OpenSocialGadgets

This tutorial will introduce you to gadgets and OpenSocial, and will walk you through the steps required to build a simple social gadget where you can display your friends and mutual friends. In addition, you will be introduced to some of the more advanced features of the OpenSocial API.

Gadget basics

At their core, social gadgets are XML files, sometimes known as gadget specifications. Here is a simple "Hello World" gadget (helloworld.xml), that illustrates the basic sections of a specification:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Module>
    <ModulePrefs title="Friends">
        <Require feature="opensocial-2.5"/>
    </ModulePrefs>
  <Content type="html">
     <![CDATA[
       Hello World !
     ]]>
  </Content>
</Module>
```

In the "Hello World" example, you can see several sections which control the features and design of the gadget.

- <Module> indicates that this XML file contains a gadget.
- <ModulePrefs> contains information about the gadget, and its author.
- <Require feature="opensocial-2.5" /> denotes a required feature of the gadget — in this case, the OpenSocial API (v2.5).
- <Content type="html"> indicates that the gadget's content type is HTML. This is the recommended content type for OpenSocial containers, but gadgets for other containers such as iGoogle support other content types.

- <![CDATA[a:...]]> contains the bulk of the gadget, including all of the HTML, CSS, and [JavaScript] (or references to such files). The content of this section should be treated like the content of the body tag on a generic HTML page.

Friend Gadget

In this section we will create a gadget that will list all of your current friends.

Inline JavaScript vs. external JavaScript

For small gadgets, it's often easier to include all the JavaScript calls for a gadget in the same XML file as the HTML. However, for larger gadgets, this can become cumbersome, so it can be helpful to offload JavaScript function definitions into a separate file.

```
function init() {
   social = new friendsWrapper();
   social.loadFriends();
}
gadgets.util.registerOnLoadHandler(init);
```

Now, of course, there needs to be a function to actually load the friend data. The following function creates a new data request object, then populates it with specific types of data that you'll need: the owner and the owner's friends. Notice that in order to request friends, the code constructs an IdSpec object. An IdSpec is used when you need to specify one or more people in a group (in this case, the owner's friends). Then, it sends the request to the server, and gives it the name of a function to call when the data is returned.

friendsWrapper.js

```
this.loadFriends = function(){

var req = opensocial.newDataRequest();
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.OWNER),'owner');
var ownerFriends = opensocial.newIdSpec({ "userId" : "OWNER", "groupId" : "FRIENDS" });
var opt_params = {};
opt_params[opensocial.DataRequest.PeopleRequestFields.MAX] = 100;
req.add(req.newFetchPeopleRequest(ownerFriends, opt_params),'ownerFriends');

req.send(displayFriends);
};
```

The callback function, displayFriends, will take the data that the server has returned, and display it on the page.

```
function displayFriends(data) {

var owner = data.get('owner').getData();
var ownerFriends = data.get('ownerFriends').getData();
html =new Array();
html.push('Owner Friends are’,' <br>');
html.push('<ul>');
ownerFriends.each(function(person) {
if (person.getId()) {
                html.push('<li>', person.getDisplayName(), '</li>');
        }
});
document.getElementById('friends').innerHTML = html.join('');
gadgets.window.adjustHeight();
}
```

Div elements have been inserted within the gadget specification as entry point for the new HTML.

```
<?xml version="1.0" encoding="UTF-8"?>
  <Module>
    <ModulePrefs title="Friends">
    <Require feature="opensocial-2.5"/>
  </ModulePrefs>
  <Content type="html">
  <![CDATA[
        <script type="text/javascript" src="friendsWrapper.js"></script>
        <script type="text/javascript">
         function init() {
         social = new friendsWrapper();
                 social.loadFriends();
         }
         gadgets.util.registerOnLoadHandler(init);
        </script>
        <div id='friends'/>
  ]]>
  </Content>
</Module>
```

Viewer Friends

If your gadget is viewed by your friend and you want the gadget to display their friends instead of yours, you can do so by getting a list fo viewer’s friends.

```
this.loadFriends = function(){

var req = opensocial.newDataRequest();
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.OWNER),'owner');
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.VIEWER),'viewer’);
var ownerFriends = opensocial.newIdSpec({ "userId" : "VIEWER", "groupId" : "FRIENDS" });
var opt_params = {};
opt_params[opensocial.DataRequest.PeopleRequestFields.MAX] = 100;
req.add(req.newFetchPeopleRequest(ownerFriends, opt_params),'ownerFriends');
req.add(req.newFetchPeopleRequest(viewerFriends, opt_params),'viewerFriends');

req.send(displayFriends);
};


function displayFriends(data) {

var owner = data.get('owner').getData();
var ownerFriends = data.get('ownerFriends').getData();
var viewer = data.get(viewer').getData();
var viewerFriends = data.get(viewerFriends').getData();
html =new Array();
html.push('Owner Friends are’,' <br>');
html.push('<ul>');
ownerFriends.each(function(person) {
if (person.getId()) {
                html.push('<li>', person.getDisplayName(), '</li>');
        }
});
html.push('Viewer Friends are’,' <br>');
html.push('<ul>');
ownerFriends.each(function(person) {
if (person.getId()) {
                html.push('<li>', person.getDisplayName(), '</li>');
        }
});
document.getElementById('friends').innerHTML = html.join('');
gadgets.window.adjustHeight();
}
```

Mutual Friends

Now lets say someone else is viewing your profile. In that case it would be nice if we can display the mutual friends between you and the person viewing your profile. To facilitate this, we make use of a filter attribute called IS_FRIENDS_WITH filter. In our case, the person who views the profile is the friend (viewer) and we would display the friends between the viewer and owner.

```
this.loadFriends = function() {
var req = opensocial.newDataRequest();
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.OWNER),'owner');
var ownerFriends = opensocial.newIdSpec({ "userId" : "OWNER", "groupId" : "FRIENDS" });
var opt_params = {};
opt_params[opensocial.DataRequest.PeopleRequestFields.MAX] = 100;
req.add(req.newFetchPeopleRequest(ownerFriends, opt_params),'ownerFriends');

var params = {};
params[opensocial.DataRequest.PeopleRequestFields.MAX] = 100;
// Usage of isFriendsWith filter to get mutual friends. filterValue should be set to the friend with whom
mutual friends is to be found.
params[opensocial.DataRequest.PeopleRequestFields.FILTER]=opensocial.DataRequest.FilterType.IS_FRIENDS_WITH;
params["filterValue"] = opensocial.IdSpec.PersonId.VIEWER;
req.add(req.newFetchPeopleRequest(ownerFriends, params),'mutualFriends');
req.send(displayFriends);
};

function
displayFriends(data) {

var owner = data.get('owner').getData();
var ownerFriends = data.get('ownerFriends').getData();
html =new Array();
html.push('Owner Friends are',' <br>');
html.push('<ul>');
ownerFriends.each(function(person) {
if (person.getId()) {
                html.push('<li>', person.getDisplayName(), '</li>');
        }
});
html.push('</ul>');

var mutualFriends = data.get('mutualFriends').getData();
html.push('Mutual Friends are',' <br>');
html.push('<ul>');
mutualFriends.each(function(person) {
if (person.getId()) {
                html.push('<li>', person.getDisplayName(), '</li>');
        }
});
html.push('</ul>');
}
document.getElementById('friends').innerHTML = html.join('');
gadgets.window.adjustHeight();
}
```

Congratulations, you've written an OpenSocial gadget. But, don't cut the party short—there's more things to learn as easy as this. Feel free to visit the below website for additional methods but some contents might be outdated.

http://docs.opensocial.org/display/OSREF/OpenSocial+Tutorial

Below is the opensocial specs and entire list of methods and APIs you can use with opensocial

http://opensocial-resources.googlecode.com/svn/spec/trunk/Social-Gadget.xml