HI!
MY NAME IS
Scotty
In 2007, Google and MySpace were worried by the launch of the Facebook Platform for developing apps, so they banded together and created the OpenSocial specification.
OpenSocial is an open specification – now governed by the OpenSocial Foundation. Anyone can implement the specification in their application, or develop gadgets and external services using the JavaScript or REST APIs
The effort was focussed on social networking platforms – so the APIs make available data about friends, activities, photos, videos, games, etc
While the spec was open, it wasn’t enough to stop the Facebook juggernaut
But then enterprises started paying attention to this thing called “Enterprise 2.0” – using social networking technologies to improve collaboration inside and between companies. Shockingly, many of the big vendors in the space jumped onto OpenSocial
The vendors realized that their customers had many information silos, and that the OpenSocial framework provided a mechanism for lightweight, loose integration.
So now there are efforts to add support for more enterprise data types to OpenSocial – business documents as well as the existing video, business relationships instead of everyone being friends, etc. There’s also an effort that’s looking at “eduSocial”
Time for some demos...
go to box.net, login as itlab.demo@gmail.com, and upload something to the demo folder
go to http://linkedin.com/ log in as itlab.demo@gmail.com; go to the More menu, select Get More Applications... and add Box.Net Files
Go to http://google.com/ig/ – log in as itlab.gmail@demo.com if needed, then click on Add Stuff, search for Box.net, and add the Box gadget with the blue box logo
Go to JIRA, login as demo, click add a new gadget, select box gadget, click finished, login to box as itlab.demo@gmail.com
So what have we done at Stanford? Where can we apply OpenSocial?
We were required to implement web based metrics and reporting. We wanted to use a dashboard model, and let users customize their views. We looked at Drupal with an OpenSocial plugin, but it didn’t provide a dashboard interface, and the plugin was not actively supported anymore. We went with Atlassian’s JIRA, and implemented every metric as an OpenSocial gadget.

http://dashboard.stanford.edu/
don’t panic – gadgets can be simple
<?xml version="1.0" encoding="UTF-8"?>

you need some XML
<!DOCTYPE html>

and some HTML
and some CSS (although we won’t use much here)
Add developer gadget
The obligatory Hello, World! example
<xml version="1.0" encoding="UTF-8" ?>
<Module>
    <ModulePrefs title="Hello World!"
        description="Very simple gadget" />

    <Content type="html">
        <![CDATA[
            <p>Hello, World!</p>
        ]]> 
    </Content>
</Module>

Friday, February 11, 2011

go to http://google.com/ig/ select add stuff, then add http://itlab.stanford.edu/gadgets/demo/helloworld.xml
Adding a user preference, so the user can get a customized experience

go to http://google.com/ig/ select add stuff, then add http://itlab.stanford.edu/gadgets/demo/helloyou.xml
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs
      title="Stanford"
      description="Stanford Home Page" />
  <Content
      type="url"
      href="http://stanford.edu/" />
</Module>

http://itlab.stanford.edu/gadgets/demo/stanford.xml

works, but it doesn’t look great
<moduleprefs
title="Blog"
description="ITArch Blog"
scrolling="true"
height="600" />

<content
type="url"
href="http://itarch.stanford.edu/blog/?wpmp_switcher=mobile" />

</module>
Let’s look a little closer at how gadgets work
So how does an OpenSocial container actually assemble a gadget? First, it returns the container HTML to the browser, with an iFrame to contain each gadget, pointing to the container server. For each gadget request, the container takes the HTML, CSS and JS from the gadget XML and combines those with container specific HTML, CSS and JS and serves that to the browser.
The logo’s quite informative – the container generates the main content of the browser window, but the contents of the gadgets either come from another server, or are proxied by the container. In both cases, the content is wrapper in an iFrame. iFrames are invisible frames that allow one page to embed the contents of another page. If the pages come from different servers, they can’t (easily) interact. Here’s a cool iFrame abuse. http://stanfordwho.stanford.edu/SWApp/detailAction.do?search=swl&key=DS590Q483
Let's look at more advanced gadgets
function hello(name) {
    alert("Hello, "+ name);
}
“Dispatch War Rocket Ajax to bring back his body”

Friday, February 11, 2011
The Flash Gordon screenwriter predicted Web 2.0 in 1980
function callback (obj) {
  ...
}

var url = "http://.../";

$.get(url, callback);

This is a simple AJAX call using the jQuery library. It makes a request to the URL, and if the request is successful, the callback function is called, and passed the data from the response. The URL protocol, hostname and port must be the same as those of the Javascript source – the same origin rule.
This complicates things for OpenSocial.
Remember this? For each html gadget request, the container takes the HTML, CSS and JS from the gadget XML and combines those with container specific HTML, CSS and JS and serves that to the browser. However, this means the browser sees the gadget as coming from the container, not the application (gadget) server. OpenSocial containers therefore have to support a proxy to get data from the application server to the gadget; thankfully, they provide an API to simplify that.
Here we set the requested content type to JSON, and again, callback will be called when the data is returned from the server.
What’s special about this page?

It’s the only page you’re supposed to enter your SUNetID and password on the web.

But the earlier gadgets were asking for username and password.
OAuth is the “valet key for the web” – it lets you delegate access to an application without giving it your username or password.

Note: this demo has a broken URL, which I’ll fix. It’s a misconfiguration on the other JIRA that I need to track down.

Add http://itarch.stanford.edu/gadgets/jira/activity-stream.xml to localhost demo
What about building your own container, or adding OpenSocial to an existing web application?
http://shindig.apache.org/
Image Credits

Open Sign: http://www.flickr.com/photos/chipgriffin/2210297177/
Skyscraper lunch: http://www.flickr.com/photos/53558245@N02/4978362207/
Shocked dolls: http://www.flickr.com/photos/prawnwarp/2939632457/
Demo graffiti: http://www.flickr.com/photos/m-j-s/158876717/
Pink code: http://www.flickr.com/photos/staciebee/4331093351/
Don’t Panic: http://www.flickr.com/photos/brighton/2153602543/
Hello sparkler: http://www.flickr.com/photos/major_clanger/382513/
Magnified eye: http://www.flickr.com/photos/chipgriffin/2210297177/
Difference Engine: http://www.flickr.com/photos/fimbrethil/137042644/
Ajax handle: http://www.flickr.com/photos/shelleygibb/3055821340/
OAuth logo: http://www.flickr.com/photos/factoryjoe/1529124811/
Dovetailed box: http://www.flickr.com/photos/windsordi/4365905418/

Vendor logos © those vendors
OpenSocial logos and diagrams from opensocial.org
Facebook screenshot from Facebook’s gallery

Friday, February 11, 2011
More Information

http://wiki.opensocial.org/
http://opensocial.org/
http://code.google.com/apis/opensocial/

http://shindig.apache.org/

http://oauth.net/

https://itarch.stanford.edu/confluence/display/OS/Home