 2.0, AJAX, XML, Web Services for Network Engineers

BRKAPP-1015
What Are We Going to Talk About?
Lots of Cool Technologies and Fancy Acronyms

- Evolution of web technologies
- Service Oriented Architectures and Web Services
  SOAP, REST, XML, WSDL, UDDI
- Web 2.0
  RSS/ATOM, AJAX, XHR, JSON
Apps Transition to SOA and Web 2.0

- **Web 1.0**
  - Siloed Applications
  - Making each app work on its own is challenging enough
  - Limited data sharing between applications
  - Challenges with Scalability, Security and Control

- **Web 2.0 and SOA**
  - Collaborative personalized User Experience
  - Inherently Internet/Web Services based
  - Dynamic Content, Rich Media

You Said Service-Oriented Architecture?

- **Service Oriented Architecture: nothing new!**
  - RPC, DCE, DCOM, CORBA, JMS, RMI

- **Enabler for distributed applications**

- **Attempt to facilitate exchange of information between apps, or between producers and consumers**

- **Goal of SOA:**
  - Modularize business functions and enable software reuse

- **The ultimate goal of SOA**
  - Reduce dependence of business users on IT to launch new plans, products, or services – decouple business logic from implementation
Example:
Cell Phone Provisioning Application

The typical application is self-contained – it works with its own database, its own business modules, its server OS and components.

Goal behind SOA/Web Services: decouple the business modules and breaking them down into services that consumers can call.

Problems with the Traditional Model

- Tied to the underlying architecture
- Binary or proprietary message formats
  
  No secret decoder ring, no joy!
- Compatibility issues
- Expensive
- Not extensible
- Complex
- Performance costs
- Not firewall friendly

Eg. Does your firewall support CORBA?
Solutions to the Problem

- Open standard-driven mechanisms that are both programming language and platform independent.
- Make sure these standards are created by an independent standards body to minimize compatibility and patent issues.
- Make these standards robust enough to handle the job and yet simple enough to facilitate widespread adoption.

XML lends itself very well to this approach
Why XML Web Services?

- XML is plain ASCII
  - Contrast with binary messaging
- XML messaging rides on top of existing protocols
  - No need to reinvent the wheel
- XML over HTTP solves the problem of distributed applications across firewalls
- Guess what the ‘Web’ in Web Services is for?
  - Communications can run over HTTP
  - SOAP is essentially XML over HTTP

XML in 10 seconds

- HTML = a set of tags to format web pages (e.g., bold <b>, tables <td><tr>, colors <font>, etc.) – entirely focused on formatting rather than data
- XML = focuses on content rather than format. XML does not have any predefined tags. No such thing as <b>, <h1> etc.
An extreme XML Example

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <qawrtawtaw>
    <vfs dkj="wfwd">svsvsv</vfs>
    <fdsfd>djdhjdjh</fdsfd>
  </qawrtawtaw>
</html>
```

- This document is considered well-formed XML!
- XML does not have a pre-defined set of tags for user data, but documents must obey some syntax rules (prologue, proper nesting, etc.)
- How can we give XML meaning?

Giving XML meaning: XML Schemas

- Schemas describe the structure of an XML document
  - Defines data types, whether an element can be empty, default values, etc.
- Popular ways to define schemas include Document Type Definition (DTD) or W3C XML Schema
- W3C XML Schema far more prevalent for data-oriented style documents (e.g. restricting content, explicit data types)—it’s safe to forget DTDs
- Schemas provides a very convenient way to inform clients about the data types and ranges accepted by exposed services
Example: Complex XML Elements

```xml
<types>
  <xsd:schema targetNamespace="http://foo.cisco.com"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="product">
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="description" type="xsd:string"/>
        <xsd:element name="price" type="xsd:double"/>
        <xsd:element name="SKU" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="us-zipcode">
      <xsd:restriction base="xs:string">
        <xsd:pattern value="[0-9]{5}(-[0-9]{4})?"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
</types>
```

Exchanging Data in a WS World: SOAP

- Simple Object Access Protocol
- XML-based messaging format
- Rides on top of HTTP (and other protocols such as SMTP)
- SOAP = XML over HTTP

**HTTP Headers for the request**
```plaintext```
Host: 172.16.80.187
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1;
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/GetLoanRates"
Content-Length: 200
```

**HTTP Body of the request**
```xml```
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:soap=">GetLoanRates/</soap-ENV:Envelope>
```xml```
What’s Inside SOAP?

- Required SOAP Body and Envelope
- Optional SOAP Header

Web Services in Action

Ok this all works fine. How do I publish the services I offer?
The Heart of Web Services: WSDL

- **Web Services Description Language**
- XML-based language for describing Web Services
  - where to find them
  - how to access them
  - what methods they accept
- **The API of the Web Service**
- Web Services frameworks (.NET, J2EE, PHP, Perl, Python, etc.) can generate WSDL using introspection
- Clients grab the WSDL file and generate code to call the service
- Google for `filetype:asmx inturl:WSDL site:com`

Consuming a WSDL File

- Web Services consumers (browsers, applications) can directly use information contained in WSDLs
- Example: (free) tools such as WSDigger or SOAPui, or the very fancy (not free) XMLSpy from Altova parse WSDLs and create sample requests for each method, sending and receiving XML data with the WS-enabled server
  
  **Takes out the “guesswork” that comes with parsing and interpreting an HTML web page**
Central Web Services Directory: UDDI

- Universal Description, Discovery and Integration
- A Yellow Pages type registry for Web Services
- Publish your business information and technical details of your Web Service
- Search for other Web Services and connect to them
- The Yellow Pages of Web Services
- Not exactly an overwhelming success on the Internet
  → Somewhat more successful inside the Enterprise

Web Services: the Big Picture
WS-Enabled Cell Phone Provisioning Application

→ Collaborative and loosely coupled environment
→ Role of the network: secure, offload, accelerate..

Web Services: Is It just XML and SOAP?

- A school of thought doesn’t feel SOAP or WS-xxxx bring any real value. Most of the people who subscribe to this thinking seem to like XML
- Enters REST for Representational State Transfer
  - URIs, HTTP, XML
  - No SOAP
- With REST, Everything Old is New Again
Who Likes RESTful Web Services?

So Which One Is Better: SOAP or REST?

- A bit like EIGRP vs OSPF, Diet or Regular, Rent or Buy
- WSDL can adapt to both styles
- For semi-anonymous Internet “fire and forget” web services, REST is quick and simple and it doesn’t exclude the structure provided XML
- For B2B, WS extensions such as WS-Security provide message-level signing and encryption as well as various authentication mechanisms
SOA: It’s Happening Today!

XML Usage Is Increasing
- “XML accounted for 15% of internet traffic in 2005. By 2008, it is expected to account for 50%” – 451 Group

Some Examples:
- Salesforce.com: reports on their blog that over 40% of all Salesforce.com traffic comes from their API
- Amazon: 140,000 registered developers. Information Week article reported 3rd party sellers generated 28% of Amazon’s Q2 unit sales, or $490 million
- eBay: Over 25,000 developers with 1,900 certified applications. A TechWeb story notes that during Q4CY05, eBay handled more than 8 billion Web service requests, up from less than 1 billion for the entire CY02

http://blog.programmableweb.com/?p=277

Web 2.0
What Is “Web 2.0”? 

- A really overhyped buzzword? 
- Actually, a collection of technologies: 
  - Rich Client Interfaces 
  - AJAX, widgets, gadgets, mashups, etc. 
  - Annotations, tagging, social bookmarking 
  - “Folksonomy”, del.icio.us, etc. 
  - RESTful Web Services 
  - www.programmableweb.com, Amazon S3, etc. 
  - Social networking, News feeds 
  - Blogs, RSS, Atom 

Web 2.0 

- Interactive web: “read/write web” 
- User participation a primary characteristic: think Wikipedia, YouTube 
- No longer HTTP daemons serving static pages – look at http://www.pageflakes.com for instance 
- The web as a platform: continuously renewed software and services 
- It’s essentially the good old Web with a bunch of new technologies, which we are going to look at 
  - JSON 
  - RSS/Atom 
  - AJAX 
  - Mashups
JSON

Javascript Object Notation (RFC4627)

- Goal: facilitate portable structured data interchange (alternative to XML)
- Not a markup language, not a document format
- Small set of formatting rules
- Derived from the object literals of JavaScript
- String, number, object, array, boolean, null
- JSON is minimal, text-based, lightweight, easy to parse
- Programming language independent – any language can send/receive JSON data
- Primary application: AJAX programming where it is very popular
Example: a JSON Object

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    "212 732-1234",
    "646 123-4567"
  ]
}
```

Example: JSON Objects and Numbers

JSON being a subset of JavaScript, JSON-formatted text is also syntactically legal JavaScript code.
Example: JSON Strings

JSON Example: Yahoo! Weather

http://news.yahoo.com/.../fetch_weather?q=Sunnyvale,CA,94087,USA

response

Light rain
Really Simple Syndication (RSS) Feeds

- Publishers make content available in a format people can subscribe to – usually RSS 2.0 or Atom (RFC4287; probably technically superior, but less common in practice)
- A news reader (or RSS reader, or News Aggregator) processes the subscription and displays the content in a way and time convenient to the subscriber
- Most modern browsers can directly read RSS feeds
RSS Feeds: How Does It Work?

RSS: The Feed Is XML
Asynchronous JavaScript and XML

- Web development technique used for creating interactive web applications
- Allows client to exchange data with server without reloading the entire page
- Clients can update individual sections of web page with data returned by server (asynchronous)
- JavaScript is the client-side language in which AJAX calls are usually made
- Data is retrieved using the XMLHttpRequest object that is available to scripting languages
- Despite its name, XML is actually optional in AJAX
“JSON is the X in AJAX”

Douglas Crockford (author of the JSON RFC)  
Yahoo!

Basic AJAX Components

- JavaScript enabled on the client-side
- Pretty much any programming language server-side
  - The server script often makes web services and/or DB calls
- Typical use cases
  - AutoSuggest
  - Drag and Drop objects functionality
  - Dynamically move around an image or a map
  - Preload content that will be shown later
- Can use XML, JSON, HTML, plain text
A Trivial AJAX Example: A Self-Refreshing Clock

- Actual source code on the server: `<?php echo(DATE_RFC822);?>`

AJAX: It’s All Happening on the Client

```html
<html>
<head>
  <script language="javascript" type="text/javascript">
    var http = new XMLHttpRequest();
    var url = "ajax-script.php";

    function makeRequest()
    {
      http.open("GET", url, true);
      http.onreadystatechange = handleHttpRequest;
      http.send(null);
    }

    function handleHttpRequest()
    {
      if (http.readyState == 4)
      {
        if (http.status == 200)
        {
          result = http.responseText.split(",");
          document.getElementById('clock').innerHTML = result[1];
        }
        else
        {
          alert("There was a problem with the request.");
        }
      }
    }

    function showTime()
    {
      makeRequest();
      setTimeout("showTime()", 1000);
    }

  </script>
</head>
<body onload="showTime()">
  <span id="clock"></span>
</body>
</html>
```
What is A Mashup?

- A web application that combines data from more than one source into a single integrated tool
  - Typical example: plotting markers on a map using cartographic data from Google Maps to show location of items listed on Ebay or Craigslist
- Exemplifies the “web is the platform” in Web 2.0
  - The data sources and applications can be anywhere
- Several tools help users create fancy mashups
  - Yahoo! Pipes, Microsoft’s Popfly, Google Mashup Editor, …
- There are client-side and server-side mashups
Dude, Where’s My Used Car?

The Technology Behind Mashups

- Data sourced via RSS, Atom, WS or APIs
  
  See [http://programmableweb.com](http://programmableweb.com) for hundreds of data providers – typical APIs: Amazon.com, Google, Yahoo!

- Most of the time data returned as XML or JSON/YAML

- Almost always a mix of client-side and server-side processing:
  
  Server interacts with APIs from Google, Ebay, Amazon, etc
  
  Client builds page dynamically using bits and pieces – usually heavy Javascript is involved (libraries such as the DojoToolkit)
From the Network’s Perspective

Network View (Cont.)

- Calls to the Google map API:

```
```

What is the Google Maps API?

The Google Maps API lets you embed Google Maps in your own web pages with JavaScript. The API provides a number of utilities for manipulating maps (just like on the http://maps.google.com web page) and adding content to the map through a variety of services, allowing you to create robust maps applications on your website.

- The backend also makes calls to various Ebay APIs. When a search is performed, ZIP codes of vehicles are returned and queried against Google maps to display the little pins.
Web 2.0 Summarized

- Semantic markup – transition to XML
  (D|X)HTML for display; XML/JSON for data
  → RSS a good example
- Web Services – breaking away from static sites
  Data and Services easily accessible independently of underlying platform
- Mixing or remixing content
  RSS, mashups
  → Focus on content more than container
- Emergent navigation and relevance – users are in control
  AJAX, highly interactive web applications
- Adding metadata over time – communities building social information
  Users add their own metadata (tags) – see Flickr, del.icio.us, etc.
- Shift to programming – separation of structure and style
  Content accessible by programmatic means
  Web 2.0 designers now often real programmers

Summary “For the Network Engineer”

- Web 2.0 and Web services: it’s all HTTP!
- Although it’s HTTP, the payloads are new
  XML, JSON; APIs and Web Services allow data to be easily pulled
- Those payloads introduce new challenges for the applications
  What if the data doesn’t follow the XML Schema?
  What if it’s intercepted mid-flight?
  Latency sensitivity of highly interactive Web 2.0 apps?
  Web application attacks (injection, etc.)
- We the Network can help address these issues
- How? See BRKAPP-2014 ;) (ACE XML Gateway)
Q and A

Recommended Reading

- Continue your Cisco Live learning experience with further reading from Cisco Press
- Check the Recommended Reading flyer for suggested books

Available Onsite at the Cisco Company Store
Complete Your Online Session Evaluation

- Give us your feedback and you could win fabulous prizes. Winners announced daily.
- Receive 20 Passport points for each session evaluation you complete.
- Complete your session evaluation online now (open a browser through our wireless network to access our portal) or visit one of the Internet stations throughout the Convention Center.

Don't forget to activate your Cisco Live virtual account for access to all session material on-demand and return for our live virtual event in October 2008.

Go to the Collaboration Zone in World of Solutions or visit www.cisco-live.com.