Web Application Development: Web Services and APIs

Week 9

Content

Using Others' Provided Functionality

- So far we have discussed building web-based systems where we have provided both the server-side and the client-side functionality
- Now we look at using services provided by others – web services, and APIs (writing of web services will be covered in the WAA unit)
- Thus we need to look at systems that involve three levels of provision: services on third-party servers, services on our own server, and user interfaces on the client
- Unfortunately, there is difference between using 3rd party services, and interacting with our own server - the XMLHttPRequest object naturally communicates with our own server, and there are security issues in letting 3rd party data invade our client machines. We need to explore some additional techniques to make it all happen

Web Services: W3C Definition

- *A Web service is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML.*
- *Its definition can be discovered by other software systems*
- *These systems may then interact with the Web service using XML based messages conveyed by Internet protocols*

Web Services

- Web services are an technology that offers a solution for providing a common collaborative architecture.
- Web services provide functional building blocks which are not tied to any particular programming language or hardware platform.
- They are accessible over standard Internet protocols.
- The main implementation technology underpinning a broader push to Service-Oriented Computing
- Think of web services as software components (even “objects”) that can be accessed and used in systems, and which will “execute” on the computer(s) on which they reside, when called upon via the API.

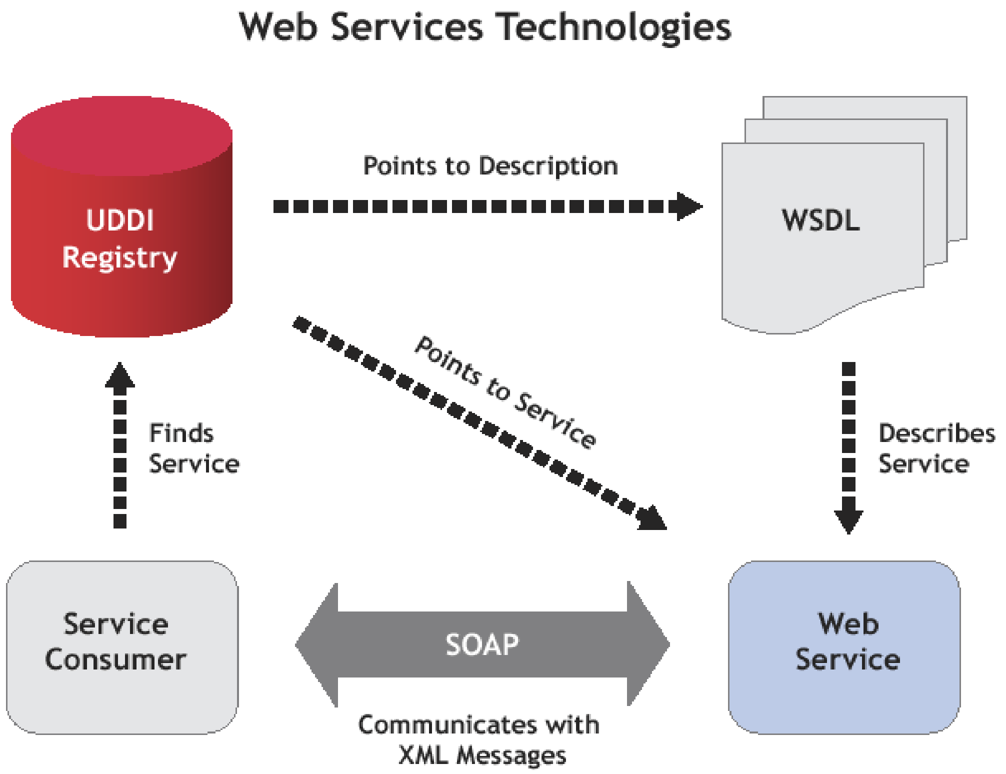
Technologies

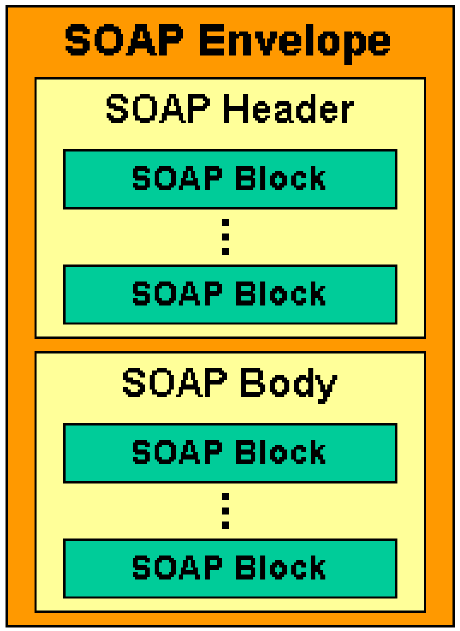
- **XML** (eXtensible Markup Language)
  - markup language that underlies most of the specifications used for Web services.
- **SOAP** (Simple Object Access Protocol)
  - A network, transport, and programming language-neutral protocol that allows a client to call a remote service. The message format is XML.
- **WSDL** (Web services description language)
  - An XML document that defines the programmatic interface –operations, methods, and parameters - for web services.
- **UDDI** (universal description, discovery, and integration)
  - Both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information on service providers and Web services.

How it works

- A Web Service is a URL-addressable software resource that performs functions (or a function).
- Web Services communicate using a standard protocol known as SOAP (Simple Object Access Protocol). (There is one major alternative approach – "REST" – that we will use in this unit. SOAP is studied in detail in WAA. So-called "RESTful" services are now popular – see, for example, http://java.sun.com/developer/technicalArticles/WebServices/restful/ http://www.ibm.com/developerworks/webservices/library/ws-restful/)
- A Web Service is located by its listing in a Universal Discovery, Description and Integration (UDDI) directory.

Technologies





Characteristics

- A Web Service is **accessible over the Web**.
- Web Services communicate using **platform-independent and language-neutral** Web protocols.
- A Web Service provides an **interface** that can be called from another program.
- A Web Service is **registered and can be located** through a Web Service Registry.
- Web Services support **loosely coupled** connections between systems.
- Web Services promote componentisation of common functions.
- Web services ease your server-side programming effort – mostly developers consume rather than create Web Services

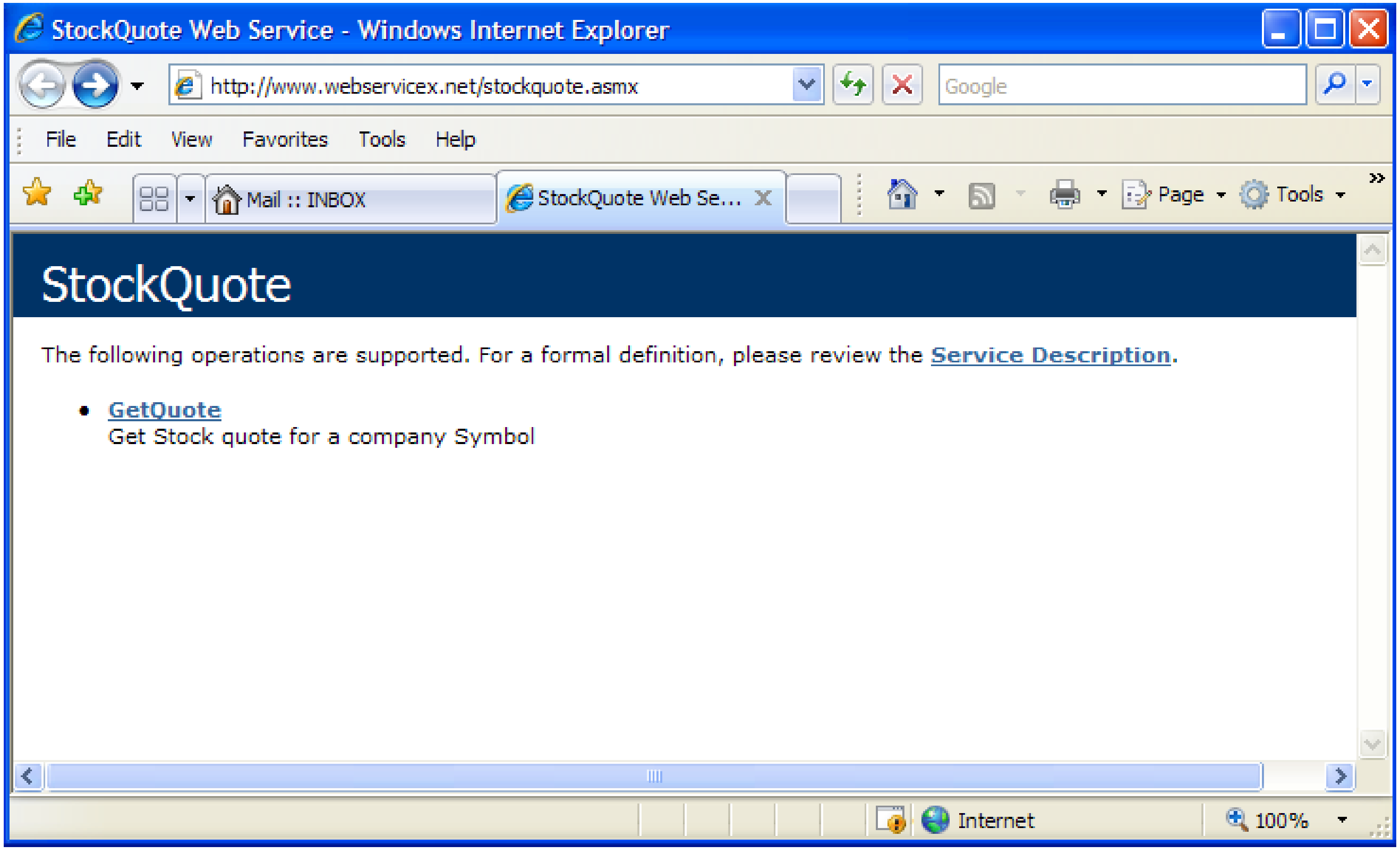
Public Web Services

- WebServiceX.Net (www.webservicex.net)
- Yahoo Web Services (http://developer.yahoo.com)
- Amazon Web Service (http://aws.amazon.com)
- Last.FM (www.audioscrobbler.net/data/webservices/)
- eBay (http://developer.ebay.com/developercenter/rest/)

Consuming a Web Service

- Type in the URL of a preexisting web service - service endpoint, say www.webservicex.net/stockquote.asmx
- A list of methods of the web service will be displayed. For the stockquote service, the single GetQuote method is displayed as in Figure 1
- Click the method, you will see Figure 2 that asks you to supply some parameters and display sample request/response to be returned by the web service
- If you provide the parameter “GOOG” for symbol and click Invoke, you will see what is displayed on Figure 3

Figure 1: StockQuote Web Service

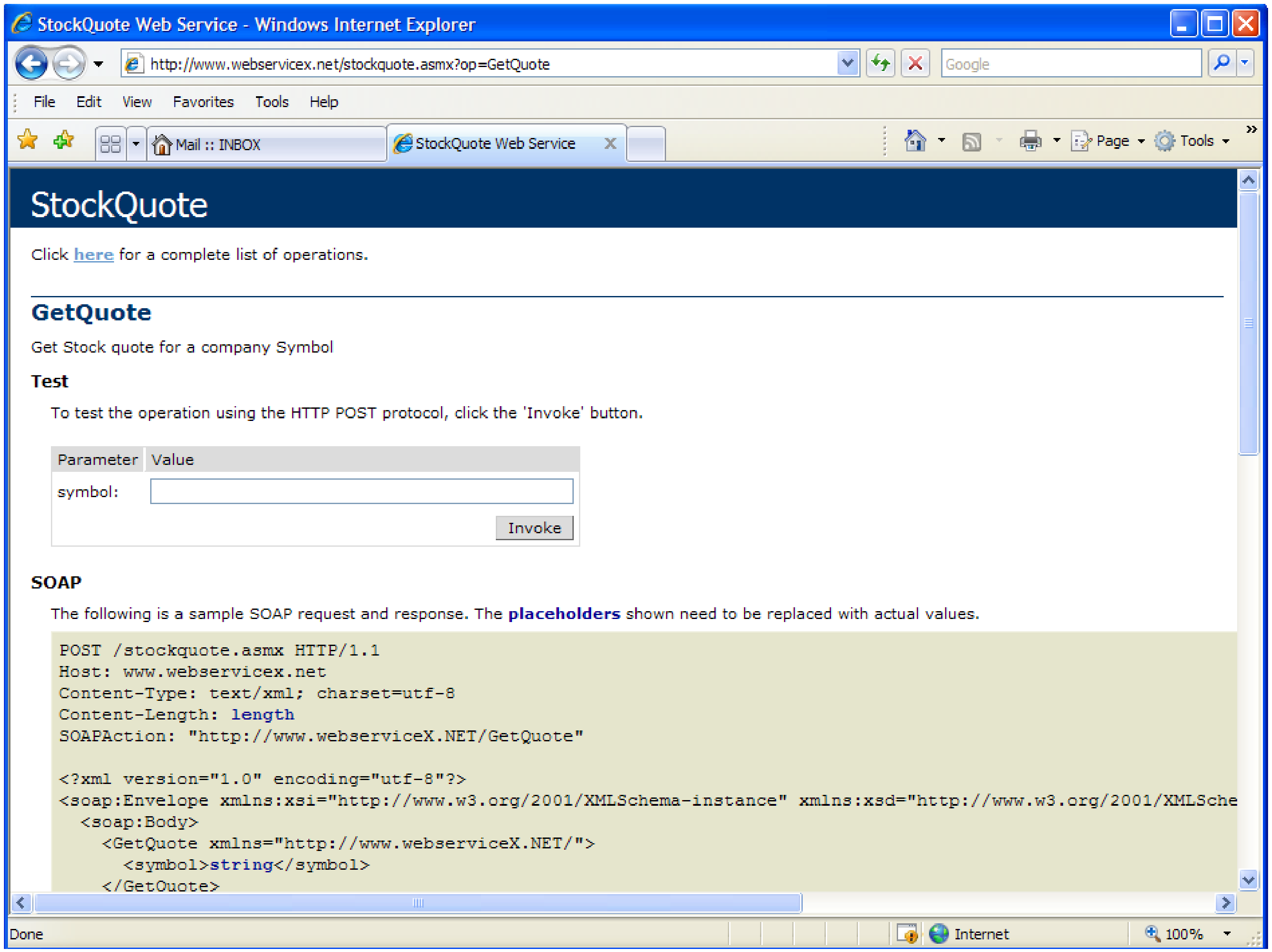

StockQuote Web Service - Windows Internet Explorer
http://www.webservicex.net/stockquote.asmx
Google
File
Edit
View
Favorites
Tools
Help
Mail :: INBOX
StockQuote Web Se...
Page
Tools
Internet
100%

StockQuote

The following operations are supported. For a formal definition, please review the Service Description.

- GetQuote
  Get Stock quote for a company Symbol

Figure 2: GetQuote Method and Its implementation


StockQuote Web Service - Windows Internet Explorer
http://www.webservicex.net/stockquote.asmx?op=GetQuote
Google
File Edit View Favorites Tools Help
Mail :: INBOX
StockQuote Web Service
Page Tools
Done
Internet
100%

StockQuote

Click here for a complete list of operations.

GetQuote

Get Stock quote for a company Symbol

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
symbol:	

Invoke

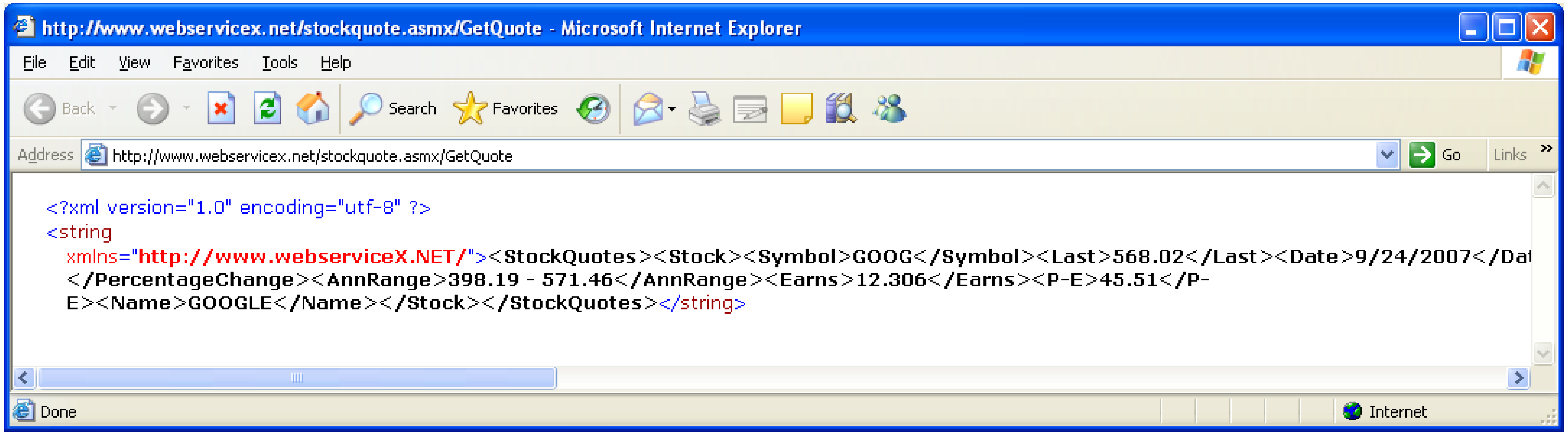
SOAP

The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with actual values.

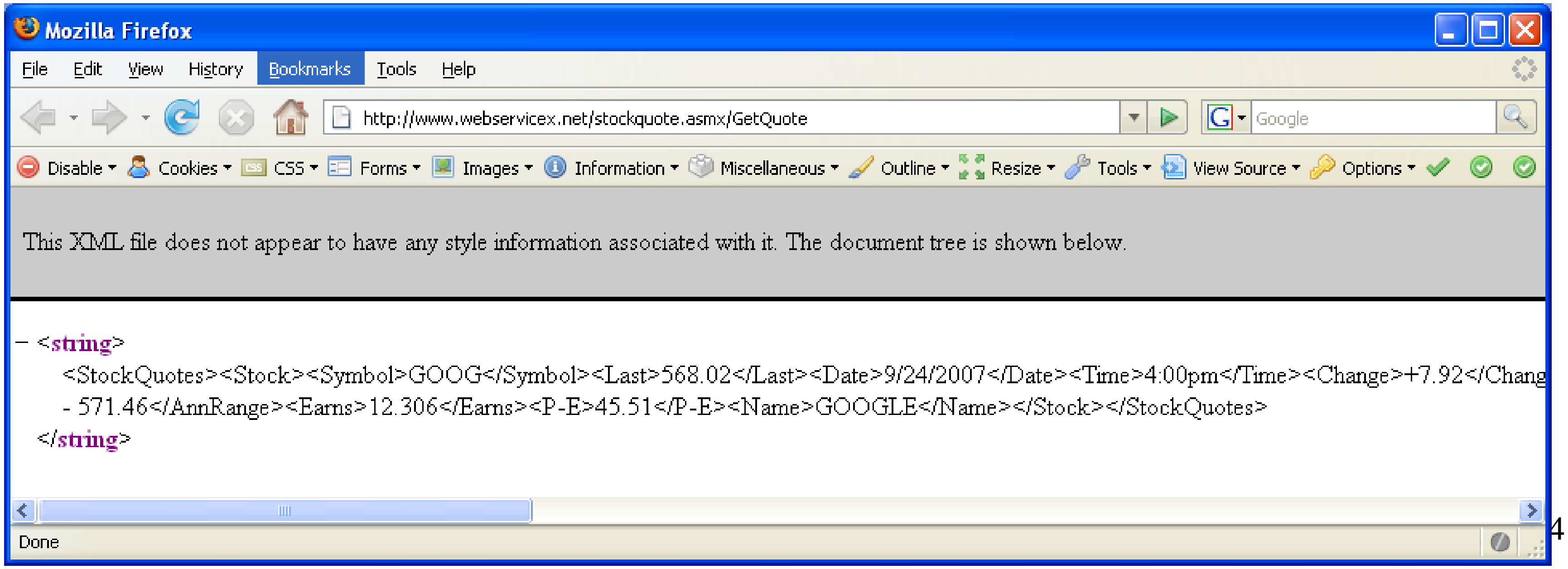
```
POST /stockquote.asmx HTTP/1.1
Host: www.webservicex.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.webserviceX.NET/GetQuote"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <soap:Body>
    <GetQuote xmlns="http://www.webserviceX.NET/">
      <symbol>string</symbol>
    </GetQuote>
```

Figure 3: Returned XML Document


http://www.webservicex.net/stockquote.asmx/GetQuote - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.webservicex.net/stockquote.asmx/GetQuote Go Links
Done Internet

```
<?xml version="1.0" encoding="utf-8" ?>
<string
  xmlns="http://www.webserviceX.NET/"><StockQuotes><Stock><Symbol>GOOG</Symbol><Last>568.02</Last><Date>9/24/2007</Da
  </PercentageChange><AnnRange>398.19 - 571.46</AnnRange><Earns>12.306</Earns><P-E>45.51</P-
  E><Name>GOOGLE</Name></Stock></StockQuotes></string>
```


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.webservicex.net/stockquote.asmx/GetQuote
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Done

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <string>
    <StockQuotes><Stock><Symbol>GOOG</Symbol><Last>568.02</Last><Date>9/24/2007</Date><Time>4:00pm</Time><Change>+7.92</Chang
    - 571.46</AnnRange><Earns>12.306</Earns><P-E>45.51</P-E><Name>GOOGLE</Name></Stock></StockQuotes>
  </string>
```

Making Use of a Web Service

- Whilst what we did on the last few slides enables us to get a single stock quote, it is not really very useful
- What is useful is that by invoking the web service we can get a chunk of XML data returned, and we can then manipulate this to include in an application
- So we need a way to integrate a “call” to a web service within a web-hosted program, and then arrange for manipulation of the data returned.

Four-Step Process of a Web Service

- The client (a browser of a certain type) calls the web service over a protocol (normally HTTP, could be SMTP or HTTPS though)
- The client selects a method and sends the method to the server with instructions - parameters and methods of transmission (HTTP-GET, HTTP-POST, or SOAP)
- The server returns value (in XML or JSON) and/or acknowledgement
- The client gets the result and acts on the received information.

The SOAP Approach

- A SOAP XML document is used for request/response
  - A SOAP document contains a <SOAP:Envelope> element
  - The <SOAP:Envelope> specifies several namespaces for XML Schema, instance and SOAP
  - The <SOAP:Envelope> contains a <SOAP:Header> element (optional) and a <SOAP:Body> element
- In a request SOAP document, the method to be called together with parameters are constructed within <SOAP:Body>
- In a response SOAP document, the response and the returned result are constructed within <SOAP:Body>
- Extra work is needed to construct and encode the SOAP documents
- SOAP will be studied in WAA unit – we won’t use it here.

The REST Approach

- Use either HTTP-GET or HTTP-POST, e.g.,
  http://www.webservicex.net/StockQuote.asmx/GetQuote?symbol=GOOG
- Assume that an XML fragment will be returned
- It’s just like what we have already been doing when “calling” a PHP file on the server
  - Send the method name (GetQuote)
  - Send the parameters (symbol=GOOG)
  - Pros: gets message stored in XML, uses HTTP (GET, POST) to transfer, uses URIs for identification, can use HTTP authentication for security, easy to use (with AJAX), easy to create mashups
- Cons: less secure than SOAP, the length of URI is limited while using GET

What is REST?

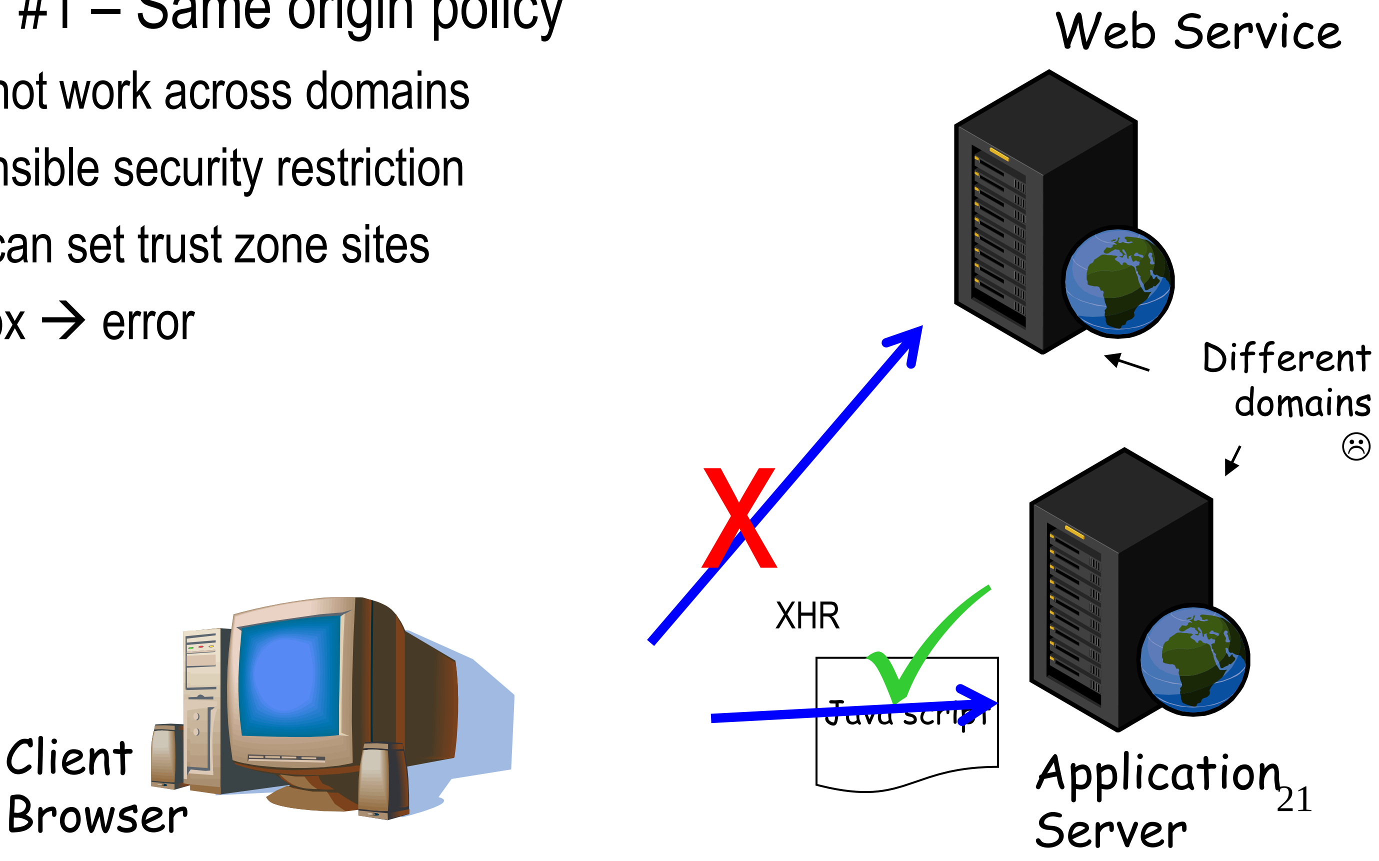
- REpresentational State Transfer
  - A **representation** of the resource is returned (e.g., purchaseConfirmed.html). The representation places the client application in a **state** (Purchase Confirmation state). The result of the client traversing a hyperlink in booking.html is that another resource is accessed. The new representation places the client application into yet another state (Booking state). Thus, the client application changes (**transfer**s) state with each resource representation --> Representational State Transfer!
- Defines a set of architectural principles by which you can design Web services
- REST not a standard - just an architectural style

The REST Approach

- Focuses on system’s resources & resource states
- A concrete implementation of a REST Web service follows four basic design principles:
  - Use HTTP methods explicitly.
  - Be stateless.
  - Expose directory structure-like URIs.
  - Transfer XML, JavaScript Object Notation (JSON), or both.
- Used by mainstream Web 2.0 service providers—including Yahoo, Google, and Facebook
- Initially REST was seen as a “kludge” – bypassing the more formal SOAP. Now REST is in quite common use.

Integrating a Web Service into an Application using Ajax and REST

- Passing web service's endpoint in the *open* method of an *XHR* object
  - ☐ Works in the same domain but …
- Problem #1 – Same origin policy
  - ☐ May not work across domains
    – sensible security restriction
  - ☐ IE - can set trust zone sites
  - ☐ Firefox → error



Same Origin Policy

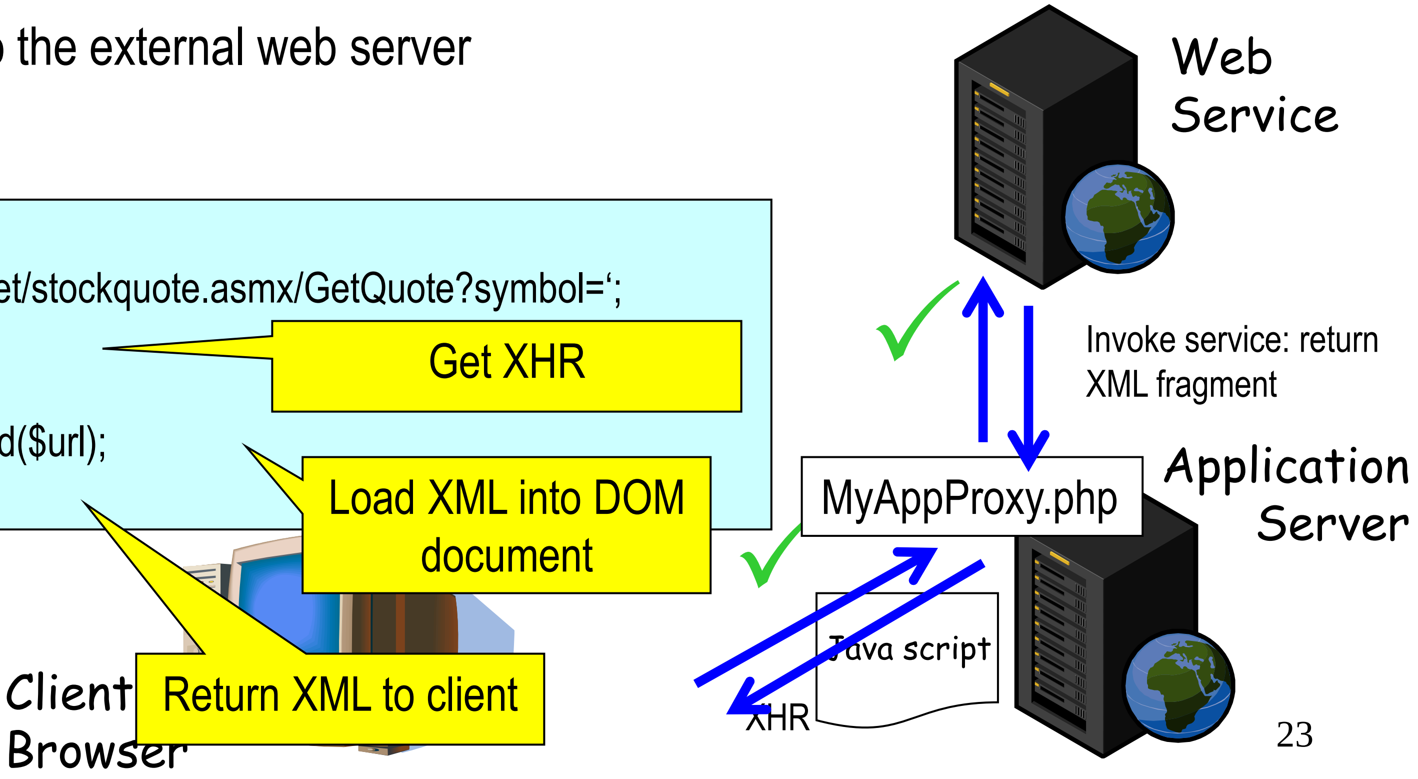
- http://www.mozilla.org/projects/security/components/same-origin.html
- Mozilla considers two pages to have the same origin if *protocol*, *port* (if given), and *host* (*domain*) are the same.
- Example: Javascript from http://store.company.com/dir/page.html.

URL	**Outcome**	**Reason**
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

Main Solution to Same Source Restriction

- Create an “Application Proxy”
  - Create a proxy page on the server
  - The XHR object calls the page on the server
    - E.g. MyAppProxy.php
  - This server-side page then relays the call on to the external web server

```
$url =
'http://www.webservicex.net/stockquote.asmx/GetQuote?symbol=';
$qs = $_GET["symbol"];
$url = $url . $qs;
$url = DOMDocument::load($url);
echo $url->saveXML();
```

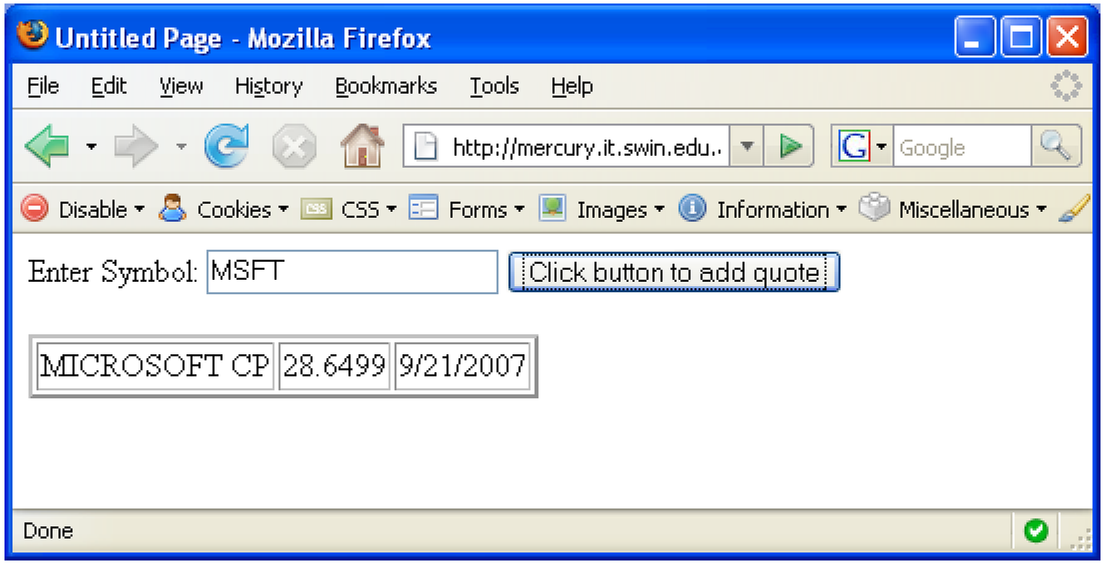


Main Solution to Same Source Restriction

- The main issue is that for security reasons, the client is not permitted to directly access a resource that is not from the same domain as the client-side page came from – this is a universal restriction of the WWW
- But there is no universal restriction placed on what a server can access – although, as we will see, a server's administrator can provide protection through use of a firewall, for example
- Hence the programming trick is for the client to get the server to go off to the external resource, get data as required, and then relay it back to the client.

Example: StockQuote

- Create an Ajax application to get latest stock quote
- 3 files (plus Web service) working together
  - Webservice.htm : (drives the application)
  - Webservice.js : (client-side processing of data)
  - ApplicationProxy.php : (server-side script to access the web service and forward data to the client)


Untitled Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://mercury.it.swin.edu..
Google
Disable Cookies CSS Forms Images Information Miscellaneous
Click button to add quote
Done

Enter Symbol: MSFT

MICROSOFT CP	28.6499	9/21/2007

Consuming a Web Service via a Proxy

webservice.htm

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Untitled Page</title>
<script type="text/javascript" src=”xhr.js"></script>
<script type="text/javascript" src="webservice.js"></script>
</head>
<body>
<form id="form1" name="form1">
Enter Symbol: <input type="text" name="stocksymbol" id="stocksymbol" />
<input type="button" onclick="sendData()" value="Click button to add quote" />
<br /><br />
<table id="table1">
</table>
</form>
</body>
</html>
```

XML will be formatted and displayed here

webservice.js & applicationproxy.php

```
var xhr = null;
xhr = createRequest();
function sendData()
{ xhr.abort(); // abort any prior activity on xhr
  var url = "applicationproxy.php?symbol=" + document.form1.stocksymbol.value;
  xhr.open("GET", url, true);
  xhr.onreadystatechange = getData;
  xhr.send(null);
}
```

This is new. It just ensures that since it uses a globally accessible XHR variable, any previous computation using that variable is aborted.

Callback function

```
<?php
header('Content-Type: text/xml');
?>
<?php
  $url = 'http://www.webservicex.net/stockquote.asmx/GetQuote?symbol=';
  $qs = $_GET["symbol"];
  $url = $url.$qs;
  $doc = DOMDocument::load($url);
  echo $doc->saveXML();
?>
```

Data passed from client

Call the web service, and load returned XML into DOM document

Return XML to client Javascript

webservice.js callback function

```
function getData()
{ if ((xhr.readyState == 4) &&( xhr.status == 200)) {
    var myXml = xhr.responseXML;
    var XMLDoc = null;
    var xmlobject = null;
    if (window.ActiveXObject) { // IE
      XMLDoc = myXml.childNodes[1].firstChild.nodeValue;
      var xmlobject = new ActiveXObject("Microsoft.XMLDOM");
      xmlobject.async="false";
      xmlobject.loadXML(XMLDoc);
    }
    else { // Eg Firefox
      XMLDoc = myXml.childNodes[0].firstChild.nodeValue;
      var parser = new DOMParser();
      var xmlobject = parser.parseFromString(XMLDoc, "text/xml");
    } …
```

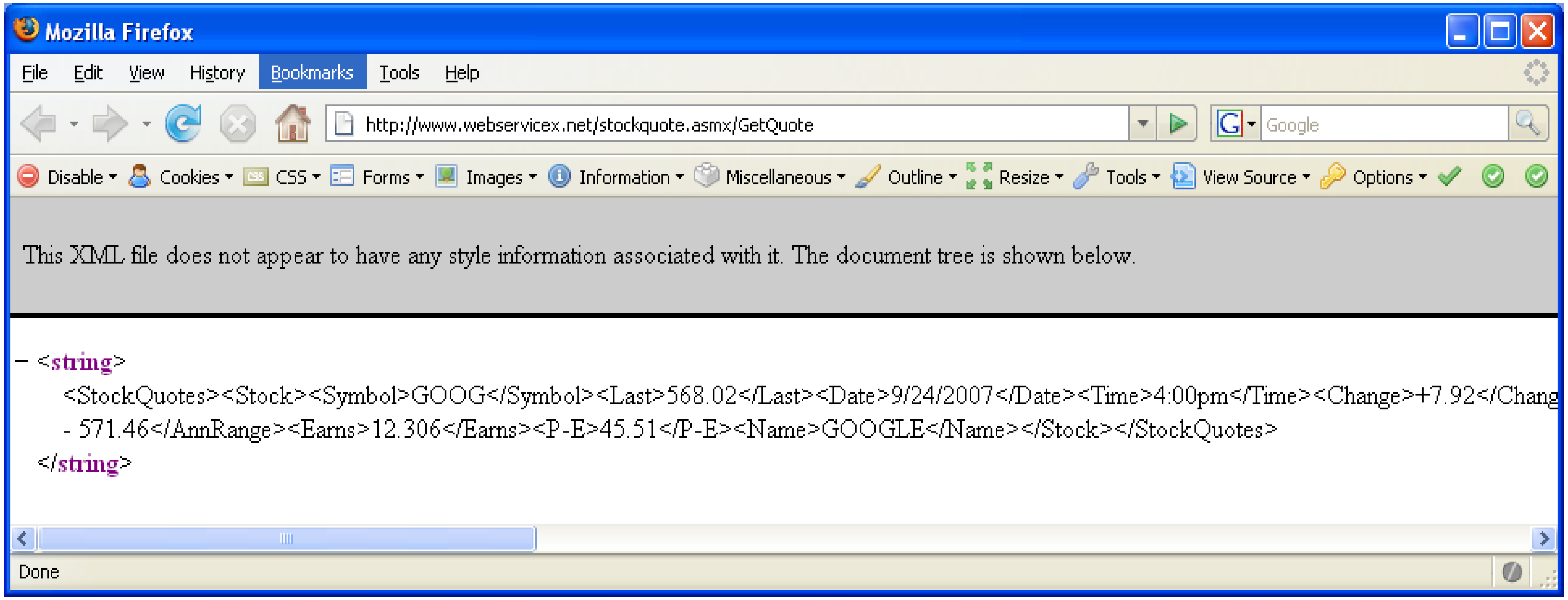
Get XML sent by PHP application proxy

Read XML DOM
IE: 2nd element
Firefox: 1st element

webservice.js (Cont'd)

```
…
var table = document.getElementById("table1");
   var row = table.insertRow(table.rows.length);
   var cell1 = row.insertCell(row.cells.length);
   cell1.appendChild(getText("Name",xmlobject));
   var cell2 = row.insertCell(row.cells.length);
   cell2.appendChild(getText("Last",xmlobject));
   var cell3 = row.insertCell(row.cells.length);
   cell3.appendChild(getText("Date",xmlobject));
   table.setAttribute("border", "2");
 }
}
function getText(tagName, xmlobject)
{ var tags = xmlobject.getElementsByTagName(tagName);
  var txtNode = null;
  if (window.ActiveXObject) { txtNode = document.createTextNode(tags[0].firstChild.text);}
  else { txtNode = document.createTextNode(tags[0].firstChild.textContent);}
  return txtNode; }
```

Format data retrieved from XML and add to the end of the table in the HTML doc

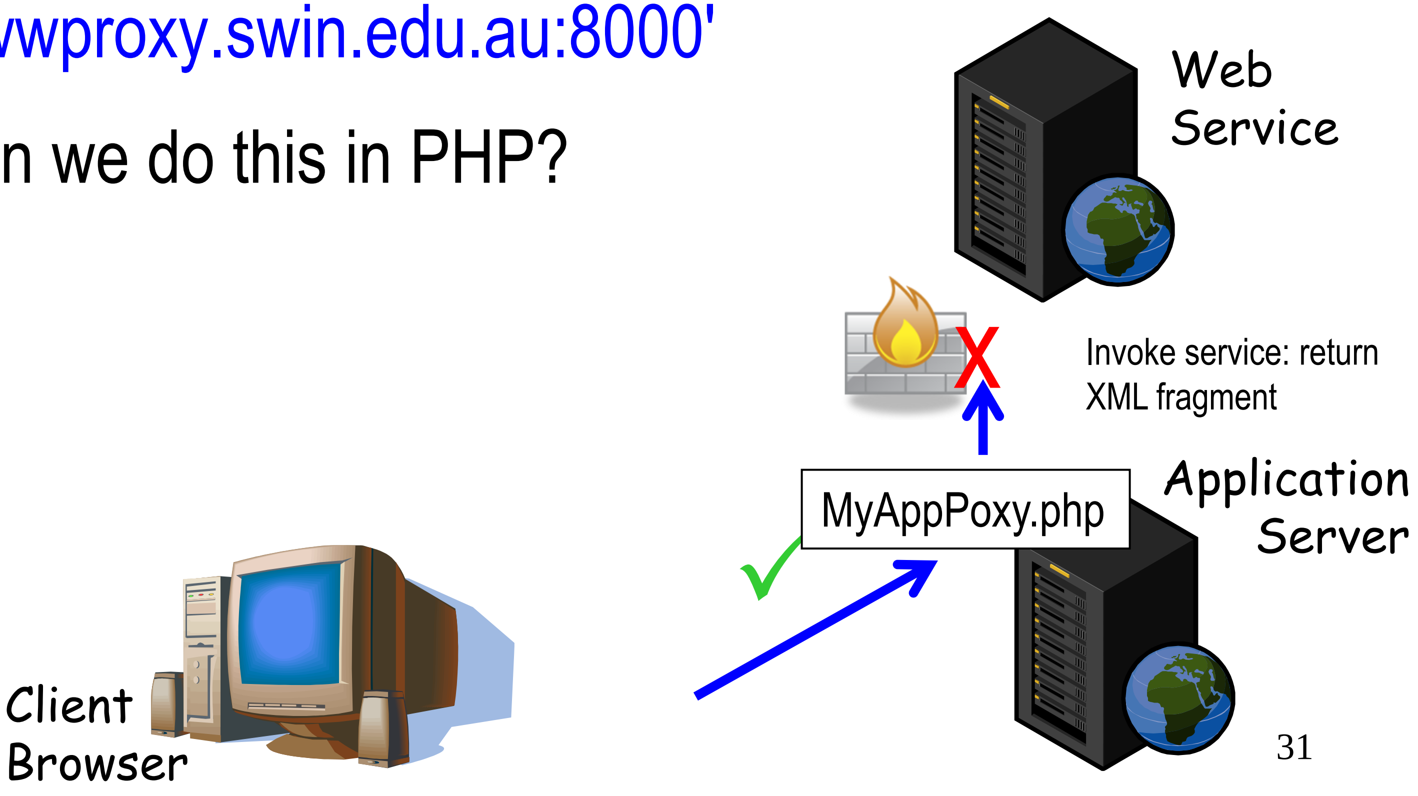

Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.webservicex.net/stockquote.asmx/GetQuote
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Done

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
– <string>
    <StockQuotes><Stock><Symbol>GOOG</Symbol><Last>568.02</Last><Date>9/24/2007</Date><Time>4:00pm</Time><Change>+7.92</Chang
    - 571.46</AnnRange><Earns>12.306</Earns><P-E>45.51</P-E><Name>GOOGLE</Name></Stock></StockQuotes>
  </string>
```

Firewall restrictions

- **Problem #2 invoking Web services using Ajax techniques – getting through the corporate firewall**
- Need to invoke service via firewall proxy server
- e.g. 'wwwproxy.swin.edu.au:8000'
- How can we do this in PHP?



cURL

- Libcurl (the multiprotocol file transfer library)
  - library that allows you to connect and communicate to many different types of servers with many different types of protocols
  - libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, and user+password authentication.
- PHP supports libcurl (& it is installed on mercury)
  - http://www.php.net/manual/en/ref.curl.php

How to use cURL with PHP

- Our example ApplicationProxy.php

```
<?php
  $url = 'http://www.webservicex.net/stockquote.asmx/GetQuote?symbol=';
  $qs = $_GET["symbol"];
  $url = $url.$qs;
  //$doc = DOMDocument::load(
  $curl = curl_init();
  curl_setopt($curl, CURLOPT_URL, $url);
  curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
  curl_setopt($curl, CURLOPT_PROXY, 'wwwproxy.swin.edu.au:8000');
  $pageContent = curl_exec($curl);
  curl_close($curl);
  $doc = DOMDocument::loadXML($pageContent);
  echo $doc->saveXML();
?>
```

cURL code

1. Initialise a cURL session

2. Set cURL options incl. proxy

3. Execute the cURL session

4.Close the cURL session

5. Load string into XML DOM document

Using APIs

- An API is similar to a web service
  - provides a set of operations (methods)
  - is publicly accessible
- Minor difference between web services and APIs
  - Web services comply with standards (REST or SOAP)
  - APIs don’t have to, they are proprietary in nature. Sometimes, a signup is needed to use it.

Most Common API Examples

- Flickr (www.flickr.com/services)
- YouTube (www.youtube.com/dev)
- Google Maps (www.google.com/apis/maps)
- eBay (http://developer.ebay.com/)
- del.icio.us (www.programmableweb.com/api/del.icio.us)
- Virtual Earth (www.viavirtualearth.com)