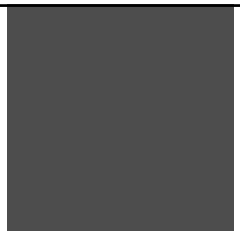


Web Application Development: Web Services and APIs

Week 9

SWINBURNE UNIVERSITY OF TECHNOLOGY

Content

- What a web service is
- Consuming a web service
- Web services and Ajax
- Using APIs

2

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

3

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*

4

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.

5

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.

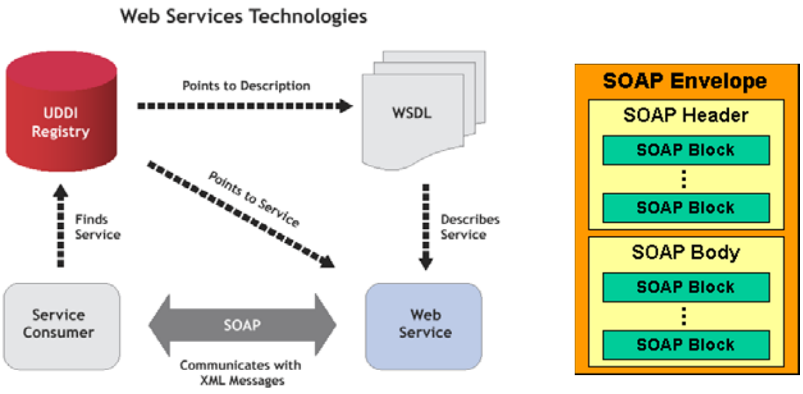
6

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.

7

Technologies



8

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

9

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/)

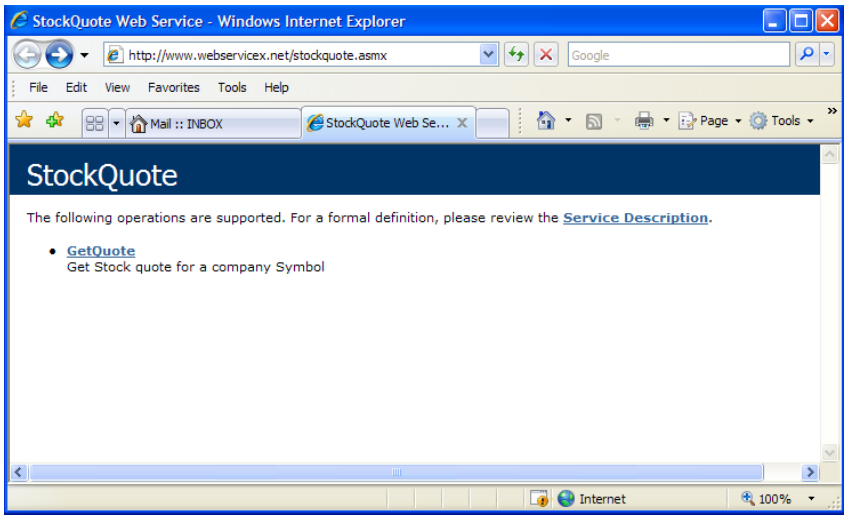
10

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

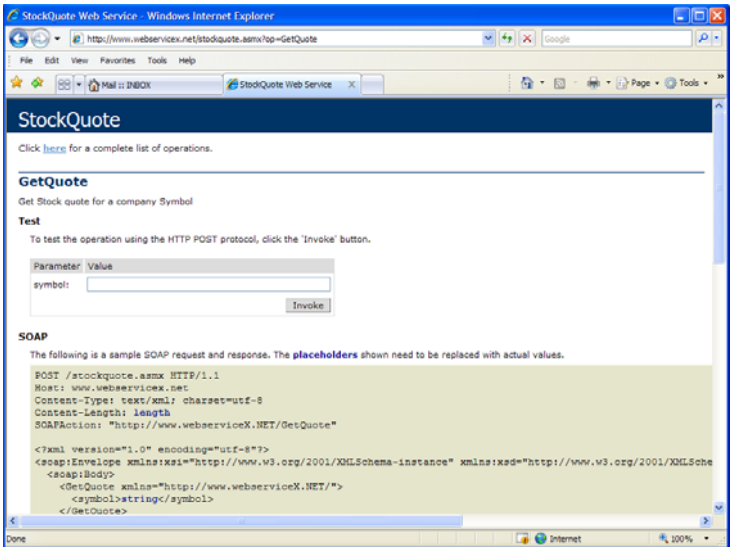
11

Figure 1: StockQuote Web Service



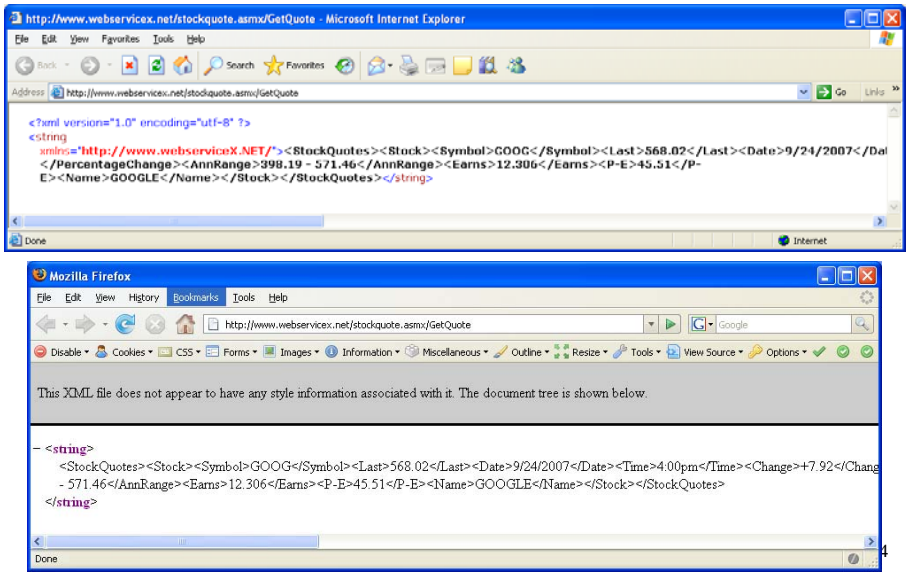
12

Figure 2: GetQuote Method and Its implementation



13

Figure 3: Returned XML Document



14

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.

15

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.

16

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.

17

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

18

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 (transfers) 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

19

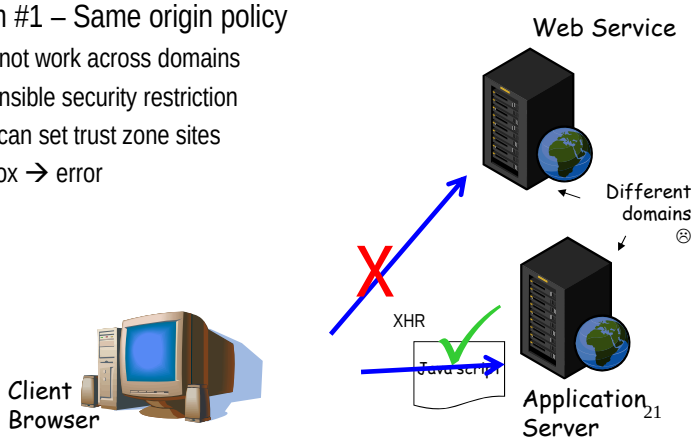
The REST Approach

- 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.

20

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



21

Same Origin Policy

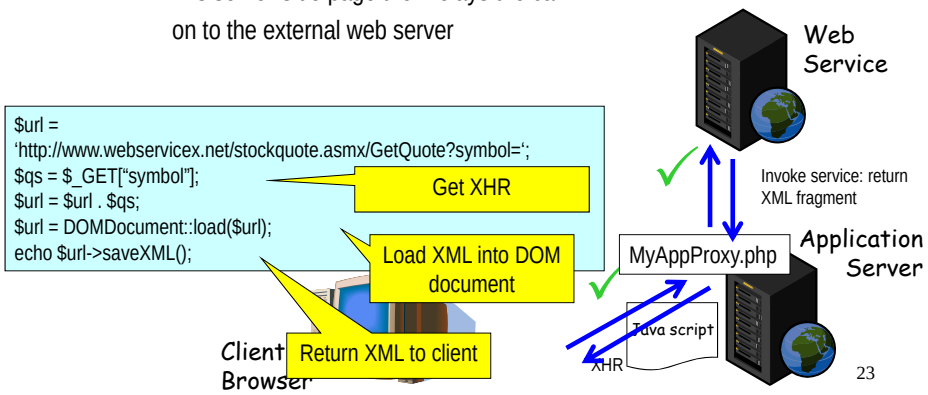
- http://www.mozilla.org/projects/security/components/same-origin.html
- Mozilla considers two pages to have the same origin if *protocol*, *port* (if given), and *host* (*domain*) are the same.
- Example: Javascript from http://store.company.com/dir/page.html.

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

22

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



23

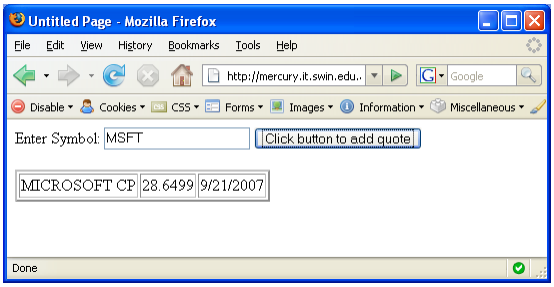
Main Solution to Same Source Restriction

- 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.

24

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)



25

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

26

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

28

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; }
```

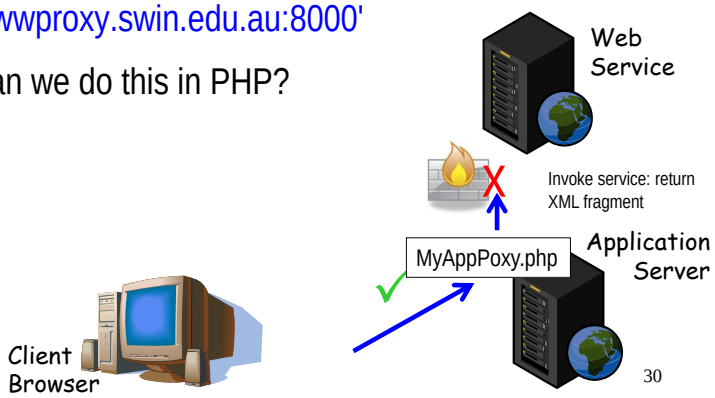
Format data retrieved from XML and add to the end of the table in the HTML doc

But it may still have problem!

29

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?



30

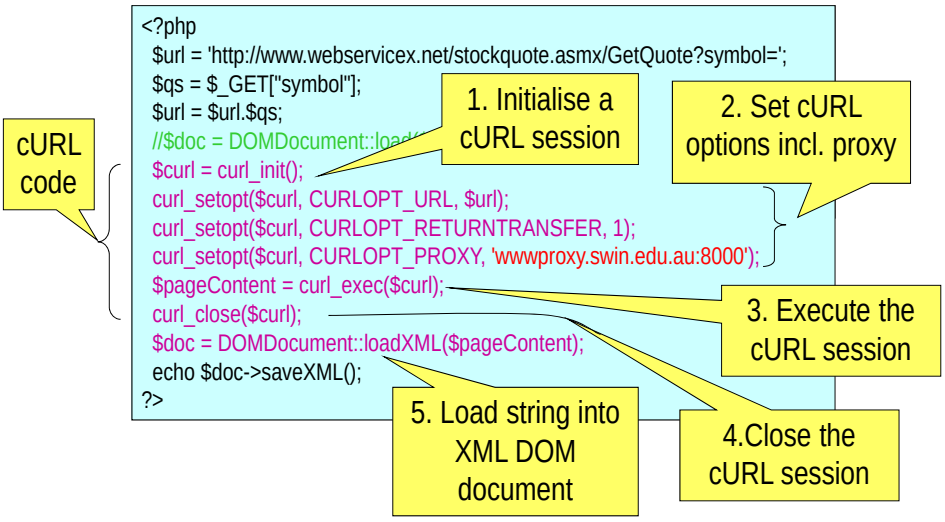
cURL

- 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

31

How to use cURL with PHP

- Our example ApplicationProxy.php



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.

33

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)

34

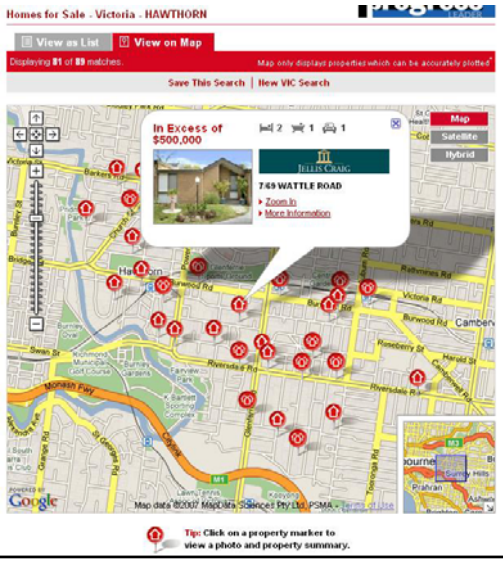
Example: Google Maps API

- We will explore in some detail the Google Maps API
- Version 2 is used in these notes though it is an old version https://developers.google.com/maps/documentation/javascript/v2/reference?csw=1
- Version 3 has been released in Google Labs https://developers.google.com/maps/documentation/javascript/

35

Example: Google Maps API

- In widespread commercial use: e.g. Real Estate



36

Using Google Maps APIs - API Key

- Although access to the GM API is free, a web-site that interacts with GM may do so under the authority and a special "key" may be needed (used to be needed and obtained from Google for Version 2 but no longer needed now; Version 3 *may or may not* need a key).
- This key is a code that is associated with a URL and has to be sent as a parameter when the API is accessed as a script.
- Need a script tag to access the API; this is where the key is included. The key is associated with your web directory, and all its sub-directories

37

Using Google Maps API: GMap2 with a key

gmexample.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="content-type" content="text/html; charset=utf-8"/>
   <title>Google Maps JavaScript API Example</title>
   <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=AB...g" type="text/javascript"></script>
   <script type="text/javascript">
      function initialize() {
        if (GBrowserIsCompatible()) {
          var map = new GMap2(document.getElementById("map_canvas"));
          map.setCenter(new GLatLng(-37.825, 145.050), 13);
        }
      }
   </script>
</head>
<body onload="initialize ()" onunload="GUnload()">
   <div id="map_canvas" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Specify the key if you have it

Where the map is to be located

Centre of map

magnification

Bold denotes an API item

Size of map

38

Using Google Maps API: GMap2 without a key

gmexample_no_key.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="content-type" content="text/html; charset=utf-8"/>
   <title>Google Maps JavaScript API Example</title>
   <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;" type="text/javascript"></script>
   <script type="text/javascript">
      function initialize() {
        if (GBrowserIsCompatible()) {
          var map = new GMap2(document.getElementById("map_canvas"));
          map.setCenter(new GLatLng(-37.825, 145.050), 13);
        }
      }
   </script>
</head>
<body onload="initialize ()" onunload="GUnload()">
   <div id="map_canvas" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Now, no key needed ☺

39

Using Google Maps API: Version 3

gmexample2.htm

```
<html>
<head>
<title>Simple Map Example</title>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>
<script>
  function initialize() {
    var myOptions = {
      zoom: 13,
      center: new google.maps.LatLng(-37.825, 145.050)
    };
    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
  }
</script>
</head>
<body onload="initialize()">
  <div id="map_canvas" style="width:100%; height:100%"></div>
</body>
</html>
```

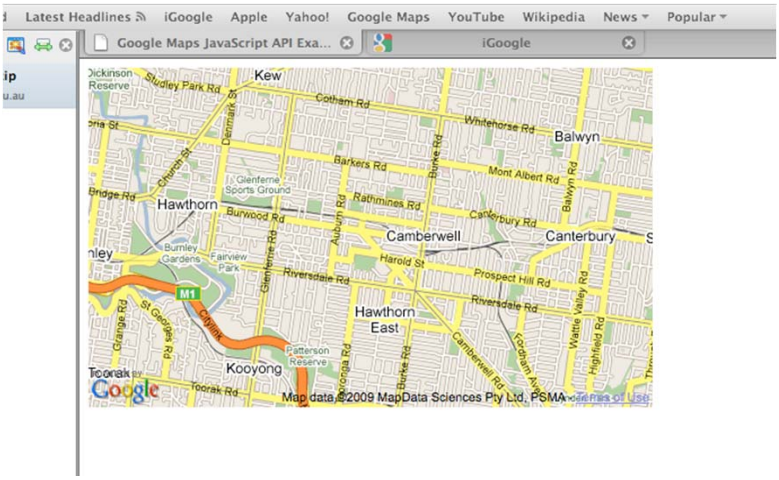
No key needed but need to specify this!!

We are not using a sensor to detect position

Here we use a JavaScript object literal to describe map parameters

40

Outcome of Running (V2 version)



41

Creating a Map Object

- GBrowserIsCompatible: Checks the compatibility of your browser with Google Maps. Should really include an “else” clause to handle failure gracefully – at least a simple message to the user saying that browser is incompatible with Google maps. (We could have used this in the V3 example too!)
- GMap2: creates a new map object, and locates it in the document (in the div which has id “map”). (Note the different name for the map object in V3).
- setCenter: sets the center of the map to a specified latitude and longitude, and also specifies the scale point (here 13).
- Note that we specify the center of the map by first constructing a GLatLng object with given latitude and longitude properties. (Note the different name for the object in V3.), then center the map on that object.

42

Geocoder Object

- A Geocoder object converts addresses to latitudinal and longitudinal coordinates
- getLatLng: returns the latitude/longitude (GLatLng) for an address (the first parameter), and passes this to a function to execute (the second parameter). For example
  geocoder.getLatLng("Paris", function(point){map.setCenter(point, 13)});
  first causes the GLatLng for the address of "Paris" to be created, and then causes this to be passed as the parameter "point" of the function defined as the second parameter. This function causes the map to be centered on the point, and to be given zoom scale 13.

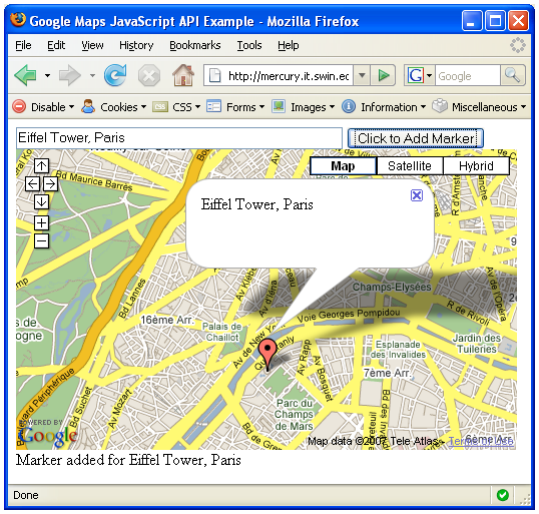
43

Using Other Google Maps APIs

- GXmlHttp object: makes use of the XMLHttpRequest via a "factory" method
  - Create(): creates an XMLHttpRequest object
  - Use this in a GM application, rather than our own version in xhr.js
  - Dropped from Version 3 – use your own, or some Ajax framework XHR creation function
- GDownloadUrl: downloads the required data directly

44

Using Google Maps API: an example



We display a map, a text box into which the user can type an address and a button. When the user presses the button, the map becomes centered on the address in the text box, a marker is displayed on the map at that point, and an information "bubble" is displayed that gives the details of the address. Below the map, a message is displayed indicating that a marker has been added.

Marker data is saved so that when the map is re-loaded, the marker details are still present. (This requires that we store persistent data on the server.)

45

System Design

- Main HTML file
  - Displays the map, text box, button and message zone
  - When loaded, the on-load event-handler reads the marker data previously stored on the server and places markers on map
  - When a place name is entered in the text box and the button is pressed, the on-click event-handler for the button creates a new marker, places it on the map, centres the map on the new marker, displays a message that the marker has been added, and writes the updated marker data to the marker file
- Main JavaScript file
  - Creates XHR and Geocoder objects for use
  - Implements the various call-back functions and subsidiary functions necessary for the system

46

System Design

- Reading from the XML file
  - Done directly using Google maps function
- Writing to XML file (every time a new marker is created)
  - Done via an Ajax call to a PHP file, which assembles the XML as text and writes it to the file
- XML File
  - Stores data about markers to be placed on the map – latitude, longitude, name
- PHP file
  - Used to manage the update of the XML file

47

map.htm : Main HTML File

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
   <meta http-equiv="Pragma" CONTENT="no-cache" />
   <meta http-equiv="Expires" CONTENT="-1" />
   <title>Google Maps JavaScript API Example</title>
   <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=AB..g"
           type="text/javascript"></script>
   <script src="map.js" type="text/javascript"></script>
</head>
<body onload="load()" onunload="GUnload()">
   <input id="address" name="address" type="text" size="50"/>
   <input type="button" onClick="addMarker()" value="Click to Add Marker" />
   <div id="map_canvas" style="width: 500px; height: 300px"></div>
   <span id="markerConfirm" />
</body>
</html>
```

map.js: Main JavaScript File

```
var map = null;  var geocoder = null; var xhr = null;

xhr = GXmlHttp.create(); // Google Maps API way to create an XMLHttpRequest object
geocoder = new GClientGeocoder();

// function to load map and saved markers; execute when system loads up
function load() { ... ... }

// function to add marker to the displayed map, based on address in input field
function addMarker() { ... ... }

// function to save the details of a marker; calls a PHP file to do the actual work, using Ajax
function saveMarker(point, address) {... ... }

// function to place the “marker added” confirmation message in document
function getConfirm() {... ... }

// function to connect with server and read marker data from file data.xml and place on map
function placeMarkers() { ... ... }

// function to create a new marker with click-open information window for a given point with given name
function createMarker(point, name) { ... ... }
```

Data.xml: XML Data File

```
<?xml version="1.0"?>
<markers>
   <marker lat="48.858205" lng="2.294359" address="Eiffel Tower"/>
</markers>
```

Note : At the start, this file needs not exist. It will be created when the first marker is placed.

50

map.js: Function load

```
// function to load the map and saved markers; executed when system loads up
function load() {
   if (GBrowserIsCompatible()) { // check Browser is compatible with API
      // create new map instance at div “map_canvas”
      map = new GMap2(document.getElementById("map_canvas"));
      // add Zoom and Type controls
      map.addControl(new GSmallMapControl());
      map.addControl(new GMapTypeControl());
      // place markers saved from last time; if there are none, map will center on Paris
      placeMarkers();
   }
}
```

51

map.js: Function placeMarkers

```
// connect with server and read marker data from file  data.xml via the Google GDownloadUrl function
// (the data file as 1st parameter, and a call-back function as 2nd parameter)
// The call-back function (data downloaded as 1st parameter, and response code from download as 2nd parameter)
function placeMarkers() {
   GDownloadUrl("getdata.php", function(data, responseCode) {
      if(responseCode == 200){  // data loaded ok
         var xml = GXml.parse(data); //convert the data to an XML DOM fragment; will get from cache if not cleared
         var markers = xml.documentElement.getElementsByTagName("marker"); // get the marker elements in the data
         if (markers.length == 0) { // if no markers, set map center on Paris
           geocoder.getLatLng("Paris", function(point){map.setCenter(point,13)});}
         else { // place the markers
           for (var i = 0; i < markers.length; i++) {
             var point=new GLatLng(parseFloat(markers[i].getAttribute("lat")), parseFloat(markers[i].getAttribute("lng")));
             map.setCenter(point, 13);
             map.addOverlay(createMarker(point, markers[i].getAttribute("address")));
           }
         }
      }
      else if (responseCode == -1) { alert("Data request timed out. Please try later."); }
      else { // center the map on Paris
         geocoder.getLatLng("Paris", function(point){map.setCenter(point, 13)});
      }
   }); // finish GDownloadUrl function call
}
```

Overlays are objects on the map that are tied to latitude/longitude coordinates, so they move when you drag or zoom the map. Overlays reflect objects that you "add" to the map to designate points, lines, or areas.

map.js: Function createMarker

```
// function to create a new marker with click-open information window
// for a given point with given name
function createMarker(point, name) {
    var marker = new GMarker(point);
    GEvent.addListener(marker, "click", function() {
        marker.openInfoWindowHtml("<b> " + name + " </b>");
     });
    return marker;
}
```

53

map.js: Function addMarker

```
// function to add a marker to the displayed map, based on address in input field
function addMarker(){
    // read address from input field
    var address = document.getElementById("address").value;
    // convert address to GLatLng, center the map there, and place marker
    geocoder.getLatLng(address, function(point) {
        if (!point) {
            alert(address + " not found");
        }
        else {
            map.setCenter(point, 13);
            var marker = createMarker(point, address);
            map.addOverlay(marker);
            marker.openInfoWindowHtml(address);
            saveMarker(point, address);
        }
    });
}
```

54

map.js: Function saveMarker

```
// function to save the details of a marker at a particular point, with a particular address
// calls a PHP file to do the actual work, using Ajax
function saveMarker(point, address){
   var lat = point.lat();
   var lng = point.lng();
   var url = "saveMarkers.php?lat=" + lat + "&lng=" + lng + "&address=" + address;
   xhr.open("GET", url, true);
   xhr.onreadystatechange = getConfirm;
   xhr.setRequestHeader("Content-Type", "text/xml" ); //use this if no response required
   xhr.send(null);
}
```

55

map.js: Function getConfirm

```
// function to place marker added confirmation message in document
// runs as call-back when a new marker has been added successfully
function getConfirm(){
   if ((xhr.readyState == 4) &&(xhr.status == 200)) {
      var markerAddConfirm = xhr.responseText;
      var spantag = document.getElementById("markerConfirm");
      spantag.innerHTML = markerAddConfirm;
   }
}
```

56

getdata.php

```
<?php
        $url = '../../data/data.xml';
        $doc = new DomDocument();
        $doc->load($url);
        ECHO ($doc->saveXML());
?>
```

57

saveMarkers.php

```
<?php
   // read data passed from client
   $url = '../../data/data.xml';
   $lat  = $_GET['lat'];
   $lng  = $_GET['lng'];
   $address  = $_GET['address'];
   // create new DOM document
   $doc = new DomDocument();
   // if marker file does not exist, set up DOM doc with new markers node
   // but no marker nodes
   if (!file_exists($url)){
      $node = $doc->createElement('markers');
      $newnode  = $doc->appendChild($node);       }
   // else read the DOM document from the XML file
   else {
      $doc->preserveWhiteSpace = FALSE;
      $doc->load($url);
   }
   // now add the new marker node
```

58

saveMarkers.php

```
    // now add the new marker node

    // select the markers element – new node has to be added as a child of this
    $markerselements = $doc->getElementsByTagName('markers')->item(0);
    // create a new marker element from the data passed in
    // and append it to the end of the list of existing marker elements
    $newmarker = $doc->createElement('marker');
    $newmarker = $markerselements->appendChild($newmarker);
    $newmarker->setAttribute("lat", $lat);
    $newmarker->setAttribute("lng", $lng);
    $newmarker->setAttribute("address", $address);
    // save the updated XML to the XML file, and send message back to the client
    $doc->formatOutput = true;
    $strConfirm = "Marker added for ".$address;
    $doc->save($url);
    ECHO ($strConfirm);
?>
```

59