Lecture 14
Web Services and Mashups

Play

Outline

- Web Services
- Mashups

What is a Web Service

- A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP.
- Using a web service could be as simple as logging into a site or as complex as facilitating a multi-organization business negotiation. Ex: Creating a Sales Order and initiating a workflow.

Web Services: Alternate definition

- A web service is just a web page meant for a computer to request and process.
- More precisely, a Web service is a Web page that's meant to be consumed by an autonomous program as opposed to a Web browser or similar UI tool.

Architecture of Web Services

- RPC – Remote Procedure Call
- SOA – Service-oriented architecture
- REST – Representational State Transfer

Remote Procedure Calls

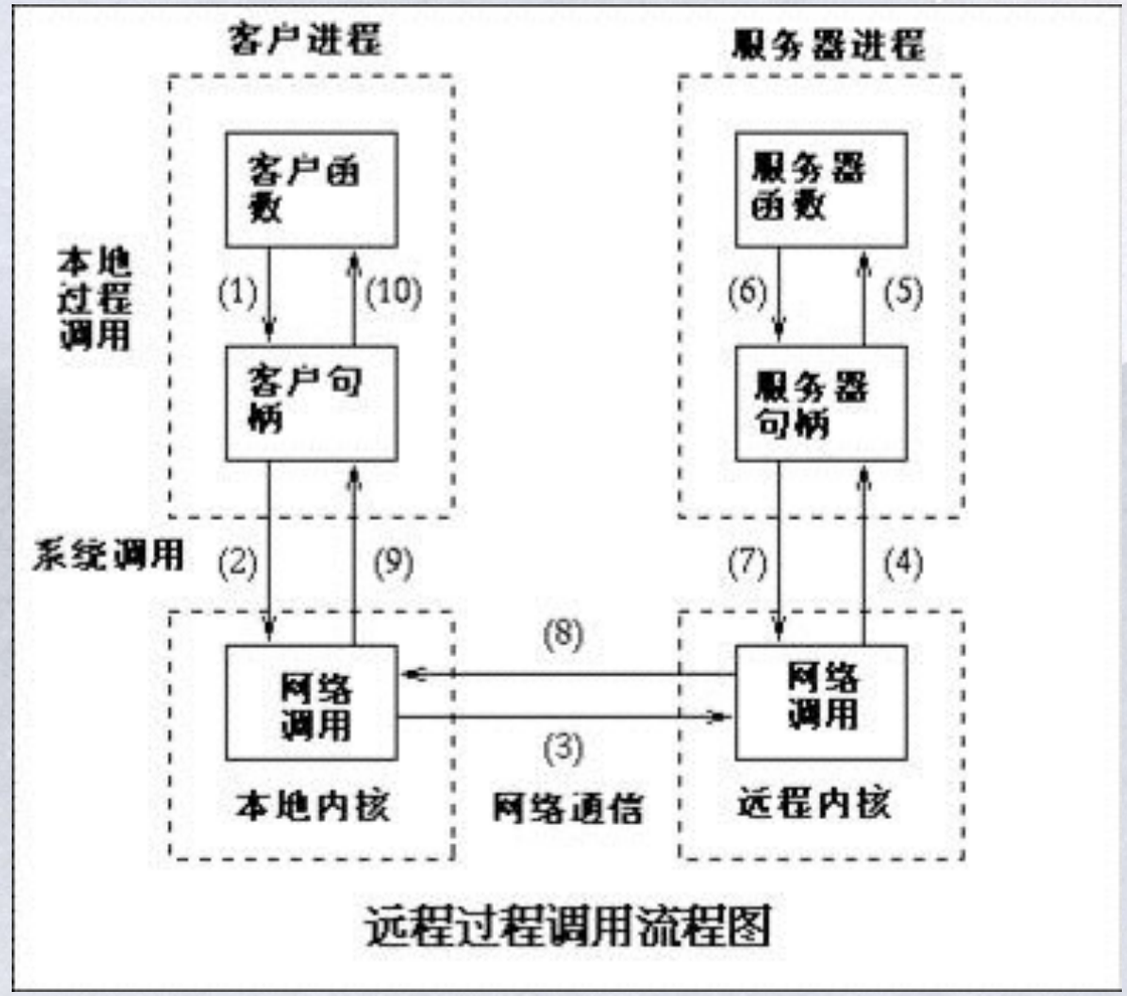
- Remote Procedure Call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space.
- RPC (Remote Procedure Call) goes back at least as far as 1976, when it was described in RFC 707.

RPC – How does it work ?

- A client makes a call to the procedure in a remote machine (server) which has the business logic with appropriate parameters.
- The server receives the parameters and executes the procedure locally and then transmits the results back to the client.
- The client receives the results.

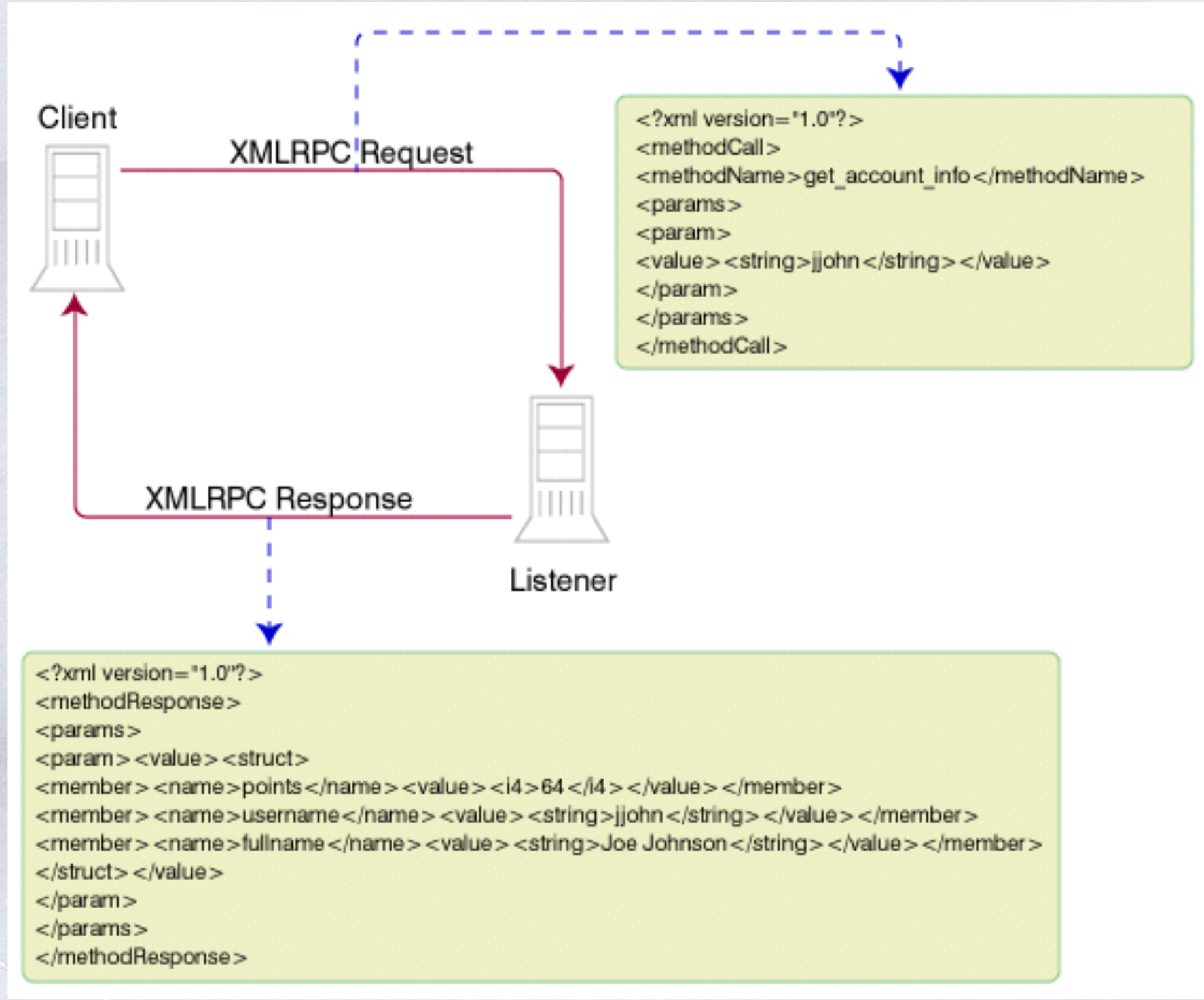
RPC – More formally

- The client calls the Client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way.
- The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshalling.
- The kernel sends the message from the client machine to the server machine.
- The kernel passes the incoming packets to the server stub.
- Finally, the server stub calls the server procedure. The reply traces the same in other direction.



远程过程调用流程图

XML RPC


Client
XMLRPC Request
<?xml version="1.0"?>
<methodCall>
<methodName>get_account_info</methodName>
<params>
<param>
<value><string>jjohn</string></value>
</param>
</params>
</methodCall>
XMLRPC Response
Listener
<?xml version="1.0"?>
<methodResponse>
<params>
<param><value><struct>
<member><name>points</name><value><i4>64</i4></value></member>
<member><name>username</name><value><string>jjohn</string></value></member>
<member><name>fullname</name><value><string>Joe Johnson</string></value></member>
</struct></value>
</param>
</params>
</methodResponse>

JSON

- Object: collection of comma separated Attributes
  - var object = { }
- Attributes: name-value pairs
  - “name”: “Duke”
  - “age”: 10
- Array: List of comma separated objects
  - [{“name”: “Duke”, “age”: 10}, {name:“Tux”, “age”: 20}]
- Data types
  - (String, Number, boolean, array, object, null)

```
{
        "firstName": "John",
        "lastName": "Smith",
        "age": 25,
        "address": {
         "streetAddress": "21 2nd Street",
          "city": "New York",
          "state": "NY",
          "postalCode": "10021"
        },
      "phoneNumber": [
                { "type": "home", "number": "212 555-1234" },
                { "type": "fax", "number": "646 555-4567" }
                ]
}
```

Service Oriented Architecture

- As per Wikipedia, SOA is a flexible set of design principles used during the phases of systems development and integration in computing.
- A system based on a SOA will package functionality as a suite of interoperable services that can be used within multiple separate systems from several business domains.

Service Oriented Architecture

Requestor

Registry

FIND, UDDI

PUBLISH, UDDI

BIND, SOAP/
WSDL

Providers

Components of SOA

- Participants:
  - Provider
  - Registry (broker)
  - Requestor
- Interactions:
  - Publishing
    - Direct
    - HTTP GET request
    - Dynamic discovery
  - Service location (finding)
  - Binding

Qualities of SOA

- It is modular, interoperable, reusable, and component-based.
- It is standards-compliant.
- It has identifiable services, providing deliverables, with monitoring and tracking.

Major Technologies for WS

- SOAP – Simple Object Access Protocol
- WSDL – Web Services Definition/ Description Language
- UDDI –Universal Description, Discovery, and Integration

SOAP

- SOAP stands for Simple Object Access Protocol. It is based on XML.
- It is a wired protocol and was defined to be highly interoperable.
- As of today there are extensive libraries support in virtually all languages to support SOAP based communication.

Syntax Rules

- A SOAP message must be an XML encoded document.
- A DTD reference must not be included in a SOAP document.
- XML processing instructions must not be included in a SOAP document.
- The SOAP Envelope namespace must be used in the document.
- The SOAP Encoding namespace must be used in the document.

the basic skeleton for SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    <!-- Header information -->
  </soap:Header>

  <soap:Body>
    <!-- Body Information -->
  </soap:Body>

</soap:Envelope>
```

A SOAP request using AWS

```
<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <namesp1:AsinSearchRequest xmlns:namesp1="urn:PI/DevCentral/SoapService">
         <AsinSearchRequest xsi:type="m:AsinRequest">
            <asin>0596528388</asin>
            <page>1</page>
            <mode>books</mode>
            <tag>associate tag</tag>
            <type>lite</type>
            <dev-tag>developer token</dev-tag>
            <format>xml</format>
            <version>1.0</version>
         </AsinSearchRequest>
      </namesp1:AsinSearchRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP: Fault

- SOAP errors are handled using a specialized envelope known as a Fault Envelope.
- If an error occurs while the server processes a SOAP message, it constructs a SOAP Fault and sends it back to the client.

```
<?xml version='1.0' encoding='UTF-8'?>
<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>
    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring>Test Fault</faultstring>
         <faultactor>/soap/servlet/rpcrouter</faultactor>
         <detail>
            <stackTrace>[SOAPException: faultCode=SOAP-ENV:Server;
                        msg=Test Fault]
            at StockQuantity.getQty(StockQuantity.java:21)
            at java.lang.reflect.Method.invoke(Native Method)
            at org.apache.soap.server.RPCRouter.invoke(RPCRouter.java:146)
            ...
            at org.apache.tomcat.util.ThreadPool$ControlRunnable.run(
                ThreadPool.java:501)
```

Web Services Description Language

- Web Services Description Language (WSDL) is an XML-based protocol used to describe web services, what public methods are available to them, and where the service is located.
- Six major elements are included in a WSDL document:
  - The data types that the web service uses
  - The messages that the web service uses
  - The operations that the web service performs
  - The communication protocols that the web service uses
  - The individual binding addresses
  - The aggregate of a set of related ports

What is REST?

- REST stands for Representational State Transfer.
- Its basically a stateless communication protocol relying on lightweight protocols like HTTP.
- It is basically an architectural design methodology.

An Overview of REST

- REST on the other hand has gained in popularity due to its simplicity, reduced transfer requirements, and usage of JSON.
  - Returns data, doesn't expose methods
  - Supports XML and JSON
  - Uses explicit HTTP Verbs
  - Point to point connections
  - Ajax (JavaScript) Friendly
  - Stateless

REST: The other verbs

- GET to retrieve information
- POST to add new information, showing its relation to old information
- PUT to update information
- DELETE to discard information

RESTful Application Cycle

Resources are identified by URIs
↓
Clients communicate with resources via requests using a standard set of methods
↓
Requests and responses contain resource representations in formats identified by media types
↓
Responses contain URIs that link to further resources

REST: Give Everything an ID

- ID is a URI
  - http://example.com/widgets/foo
  - http://example.com/customers/bar
  - http://example.com/customers/bar/orders/2
  - http://example.com/orders/101230/customer

REST vs. SOAP

REST	SOAP
✦ Returns data ✦ Supports XML and JSON ✦ Uses CRUD/ HTTP Verbs • Point to point connections ✦ Ajax (JavaScript) Friendly • Stateless	• Exposes operations/ method calls • All calls sent through POST • Larger packets of data, XML based ✦ Can be stateless or stateful ✦ WSDL Support

✦ Advantage goes to...

Ajax and Web Services

- Any client request for a web service is handled in one of two ways.
  - Requests are made using a hidden <iframe> or <frame> element to handle the sending and receiving, and then the data is collected from the frames.
  - a call is made to a server script that handles the sending and receiving, and the client gets the data from the server-side script's response.

The client request for a web service

```
/*
 * Example  The client request for a web service.
 */

new Ajax.Request('amazon.php', {
    method: 'get',
    parameters: { asin: '0596528388' },
    onSuccess: distributeResults
});
```

Server-Side Scripting to Services

- Seriously, using PHP to access web services is fairly simple.
- The one detail that is necessary to address is whether the web service interfaces with SOAP or REST. This will determine what our server script will look like.

A SOAP request to AWS using PHP

```
<?php
/**
 * Example  A SOAP request to AWS using PHP.
 * This example shows how to create a SOAP request using PHP's SOAP extension to
 * an AWS method.
 */

/* Create a new instance of the SOAP client class */
$client = new SoapClient('http://soap.amazon.com/schemas2/
AmazonWebServices.wsdl');
/* Create the parameters that should be passed */
$params = Array(
   'asin'      => mysql_real_escape_string($_REQUEST['asin']),
   'type'      => 'lite',
   'tag'       => '[associates id]',
   'devtag'    => '[developer token]'
) ;
/* Call the AWS method */
$result = $client->ASINSearchRequest($params);
?>
```

A REST request to AWS using PHP

```
<?php
/**
 * Example  A REST request to AWS using PHP.
 * This example shows how to create a REST request using PHP to an AWS
method.
 */

$assoc_id = '[associate id]';
$dev_token = '[developer token]';

$xml_link = sprintf('http://xml.amazon.com/onca/xml3?t=%s&dev-t=
%s&AsinSearch=%s&mode=books&type=lite&f=xml',
    $assoc_id,
    $dev_token,
    urlencode($_REQUEST['asin'])
);

$results = file_get_contents($xml_link);
?>
```

Using feeds to distribute information with PHP

```
<?php
/* Get the RSS feed from Yahoo! for my zip code */
$results = file_get_contents('http://weather.yahooapis.com/forecastrss?p=62221');
$xml = new SimpleXMLElement($results);

/* Create the XML for the client */
$response = '<response code="200">';
$response .= '<weather>';

/* Create the prefix context for the XPath query */
$xml->registerXPathNamespace('y', 'http://xml.weather.yahoo.com/ns/rss/1.0');
/* Gather the temperature data */
$temp = $xml->xpath('//y:condition');

/* Fill in information for the client */
$response .= '<temp>'.$temp[0]['text'].', '.$temp[0]['temp'].' &#176;F</temp>';
$response .= '<img>'.$xml->channel->image->url.'</img>';
$response .= '<link>'.$xml->channel->image->link.'</link>';
$response .= '</weather>';
$response .= '</response>';

/*
 * Change the header to text/xml so that the client can use the return string
 * as XML
 */
header('Content-Type: text/xml');
/* Give the client the XML */
print($response);
?>
```

Web Feeds

- There is nothing new about syndication, RSS, or Atom unless you create a new web service that aggregates feeds and sends them to the client.

Web Service APIs

- The API is the easiest means by which a developer can create a way to interact with an existing service as it was intended.
  - This creates the opportunity for cleaner code,
  - it will reduce the amount of time needed to create the methods of interaction.

Publicly Available Web Services

- Blogging Services
- Bookmark Services
- Financial Services
- Mapping Services
- Music/Video Services
- News/Weather Services
- Photo Services
- Reference Services
- Search Services
- Shopping Services
- Other Services

Arguments against REST

- REST APIs are difficult to build and maintain
- REST is not secure (anyone can access it)
- There are no strict standards for REST APIs
- REST services are not reliable

Arguments against REST

- REST APIs are difficult to build and maintain

➡ And as REST has gained in popularity there are more tools to help keep documentation up to date. As long as basic guidelines are followed REST APIs can be maintained and provide backwards compatibility.

Arguments against REST

- REST is not secure (anyone can access it)

➡ There are many precautions we can take to ensure our REST API only serves those who we want to have access to the data, including OAuth tokens and IP restrictions. However, it's important to remember that like anything, JavaScript/ AJAX should only be used to make calls to public or insensitive data.

Arguments against REST

- There are no strict standards for REST APIs

➡ This is true as there is no governing body for REST. However, we will be going over some general guidelines that should be used when building your API and as long as these are followed your API should be totally awesome.

Arguments against REST

- REST services are not reliable

➡ REST services are just as reliable as SOAP APIs when the proper checks are implemented on the client's side. These should include checking the request headers returned by CURL requests. For this reason it is important to make sure you are using the correct response codes and giving easy to understand error messages.

A Good REST API Offers

- Generality – language agnostic
- Familiarity – developers are used to it
- Scalability – can grow with usage/ demand
- Segmentation – sections can be updated independently
- Speed – low data transfer, cacheable
- Security – personal data remains protected
- Encapsulation – ability to send data as objects

A Good REST API Uses Explicit HTTP Verbs

- Create – POST
- Read – GET
- Update – PUT
- Delete - DELETE

A Good REST API

- Is built for longevity and designed to evolve...

A Good REST API Uses

- Hypermedia as the Engine of Application State (HATEOAS)
  - Hypermedia constraint states that interaction with an endpoint should be defined within metadata returned with the output (URL)
  - Apply state transitions (at run time) by following links
- Resource APIs allow URL to be known when code is written, and not discover at run time

A Good REST API Uses

- Hypermedia As The Engine Of Application State(HATEOAS)
  - HATEOAS relies on the concept that applications will change, data required will change, and paths can change. However, these changes should not effect existing clients. Hypermedia is stressed over a server oriented architecture.

A Good REST API Uses

- In order to accommodate these changes HATEOAS not only sends back the data requested, but the next possible actions as urls, this allows the client's application to be dynamic while accommodating any updates to the API itself

A Good REST API Uses

- Hypermedia Application Language provides a format for describing additional resources (ie links) using a _links -> item -> connection type => url grouping.
- Content type: application/hal+json
-

Using HATEOAS

Sample JSON Response


```
{"data":{"user":{"fname":"first","lname":"last"}},
"_links":{
"edit":{"href":"/api/user/id/1"},
"message":{"href":"/api/message/id/1"}
},"id":"1"}
```

A Good REST API Uses

- By sending back the next possible actions we are able to modify the endpoints without breaking backwards compatibility... Let's say we now required the last name in the message endpoint, just to prevent accidental message from being sent...

Use JSON instead of XML

- Better Language Support
- Lightweight (much less code than XML)

```
{"object1":{"object":{"string":"rock","string2":"
star"}},"object2":{"string":"hello","string2":
"world"},"string":"json is cool"}
```

- Object Oriented by nature
- Simple to encode/ decode
  - To encode/ decode JSON in PHP simply use the json_encode() and json_decode() functions! Many frameworks also have JSON libraries that take care of all of the error checking for you as well!

Use Authentication Tokens

- Users should be provided with a unique API key/ identifier that allows you to track, limit, and enable features
- An OAuth token should be used to link API keys to accounts instead of requesting account usernames and passwords

Throttle API Key Usage

- Be sure to place limits on the number of calls/ queries an API key can make per hour or per day to prevent your application from becoming bogged down and susceptible to DOS attacks.
- This risk can also be reduced by ensuring your API is operating within the cloud and has the ability to scale to increased traffic demands.

OAuth Tokens

- Tokens help prevent misuse of the system and limit access to the control panel. Usernames and passwords allow hackers to go in beyond the API, often accessing billing and profile information that exposes your clients to catastrophic damage should this information fall into the wrong hands

Use HTTP Status Codes

- Proper use of HTTP status codes give your users immediate feedback on the result. Some of the more widely used Header codes include:
  - 200–OK
  - 201 – Created
  - 304 – Not modified
  - 400 – Bad Request
  - 401 – Not Authorized
  - 403 – Forbidden
  - 404 – Page/ Resource Not Found
  - 405 – Method Not Allowed
  - 500 – Internal Server Error

Use Descriptive Error Messages

- Error messages should give a clear description of what went wrong and how the client can fix their code to avoid the error in the future.
- Providing a link to additional information and live debugging examples is often extremely useful to developers, and allows your error messages to be more compact, but requires you to keep additional documentation up to date for your users.

Use Descriptive Error Messages

- These are not good error messages:
  - Something went wrong
  - Could not complete aclon
  - Invalid parameters
  - Hey look, it’s a rocket.

Use Descriptive Error Messages

- Good Error Messages:
  - The Authentication token is not recognized, learn more at http://...
  - A User ID is required to perform this acIon, learn more at http://...
  - Error Code 313. Read more at http://...

Outline

- Web Services
- Mashups

What is a Mashup?

In Food	In Music	In Web
To crush a particular type of fruits or vegetables until it changes its shape and produce a new dish.	An audio / video recording that is a composite of samples from other recordings, usually from different musical styles.	Website or web application that combines content from more than one source into an integrated experience.
Cambridge	Dictionary.com	TheFreeDictionary

What is a Mashup?

- Definition : Wikipedia
  - Mashup is a Web application that combines data from two or more sources into a single integrated application.

- What's missing?
  - The integration/combination of different sources is not limited to data but also to functionality and layout styles.

- A Complete, Accurate definition is required.

Mashup Definition

- Mashup is a Web-based Application that is created by combining and processing on-line third party resources, that contribute with data, presentation or functionality.

- Note: Online third party resources refer to any type of resource available in the Internet.
  - API
  - Web feeds
  - Screen Scraping
  - Excel/PDF files etc

What is a Mashup?

In Food

Cambridge

In Music

U2 - Window In The Skies (latest album)

Dictionary.com

In Web

Digg.com

TheFreeDictionary

Housingmaps.com

- History
  - Creation of Paul Rademacher.
  - One of the first Mashups created.

- Whats being Combined?
  - List out houses, apartments, and rooms that are for sale or rent from Craigslist and displays them on a Google map.

- Housingmaps = Cragilist + Google Maps

www.housingmaps.com

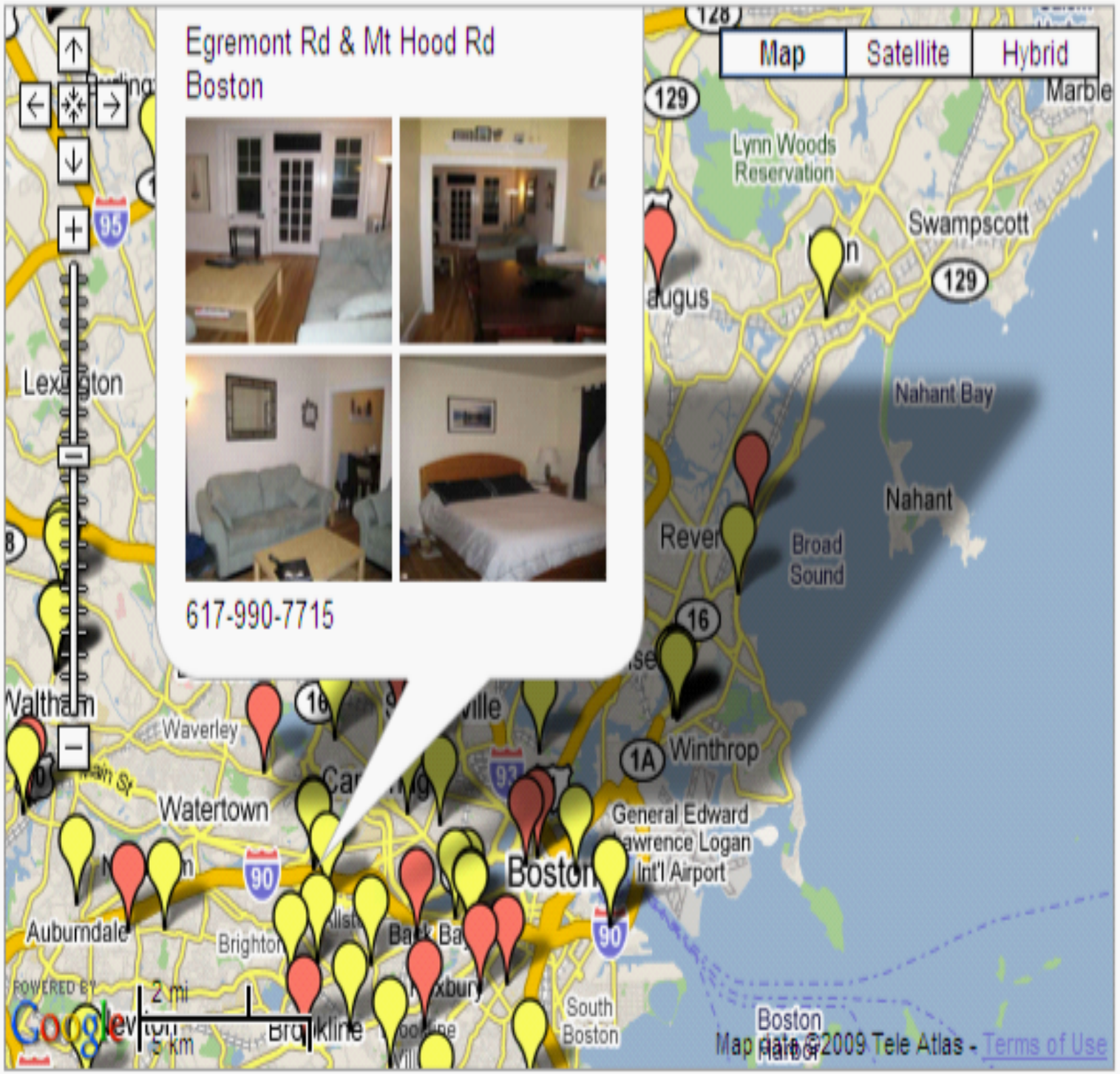
http://www.housingmaps.com/

For Rent For Sale Rooms Sublets

Powered by craigslist and Google Maps
(this site is in no way affiliated with craigslist or Google)

About / Feedback

City: Boston Price: $1500 - $2000 Show Filters[New] Refresh Link



pics	price	bd	description	city	date
	$1500	2bd	Charming 2-Bedroom	Arlington	5/31
	$1600	1bd	Spacious One Bedroom with balcony in well-maintained Prewar building	Brookline	5/31
	$1500	3bd	Small Upscale Home Steps To The Beach Year Round Or Seasonal Weekly~	18th St	5/30
	$1700	2bd	Heart of Cleveland Circle	Brighton	5/30
	$1500	3bd	1.5 bath Duplex	Framingham	5/30
	$1945	2bd	Beautiful and Spacious Two Bedroom Handicap Unit!	Quincy	5/30
	$1850	2bd	Showing Today 5:30-6:15* 10 mins from Harvard, pets Ok	Somerville	5/30
	$1600	2bd	BUMedical district sunny apt	Boston	5/30
	$1500	2bd	Beautiful 5 room, 2 bed w/hardwoods, laund,storage & pkg	Melrose	5/30
	$1600	1bd	Luxury without the Price! ***	Quincy	5/30
	$2000	4bd	4 Bedroom Apartment near T	Newton	5/30
	$1500	2bd	Luxury Loft, Near Lake, Avl Aug 1, No Fee	Wakefield	5/30

Why Mashups are Popular?

- Even non technical users can create Mashups without any programming knowledge.
- Explosive growth in “user-generated content”
- Wide deployment of XML web services.
- Increased broadband access.
- Wider conceptualization of the Internet as a platform (“Web 2.0”)
- Lot of user friendly tools are available.
  - Microsoft Popfly
  - Dapper

Classification

- Explored various Mashups available on ProgrammableWeb.
- Studied existing Mashup categorization models.
- They are:
  - Market Overview of Enterprise Mashup Tools: By Hoyer, V., and Fischer, M
  - A New Way of Providing Web Mapping and GIS Services.: By Li, S., and Gong, J
  - Is IBM making enterprise mashups respectable? ZDNet Blog 2006.

    - Available: http://blogs.zdnet.com/Hinchcliffe/?p=49&tag=nl.e622

Classification contd..

- Classified Mashups based on following four questions:
  1. What to Mash up?
  2. Where to Mash up?
  3. How to Mash up?
  4. For Whom to Mash up?.

1. What to Mash up?

- Depending on the sort of assets being **combined** or **integrated**, Mashups are assigned to one of the following three categories:

I. **Presentation Mashups:**
   - Focuses on retrieving information and layout of different Web sources without regarding the underlying data and application functionality.
   - The creation of a presentation mashup requires little or no knowledge of programming languages.
   - Example:
     - Pre-built widgets that can be drag and drop to common user interface

1. What to Mash up?

2. Data Mashups:

- Merges data provided by different sources into one content page.
  - The user mixes data from multiply sources and customizes the data flow of for example the Web page containing data from different sources.
  - Different sources are: Web Services, Feeds, and HTML etc.

3. Functionality Mashups

- Combines data and application functionality provided by different sources to a new service.
- The functionalities are accessible via APIs.

2. Where to Mash up?

- Mashups can be distinguished depending on the location where they are mashed up.
  1. Server-side Mashups integrate resources on the server.
  2. Client-side Mashups integrate resources on the client, often a browser.

- Usually a mixture of Client-side and Server-side applications is used for the creation of Mashups.

3. How to Mash up?

- Depending on the modality the resources are integrated or combined to one representation.
  - Extraction Mashup: data wrapper collecting and analyzing resources from different sources and merging the resources to one content page.
  - Flow Mashup: user customizes the resource flow of the Web page combining resources from different sources.

4. For Whom ?

- Target audience the Mashups are created for and addressed to:
  - Consumer Mashups: For public use
    - Combines resources (e.g., layout or data) from different public or private sources in the browser and organizes it through a simple browser based user interface.
  - Enterprise Mashups: For Business use
    - Merges multiple resources of systems in an enterprise environment. (e.g., data and application functionality).
    - Requires considering security, governance or enterprise policies.

Classification of Mashups

What?

- Presenation Mashups
- Data Mashups
- Functionality Mashups

How?

- Extraction Mashups
- Flow Mashups

Where?

- Server – side Mashups
- Client – side Mashups

For Whom?

- Consumer Mashups
- Enterprise Mashups

Tools

- Several tools have been published that provide functionalities for building, storing and publishing Mashups.
- The range of these Mashup tools spans from open- source tools to highly-cost license tools.
- Some of the vendors offer a coding editors.
- While others focus on users with no programming skills that provide easy-to-use access and application to their tool suites.

Tools contd..

- Some examples :
  - Yahoo Pipes
  - IBMMashupCenter
  - Intel Mashmaker
  - Microsoft Popfly
  - Derri Pipes
  - Dapper

- Evaluated how these tools can be used to create Mashups and were classified.

	Presentation	Data	Functionality	Extraction	Flow	Consumer	Enterprise
Apatar		×			×		×
Data Mashups	×	×			×		×
Dapper	×			×		×	
DERI pipes		×			×	×	
Grazr	×			×		×	
IBM InfoSphere MashupHub		×		×			×
Intel MashMaker	×				×	×	
JackBe Presto		×	×		×		×
Microsoft Popfly	×	×			×	×	
Openkapow	×			×		×	
Procession		×	×		×		×
Rssbus	×			×	×		×
Serena Mashup Suite		×	×		×		×
Snap Logic		×			×		×
TIBCO PageBus		×			×		×
Yahoo! Pipes		×			×	×	

Challenges

- Cataloguing.
  - Some Web pages are already available that list Mashups and provide an interface for searching of mashups such as programmableweb.com.
  - Mashup creators can insert their mashups in the list and share their Mashups with others.
  - But what is missing is a directory that stores and catalogues the mashups in a consistent way.
- Making Data Web enabled
  - Currently a lot of data and functionalities are not set up on the Web and they are not accessible via feeds, HTML or Web services.
  - To make more resources “Web-enabled” require formats and tools that facilitate an efficient access and connection of resources to the Web.

Challenges contd..

- Security and Identity
  - Requires mechanisms to control the user connection and the data security.
- Sharing and Re-using
  - Vendors of Mashup tools should provide mechanisms to allow end-users sharing their built Mashups with others to facilitate the reuse of pre-built Mashups.
  - Easy-to-use access to Mashups.
  - Efficient Mashup search functionalities lightweight formats that enable even for non-programmers a smooth Mashup reuse.

Challenges contd..

- Version Control Mechanisms
  - Mashups consist of different resources collected from various sources.
  - Resource owners are responsible for their content and can change and update its content whenever they regard it as necessary.
  - To keep the content up-to-date a version control mechanism is required that automatically informs the Mashup owner about updates.
- Trust Certificates
  - No certification mechanisms exist that guarantee end-users the trustworthiness of the Mashup.

Conclusion

- Mashups are suitable to build novel Web applications and to create new forms of visualization without little knowledge of programming languages.

- Further research is especially needed in the fields of version control mechanisms, Mashup certification, Mashup quality and data integrity.

Thanks!!!