

Web Services

INFO 344 Winter 2007

Administrivia

- No Lab Work Due From Now On
 - You are still expected to do the labs
- Writing Assignment 2 Due March 6
- Project 4 (Database Design) Due TODAY
- Final project due March 16 at 10:00am

Today

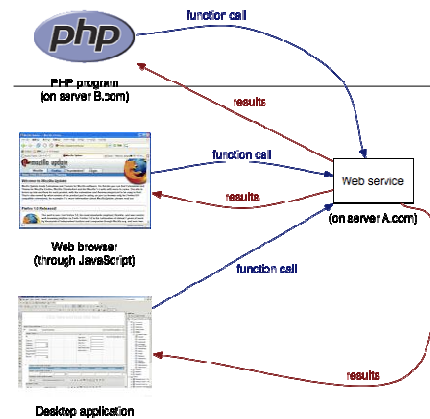
- Introduction to web services
- Producing and consuming web services

Web services

- A different model for programming on the Web
- Previously we've been programming using a browser-based model
 - PHP files produce *pages* using HTML
 - User navigates to site, views pages
- *Web services* decouple program from the presentation of pages
 - An API over the web

Web service APIs

- A set of functions that can be called remotely using HTTP
 - Used by other programs and programmers
 - Define functions and arguments
 - Return *data* rather than a Web page
- Can be used by any system that can support HTTP requests
 - Server-side applications (PHP, Java, etc.)
 - Client-side applications (e.g. Google Earth)
 - Directly from the browser with JavaScript



Why bother?

- Why would we call functions over the web rather than using them locally?
- Web services can tie in to *data sources* online to provide other programs access to data and functionality
 - Leverage data sources for other uses without having to share whole database

Google APIs

- Search API
 - Allows programmers to perform a Google search directly within their application
 - API returns a list of query results
- Maps API
 - Web service returns HTML code for maps
 - Can be combined with other services to create "mashups"

Unthirsty

<http://www.unthirsty.com/index.php?Address=seattle%2C+wa>



Yahoo! Web services

- Traffic alerts
 - Provides real-time data about street traffic
- Travel search
 - Search travel deals online
- Shopping
 - Fetch user reviews about a given product
- Geocoder API
 - Provide latitude and longitude data for US street addresses

Geocoder API

- Request latitude and longitude from a street address
- Request through HTTP query
 - <http://api.local.yahoo.com/MapsService/V1/geocode?appid=YahooDemo&street=701+First+Street&city=Sunnyvale&state=CA>
- Response as XML

Geocoder API response

```
<ResultSet xsi:schemaLocation="urn:yahoo:maps
http://api.local.yahoo.com/MapsService/V1/GeocodeResponse.xsd"
  >
  <Result precision="address" warning="The exact location
could not be found, here is the closest match: 701 First Ave,
Sunnyvale, 94089"
    >
    <Latitude>37.416384</Latitude>
    <Longitude>-122.024853</Longitude>
    <Address>701 FIRST AVE</Address>
    <City>SUNNYVALE</City>
    <State/>
    <Zip>94089-1019</Zip>
    <Country>US</Country>
  </Result>
</ResultSet>
```

Other web services

- Amazon: sale information, historical pricing
- eBay: Information about available auctions
- Flickr: Retrieve photos by keyword

Web services in PHP

- Producing web services
- Consuming web services

Web service standards

- A number of standards exist for providing XML-based web services
- SOAP: XML-based format for describing OO function calls over the Web
- XML-RPC: Derivative of SOAP, not object-oriented

SOAP

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:getPrice xmlns:ns1="urn:xmethods-BNPriceCheck"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<isbn xsi:type="xsd:string">0385503954</isbn>
</ns1:getPrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<wsdl:portType name="HelloWorld">
<wsdl:operation name="sayHello">
<wsdl:input
message="impl:sayHelloRequest"
name="sayHelloRequest"/>
<wsdl:output
message="impl:sayHelloResponse"
name="sayHelloResponse"/>
</wsdl:operation>
</wsdl:portType>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getPriceResponse xmlns:ns1="urn:xmethods-BNPriceCheck"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:float">14.65</return>
</ns1:getPriceResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Another Way ... REST

- Some argue that these complex XML formats are unnecessary
 - We already have methods for accessing web resources: HTTP GET and POST
 - Advantages: No need to struggle with complex XML
 - Disadvantages: Can be somewhat more informal, less "discoverable"
- REST (REpresentational State Transfer) web services use HTTP requests and plain text/XML for web services
 - HTTP requests to call functions
 - Return response as plain text or simple XML

Religious Warfare

- SOAP
 - Requires WSDL files to publish resources
 - XML in request and response
 - Developer needs to know the XML syntax for the service
 - Uses Remote Procedure Calls (RPCs) over HTTP
 - XML-wrapped RPC difficult to "sniff"
- REST
 - Uses a URI to locate objects
 - Passes method calls as GET parameters
 - Uses well-known calls (GET, POST, PUT, DELETE)
 - Uses plain HTTP
 - Calls can be secured by the firewall or via certificates

Different Designs

- SOAP
 - RPC applications with methods

```
exAppObj = new ExApp("example.com:1234")
exAppObj.getUser()
```
- REST
 - Resources defined as URI's

```
http://example.com/users/
http://example.com/users/{user}
http://example.com/findUserForm
http://example.com/locations/
http://example.com/locations/{location}
http://example.com/findLocationForm
```
 - ```
userResource = new
Resource("http://example.com/users/001")
userResource.delete()
```

## Benefits of SOAP

- Can include lots of complex data in the **request**
  - Such requests can be cumbersome or impossible over GET or POST

## Creating RESTful web services in PHP

- Web service code looks similar to MVC Controller
  - Retrieve parameters from request
  - Return a response
- Example web service: addTwo.php
  - Parameters: *a* and *b*
  - Returns: their sum as a plain text value

## addTwo.php

```
<?php
$a = $_REQUEST["a"];
$b = $_REQUEST["b"];

echo $a + $b;
?>
```

## Consuming web services

- We can also consume web services in PHP
- Use `file_get_contents` function to connect to a URL and retrieve text
  - Can be used to call parameterized web service functions
- Syntax: `$output = file_get_contents($url)`
  - Uses GET request
  - POST possible with additional libraries

## PHP client using addTwo

```
<?php
// use addTwo to add two numbers
$firstParam = 14;
$secondParam = -4;
$url =
"http://linux.ischool.washington.edu/~skane/addTwo.p
hp";
$url .= "?a=$firstParam";
$url .= "&b=$secondParam";
$result = file_get_contents($url);
?>
<html><head><title>Web service client</title></head>
<body>
<p><?php echo $firstParam; ?> plus <?php echo
$secondParam; ?> equals <?php echo $result; ?></p>
</body></html>
```

## Choosing an output format

- Plain text
  - + Simple, good for singular values
  - Can't easily handle objects, difficult to understand
- Simple XML files
  - + Standard format for passing data values
  - Can be cumbersome to manipulate in program
- HTML
  - + Can be inserted directly into page
  - Bad separation between code and presentation

## Using XML in web services

- When working with objects, web service can send response object as XML
  - Standardized format, can be read everywhere
  - Human-readable
- On web service end, write method to output object as XML
  - *Serialize* object as XML
  - Client can read object using SimpleXML

## XML serialization

```
<?php
class Response {
 var $first;
 var $second;
 var $operation;
 var $result;

 // constructor
 function Response($f, $s, $o, $r) {
 $this->first = $f;
 $this->second = $s;
 $this->operation = $o;
 $this->result = $r;
 }
}
?>
```

## Response toXML() method

```
function toXML() {
 $output = "<response>";
 $output .= "<first> $this->first</first>";
 $output .= "<second> $this->second</second>";
 $output .= "<operation> $this->operation</operation>";
 $output .= "<result> $this->result</result>";
 $output .= "</response>";
 return $output;
}
```

## Producing XML web services

```
<?php
include ("Response.php");
$a = $_REQUEST["a"];
$b = $_REQUEST["b"];

$r = new Response($a, $b, "plus", $a+$b);
echo $r->toXML();
?>
```

↙

```
<response>
<first>4</first>
<second>14</second>
<operation>plus</operation>
<result>18</result>
</response>
```

## Consuming XML web services

```
<?php
// use XML service to add two
$firstParam = 14;
$secondParam = -4;
$url =
"http://linux.ischool.washington.edu/~skane/addTwoXML.php";
$url .= "?a=$firstParam";
$url .= "&b=$secondParam";
$xml = file_get_contents($url);

// convert XML response to object
$response = new SimpleXMLElement($xml);
echo "The sum is " . $response->result;
?>
```