

## IT452 Advanced Web and Internet Systems

<h3>Set 8: Web Services (Chapter 28)</h3>
---

### Web Services

- “any service available on the Web that has been designed for consumption by programs, independent of the technology being used”
- Two primary camps
  - REST (sometimes just “HTTP”)
    - Exchanging documents
    - Many HTTP actions
  - SOAP
    - Exchanging messages, or RPC
    - Mainly HTTP POST

## REST

- Use all of HTTP for sensible document management and caching:
  - POST – make new document
  - HEAD – get doc metadata
  - GET – retrieve document (no state change)
  - PUT – update document
  - DELETE – delete document
- Request – all in the URL
- Response format?
  
- In practice – often GET for everything
  - Works in browser
  - But violates “no side effects” rule

## SOAP

- Originally “Simple Object Access Protocol”
- Two views
  - 1. Exchanging messages
  - 2. Performing RPC
- Request
  - Mostly POST (but need not be just HTTP!)
  - A complex XML document
  - What parameters/functions are legal??
  
- Response format: XML

## (Ex 1) Weather XML Data

From: [http://xoap.weather.com/weather/local/98125?cc=\\* &prod=xoap&par=1050661447&key=0167ad7e1ac45a20&link=xoap](http://xoap.weather.com/weather/local/98125?cc=* &prod=xoap&par=1050661447&key=0167ad7e1ac45a20&link=xoap)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<weather ver="2.0">
  <head>
    <locale>en_US</locale>
    <form>MEDIUM</form>
    <ut>F</ut>
    <ud>mi</ud>
    <us>mph</us>
    <up>in</up>
    <ur>in</ur>
  </head>
  <loc id="98125">
    <dnam>Seattle, WA (98125)</dnam>
    <tm>8:21 PM</tm>
    <lat>47.72</lat>
    <lon>-122.30</lon>
    <sunr>7:43 AM</sunr>
    <sunss>6:03 PM</sunss>
    <zzone>-7</zzone>
  </loc>
  <cc>
    <lsup>10/16/08 8:00 PM PDT</lsup>
    <obst>Seattle Jackson Park, WA</obst>
    <tmp>47</tmp>
    <flik>45</flik>
    <t>Cloudy</t>
    <icon>26</icon>
    <bar>
      <r>30.34</r>
      <d>rising</d>
    </bar>
  </cc>
</weather>
```

## (Ex 1) weather.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head> <title>XSLT Example with Web services</title>
    <script type="text/javascript" src="transform.js"/>
  </head>
  <body>
    <p>Web 2.0 is great!</p>
    <p>Zip code: <input type="text" id="zipcode" />
      <input type="button" value="Get weather!" onclick="getWeather()" />
    </p>
    <h2>Below will be some content from elsewhere:</h2>
    <div id="planet">
      <h1>This is some boring content.</h1>
    </div>
    <p>
      <a href="http://www.cs.usna.edu/w3c-validator/check?uri=referer">
        
      </a>
    </p>
  </body>
</html>
```

## (Ex 1) transform.js

```
function getWeather() {
    var box = document.getElementById("zipcode");
    var zip = box.value;
    var url = "weather_proxy.cgi?zipcode=" + zip;
    transform("simple.xml", url);
}

function transform(xslFileName, url) {
    // ASIDE: Why won't this work?
    // xmlhttp.open("GET",
    // "http://xoap.weather.com/weather/local/21402?cc=*%prod=xoap&par=1050661447&key=0167ad7e1ae45a20", false);

    // Get the XML input data
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("GET", url, false);
    xmlhttp.send('');

    // Get the XSLT file
    var xslhttp = new XMLHttpRequest();
    xslhttp.open("GET", xslFileName, false);
    xslhttp.send('');

    // Transform the XML via the XSLT
    var processor = new XSLTProcessor();
    processor.importStylesheet(xslhttp.responseText);
    var newDocument = processor.transformToDocument(xmlhttp.responseText);

    // Replace part of original document with the new content
    var o = document.getElementById("planet");
    var n = newDocument.getElementById("planet");
    o.parentNode.replaceChild(n, o);
}
```

## (Ex 1) weather\_proxy.cgi

```
#!/usr/local/bin/perl

use CGI ":standard";
use strict;

# Need this to get web pages from Perl
# these all work on chessie
use LWP::Simple;
use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;

# We want to send XML back
print "Content-type: text/xml\n\n";

# Construct URL to get the weather
my $zip = param("zipcode");
my $URL =
    "http://xoap.weather.com/weather/local/$zip?cc=*%prod=xoap&par=1050661447&key=0167ad7e1ae45a20&link=xoap";

# Get the XML document and send it back to requestor (the browser)
my $contents = get($URL);
print $contents;
```

## (Ex 1) simple.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" >
  <xsl:template match="/">
    <html>
      <head>
        <title> Current weather in: <xsl:value-of select="/weather/loc/dnam"/> </title>
      </head>
      <body> <div id="planet">
        <h1>Current weather in:
          <xsl:value-of select="/weather/loc/dnam"/>
        </h1>
        <xsl:apply-templates select="/weather/cc"/>
      </div> </body>
    </html>
  </xsl:template>

  <!-- Handle current conditions -->
  <xsl:template match="cc">
    <ul>
      <li>Status: <xsl:value-of select="t" />
        <xsl:apply-templates select="icon" /> </li>
      <li>Temperature: <xsl:value-of select="tmp" /> </li>
      <li>Observed from: <xsl:value-of select="obst" /> </li>
    </ul>
  </xsl:template>

  <!-- deal with weather icon -->
  <xsl:template match="icon">
    
  </xsl:template>
</xsl:stylesheet>
```

## (Ex 2) Photos XML Data

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <s:Body>
    <x:FlickrResponse xmlns:x="urn:flickr">

      <photos page="1" pages="20668" perpage="5" total="103339">
        <photo id="2944625312" owner="41086422@N00" secret="1975114cb7" server="3057" farm="4" title="Bad Mascot"
          ispublic="1" isfriend="0" isfamily="0" />
        <photo id="2944368362" owner="29542413@N07" secret="0f3f076cd1" server="3020" farm="4" title="_MG_3447"
          ispublic="1" isfriend="0" isfamily="0" />
        <photo id="2943510303" owner="29542413@N07" secret="7c04e22d9b" server="3283" farm="4" title="_MG_3462"
          ispublic="1" isfriend="0" isfamily="0" />
        <photo id="2944369890" owner="29542413@N07" secret="fe9271a3b0" server="3035" farm="4" title="_MG_3454"
          ispublic="1" isfriend="0" isfamily="0" />
        <photo id="2944370484" owner="29542413@N07" secret="451a349bb0" server="3184" farm="4" title="_MG_3456"
          ispublic="1" isfriend="0" isfamily="0" />
      </photos>
    </x:FlickrResponse>
  </s:Body>
</s:Envelope>
```

## (Ex 2) photos.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head <title>XSLT Example with Web services</title>
    <script type="text/javascript" src="transform2.js"/>
  </head>
  <body>
    <p>Web 2.0 is great!</p>
    <p>Tags to search for: <input type="text" id="tags" />
      <input type="button" value="Get photos!" onClick="getPhotos()" />
    </p>

    <h2>Below will be some content from elsewhere:</h2>

    <div id="planet"> <h1>This is some boring content.</h1> </div>

    <p>
    <a href="http://www.cs.usna.edu/w3c-validator/check?uri=referer">
      
    </a>
    </p>
  </body>
</html>
```

## (Ex 2) Sample SOAP request

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
  xmlns:xsd='http://www.w3.org/1999/XMLSchema'
>
  <s:Body>
    <x:FlickrRequest xmlns:x='urn:flickr'>
      <method>flickr.photos.search</method>
      <name>value</name>
      <tags>tigers</tags>
      <privacy_filter>1</privacy_filter>

      <per_page>5</per_page>
      <api_key>83ec7bab1628defd47d893288348fee5</api_key>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

## (Ex 2) flickr\_proxy.cgi (part 1)

```
#!/usr/local/bin/perl
use CGI "standard";

# Need this to get web pages from Perl
use LWP::Simple;
use HTTP::Request;
use LWP::UserAgent;

# We want to send XML back
print "Content-type: text/xml\n\n";

my $ua = LWP::UserAgent->new();
my $method = "POST";
my $url = "http://api.flickr.com/services/soap/";

my $tags = param("tags");

my $content = "
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
  xmlns:xsd='http://www.w3.org/1999/XMLSchema'
>
  <s:Body>
    <x:FlickrRequest xmlns:x='urn:flickr'>
      <method>flickr.photos.search</method>
      <name>value</name>
      <tags>$tags</tags>
      <privacy_filter>1</privacy_filter>
      <per_page>5</per_page>
      <api_key>83ec7bab1628defd47d893288348fee5</api_key>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
";
```

## (Ex 2) flickr\_proxy.cgi (part 2)

```
use HTTP::Headers;
my $header = HTTP::Headers->new();
my $request = HTTP::Request->new($method, $url, $header, $content);

use HTTP::Response;
my $response = $ua->request($request);
if($response->is_success) {
  # The Flickr response via SOAP is encoded -- won't be recognized right away as XML.
  # So we need to decode some of the thing like &lt; &quot; etc.
  my $the_response = $response->content;
  $the_response =~ s/&lt;/&lt;/eg; # convert &lt;
  $the_response =~ s/&gt;/&gt;/eg; # convert &gt;
  $the_response =~ s/&quot;/"/eg; # convert &quot;
  print $the_response;
}
else {
  print $response->error_as_HTML;
}
```

## (Ex 2) flickr.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:x="urn:flickr" >
  <xsl:template match="/">
    <html>
      <head>
        <title> Flickr test </title>
      </head>
      <body> <div id="planet">
        <!-- Define variable for total number of matches. @total gets the 'total' attribute of <photos> -->
        <xsl:variable name="var_total" select="/s:Envelope/s:Body/x:FlickrResponse/photos/@total" />

        <p>There were <xsl:value-of select="$var_total" /> results. Here are just some: </p>
        <ul> <xsl:apply-templates select="/s:Envelope/s:Body/x:FlickrResponse/photos/photo" /> </ul>
      </div> </body>
    </html>
  </xsl:template>
  <xsl:template match="photo"> <!-- Handle one photo -->
    <!-- create a variable for the image url
      A whole URL for this is like: http://farm1.static.flickr.com/2/1418878_le92283336_m.jpg -->
    <xsl:variable name="url">http://farm<xsl:value-of select="@farm" />.static.flickr.com/<xsl:value-of
      select="@server" />/<xsl:value-of select="@id" />_<xsl:value-of select="@secret" />_m.jpg
    </xsl:variable>

    <!-- show the actual image, with a link to it -->
    <p> <li> <a href="{ $url }">  </a> </li> </p>
  </xsl:template>
</xsl:stylesheet>
```

## (Ex 2) transform2.js

```
function getPhotos() {
  var box = document.getElementById("tags");
  var tags = box.value;
  var url = "flickr_proxy.cgi?tags=" + tags;
  transform("flickr.xsl", url);
}

function transform (xslFileName, url) {

  // Get the XML input data
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.open("GET", url, false);
  xmlhttp.send('');

  // Get the XSLT file
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.open("GET", xslFileName, false);
  xmlhttp.send('');

  // Transform the XML via the XSLT
  var processor = new XSLTProcessor();
  processor.importStylesheet(xmlhttp.responseText);
  var newDocument = processor.transformToDocument(xmlhttp.responseText);

  // Replace part of original document with the new content
  var o = document.getElementById("planet");
  var n = newDocument.getElementById("planet");
  o.parentNode.replaceChild(n, o);
}
```