Apache Shindig

v. 1.0

User Guide

The Apache Software Foundation **2012-03-11**

Table of Contents

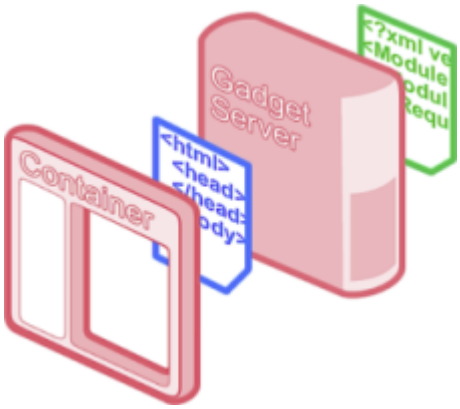
1 Introduction

1.1 Welcome To Apache Shindig !

Apache Shindig is an OpenSocial container and helps you to start hosting OpenSocial apps quickly by providing the code to render gadgets, proxy requests, and handle REST and RPC requests.

Apache Shindig's goal is to allow new sites to start hosting social apps in under an hour's worth of work.

1.2 What is Apache Shindig?



Apache Shindig is a container for hosting social application consisting of four parts:

- **Gadget Container JavaScript**: core JavaScript foundation for general gadget functionality (read more about gadget functionality). This JavaScript manages security, communication, UI layout, and feature extensions, such as the OpenSocial API.
- **Gadget Rendering Server**: used to render the gadget XML into JavaScript and HTML for the container to expose via the container JavaScript.
- **OpenSocial Container JavaScript**: JavaScript environment that sits on top of the Gadget Container JavaScript and provides OpenSocial specific functionality (profiles, friends, activities, datastore).
- **OpenSocial Data Server**: an implementation of the server interface to container-specific information, including the OpenSocial REST APIs, with clear extension points so others can connect it to their own backends.

Apache Shindig is the reference implementation of OpenSocial API specifications, versions 0.8.x and 0.9.x, a standard set of Social Network APIs which includes:

- Profiles
- Relationships
- Activities
- Shared applications
- Authentication

- Authorization

See the overview page for more information.

2 Download

2.1 Download Apache Shindig

Apache Shindig is distributed under the Apache License, version 2.0.

Apache Shindig is a distributed as API and as a Web application. You must have a Servlet container for the Java version or a Web server for the PHP version installed in order to proceed. Shindig was successfully tested on Jetty (Java), Tomcat (Java), Apache (PHP).

2.1.1 General Availability Releases

2.1.1.1 Current

Apache Shindig 2.0.0 was released the 2010-09-10.

http://www.apache.org/dist/shindig/2.0.0/

	Implementation	Download Link
Apache Shindig (tar.bz2)	Java	shindig-2.0.0-java.tar.bz2
Apache Shindig (tar.gz)	Java	shindig-2.0.0-java.tar.gz
Apache Shindig (zip)	Java	shindig-2.0.0-java.zip
Apache Shindig (war)	Java	shindig-server-2.0.0.war
Apache Shindig (tar.bz2)	PHP	shindig-2.0.0-php.tar.bz2
Apache Shindig (tar.gz)	PHP	shindig-2.0.0-php.tar.gz
Apache Shindig (zip)	PHP	shindig-2.0.0-php.zip
Apache Shindig (tar.bz2)	Source	shindig-2.0.0-source.tar.bz2
Apache Shindig (tar.gz)	Source	shindig-2.0.0-source.tar.gz
Apache Shindig (zip)	Source	shindig-2.0.0-source.zip

2.1.1.2 Old Releases

Apache Shindig 1.1 beta5 is available here.

Apache Shindig 1.0.1 is available here.

2.2 Documentation

You could also download the current documentation, i.e. this website, as a single jar file or as PDF file.

2.3 Integration Builds

Apache Shindig uses Hudson as Continuous Integration tool. You could download the latest build from the trunk (i.e. implementation of OpenSocial Spec 0.9) or the branch (i.e. implementation of OpenSocial Spec 0.8.1):

2.3.1 Trunk

http://hudson.zones.apache.org/hudson/job/Shindig%20Assembly/lastBuild/org.apache.shindig$shindig/

	Implementation	Download Link
Apache Shindig (tar.bz2)	Java	shindig-1.1-incubating-SNAPSHOT-java.tar.bz2
Apache Shindig (tar.gz)	Java	shindig-1.1-incubating-SNAPSHOT-java.tar.gz
Apache Shindig (zip)	Java	shindig-1.1-incubating-SNAPSHOT-java.zip
Apache Shindig (war)	Java	shindig-server-1.1-BETA6-incubating-SNAPSHOT.war
Apache Shindig (tar.bz2)	PHP	shindig-1.1-incubating-SNAPSHOT-php.tar.bz2
Apache Shindig (tar.gz)	PHP	shindig-1.1-incubating-SNAPSHOT-php.tar.gz
Apache Shindig (zip)	PHP	shindig-1.1-incubating-SNAPSHOT-php.zip
Apache Shindig (tar.bz2)	Source	shindig-1.1-incubating-SNAPSHOT-source.tar.bz2
Apache Shindig (tar.gz)	Source	shindig-1.1-incubating-SNAPSHOT-source.tar.gz
Apache Shindig (zip)	Source	shindig-1.1-incubating-SNAPSHOT-source.zip

2.3.2 Branch 1.0.x (0.8.1 spec)

http://hudson.zones.apache.org/hudson/job/Shindig%201.0.x%20branch%20Assembly/lastBuild/org.apache.shindig$shindig/

2.4 System Requirements

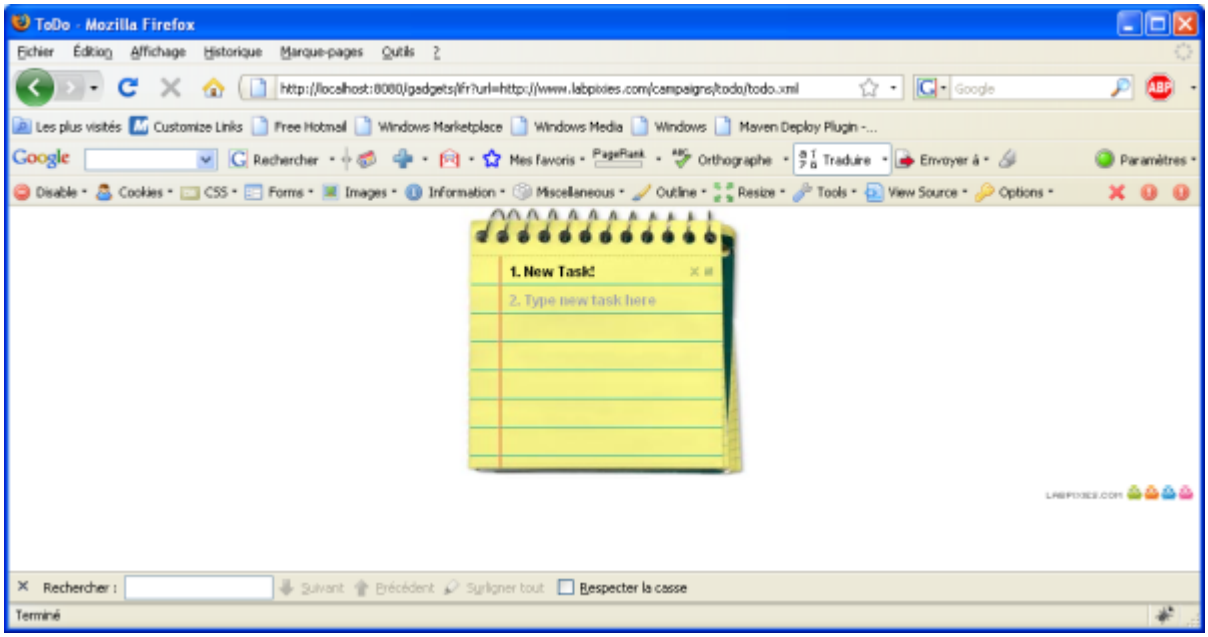
- **Java version**: Servlet container supporting Web Application 2.3 or above and JDK 1.5 or above
- **PHP version**: Apache Web Server with mod_rewrite enabled and PHP 5.2.x with the json, simplexml, mcrypt and curl extensions enabled.

2.5 Installation Instructions

Note: Read the README files inside the distributions for detailed instructions.

1. Unzip the distribution archive to the directory you wish to install Apache Shindig.
2. Install Apache Shindig in your Servlet container or in your Web server depending the wanted implementations. Have a glance to your server documentation before processing. For instance, you could do:
   A. Java version: copy the Apache Shindig WAR file to *$TOMCAT_HOME/webapps* for Tomcat.
   B. PHP version: copy shindig-php into */var/www/html/shindig* for Apache.

3. Open in you browser for instance http://localhost:8080/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml



Sample Container

3 Overview

3.1 Introduction

Apache Shindig, a word meaning party, was originally started by Google in 2007 as a reference container for hosting OpenSocial compatible widgets in any website. Originally a port of Google's iGoogle gadget container, with Brian McCallister's PHP code, Apache Shindig threw off it's egg shell, and showed that Google was serious about making OpenSocial accessible to a larger number of sites. Since December 2007, Apache Shindig is now an Apache project.

3.2 Apache Shindig's objectives

Apache Shindig's primary goal is to provide infrastructure for those wishing to host OpenSocial apps on their websites. Another goal of Apache Shindig is to be language neutral and cover multiple languages.

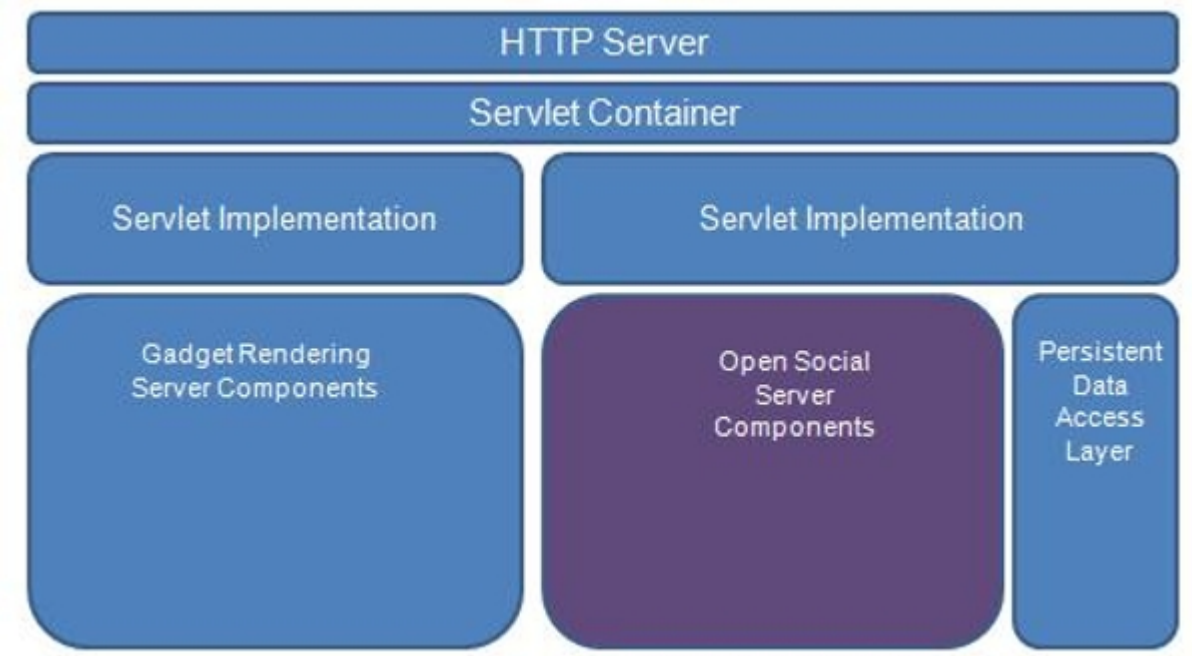
Currently, Java and PHP versions are availables and supported.

3.3 Apache Shindig Components

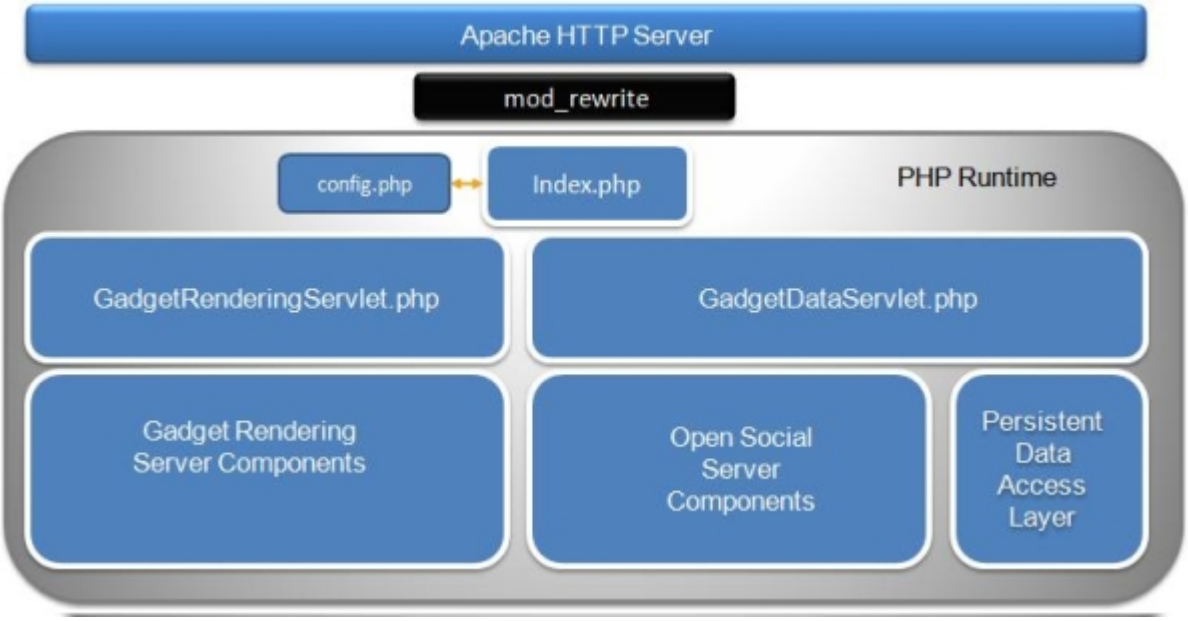
3.3.1 Server Side

The Java and PHP version of Apache Shindig have 3 major server side components:

1. Persistent Data Loading Mechanism;
2. Gadget Rendering Infrastructure;
3. OpenSocial server side implementation.



Components of Apache Shindig Java Server Side container

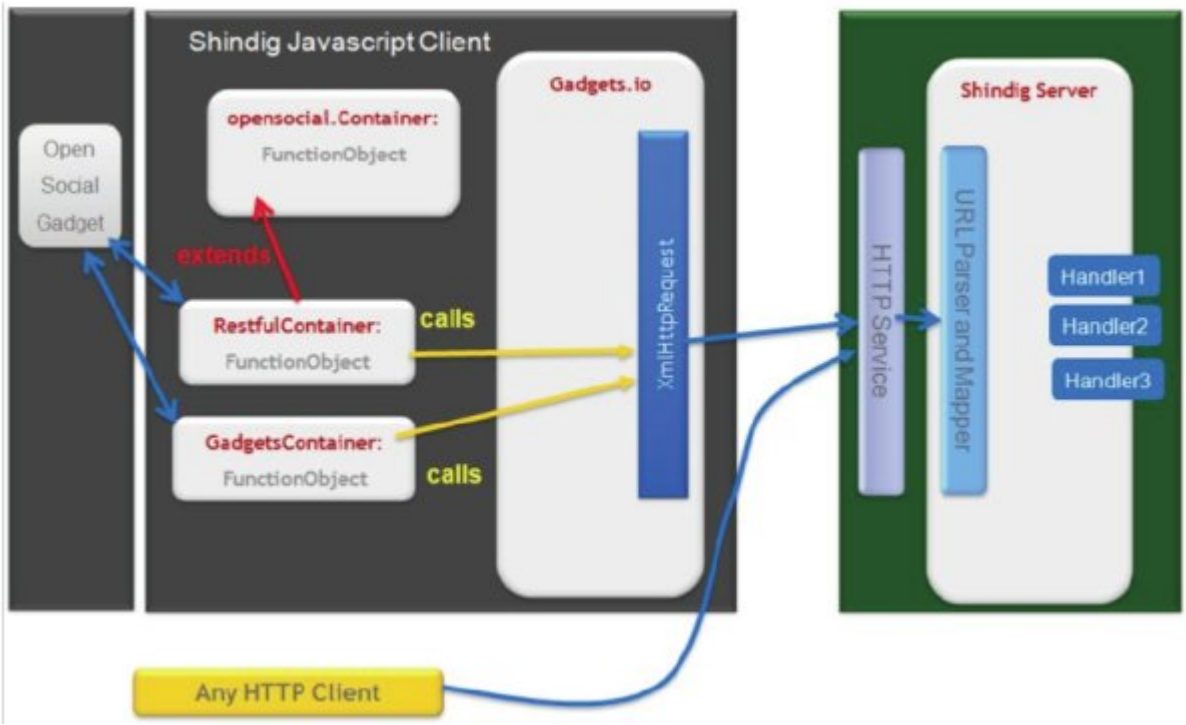


Components of Apache Shindig PHP Server Side container

3.3.2 Client Side

The Javascript features are:

- Gadget container (gadget.js), fully opensocial gadget compliant;
- OpenSocial container;
- JSON, Restful container and Caja support.



3.3.3 Put It Together: OpenSocial Flow

The following is a typical flow to get a list of Friends.

3.3.3.1 Server Side Flow

1. Call JsonRpcServlet;
2. Get the appropriate handler;
3. Get the JSON object from the DB;
4. Populate responses into a list;
5. Return to the client.

In the case of the REST use, the flow will be:

1. Call DataServiceServlet;
2. Get the appropriate converter;
3. Get the handler;
4. Get the JSON object from the DB;
5. Return to the client.

3.3.3.2 Client Side Flow

1. Create request object;
2. Populate request parameters;
3. Send the request.

3.4 OpenSocial APIs

Apache Shindig implements several OpenSocial APIs:

- **OpenSocial REST**: for server to server communication

- **OpenSocial JSON-RPC**: for gadget to server communication
- **Javascript**: for gadgets

3.4.1 OpenSocial REST

There are four types of REST services which can be exposed by Apache Shindig: *people*, *activities*, *appdata* and *groups*.There are URI Templates defined for each type of service.

For instance, http://localhost:8080/social/rest/people/john.doe

Type	Spec.	URI-Template

People	Spec.	**/social/rest/people/{guid}/@all** Collection of all people connected to user {guid} **/social/rest/people/{guid}/@all** Collection of all people connected to user {guid} **/social/rest/people/{guid}/@friends** Collection of all friends of user {guid}; subset of @all **/social/rest/people/{guid}/{groupid}** Collection of all people connected to user {guid} in group {groupid} **/social/rest/people/{guid}/@all/{pid}** Individual person record for a specific person known to {guid}; shows {guid}'s view of {pid}. **/social/rest/people/{guid}/@self** Profile record for user {guid} **/social/rest/people/@me/@self** Profile record for requestor

Activities	Spec.	**/social/rest/activities/{guid}/@self** Collection of activities generated by given user **/social/rest/activities/{guid}/@friends** Collection of activities for friends of the given user {guid} **/social/rest/activities/{guid}/{groupid}** Collection of activities for people in group {groupid} belonging to given user {uid} **/social/rest/activities/{guid}/@self/{activityid}** Individual activity resource; usually discovered from collection
Appdata	Spec.	**/social/rest/appdata/{guid}/@self/{appid}** All app data for user {guid}, app {appid} **/social/rest/appdata/{guid}/@friends/{appid}** All app data for friends of user {guid} and app {appid}; read-only (only GET and HEAD supported)} **/social/rest/appdata/{guid}/@self/{appid}?fields=count** Just the count field for user {guid}, app {appid}

Group	Spec.	**/social/rest/groups/{guid}** Collection of groups associated with user {guid} **/social/rest/groups/{guid}/{groupid}** Individual group {groupid} associated with user {guid}

3.4.2 OpenSocial JSON-RPC

Apache Shindig implements all required RPC services: *People*, *Activities*, *Appdata*, *Messages*, *Albums*, *MediaItems* and *System*.

Note: Apache Shindig doesn't implement all methods defined in OpenSocial RPC Protocol, Section 2, Services but these missing methods are not required to be OpenSocial compliant.

Type	Spec.	Service method
People	Spec.	**people.get** Retrieve a single person or list of opensocial.Person objects. **people.supportedFields** List the supported fields for this service.

Activities	Spec.	**activities.create** Support creating opensocial.Activity objects as the targets of a relationship with the specified user. **activities.get** Retrieve a one or list of opensocial.Activity objects. **activities.update** Support updating opensocial.Activity objects as the targets of a relationship with the specified user. **activities.delete** Support removing the relationship between an opensocial.Activity and the specified user. **activities.supportedFields** List the supported fields for this service.

Appdata	Spec.	**appdata.create** Support creating key-value pairs in a user appdata. **Not defined in the spec.** **appdata.get** Retrieve a map of key-value pairs for the list of specified keys. **appdata.update** Add or replace key-value pairs stored in a users appdata with the key-vaues in the data parameter. **appdata.delete** Remove the specifed keys from a users appdata and returned the values associated with those removed keys.
Messages	Spec.	**messages.create** Support creating opensocial.Message objects. **Not defined in the spec.** **messages.get** etrieve a one or list of opensocial.Message objects. **Not defined in the spec.** **messages.modify** Support updating opensocial.ActivityMessage objects. **Not defined in the spec.** **messages.delete** Support removing opensocial.ActivityMessage objects. **Not defined in the spec.**

System	Spec.	**system.listMethods** Returns an array of all methods supported by the endpoint including the system methods. **messages.methodSignatures** Returns a method signature describing the types of the parameters. **Not implemented yet.** **messages.methodHelp** Returns a textual description of the operation identified by the methodName parameter. **Not implemented yet.**

3.4.3 OpenSocial JavaScript

Apache Shindig includes all JavaScript APIs as described in OpenSocial API Reference and Gadgets API Reference.

See also the JavascriptDoc for Apache Shindig 1.1.x.

3.5 Resources

- Apache Shindig : An Architectural Overview (PHP Version) by Rajdeep Dua.
- Architectural Overview of Apache Shindig , an OpenSocial Reference Implementation by Rajdeep Dua.
- Overview of REST Implementation in Apache Shindig - Java Version by Rajdeep Dua.
- OpenSocial specifications

4 Getting Started

4.1 Getting Started

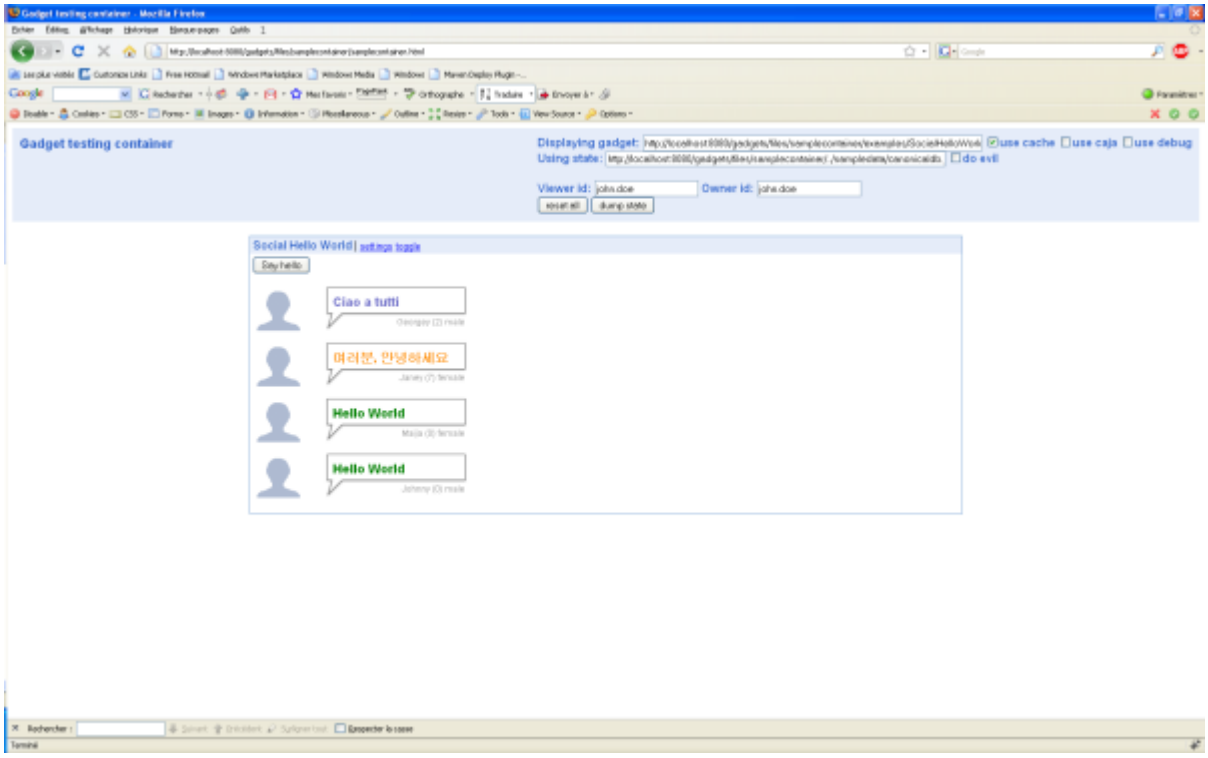
So, you correctly downloaded and installed Apache Shindig, what's next?

4.2 Discovering Apache Shindig Samples

The sample container should be your first starting point:

http://localhost:8080/gadgets/files/samplecontainer/samplecontainer.html

The page displayed in the figure bellow is the Apache Shindig sample container. Its goal is to help you to deploy and test your applications. You could also change the application state: changing the number and kind of friends, application data, or changing the viewer (i.e. the user accessing the application).



Sample Container

By default, the gadget is http://localhost:8080/gadgets/files/samplecontainer/examples/SocialHelloWorld.xml. You could try to use another gadget by specifying one of following URLs in the *"Displaying gadget"* field and click on *"reset all"*:

- http://localhost:8080/gadgets/files/samplecontainer/examples/SocialCajaWorld.xml
- http://localhost:8080/gadgets/files/samplecontainer/examples/SocialHelloWorld.xml
- http://localhost:8080/gadgets/files/samplecontainer/examples/getFriendsHasApp.xml

By default, all the datas are in the http://localhost:8080/gadgets/files/samplecontainer/../sampledata/canonicaldb.json. You could try to use another state by specifying one of following URLs in the *"Using state"* field and click on *"reset all"*:

- http://localhost:8080/gadgets/files/samplecontainer/state-basicfriendlist.xml
- http://localhost:8080/gadgets/files/samplecontainer/state-smallfriendlist.xml

You could also discover other samples on http://localhost:8080:

- http://localhost:8080/gadgets/files/container/sample1.html
- http://localhost:8080/gadgets/files/container/sample2.html
- http://localhost:8080/gadgets/files/container/sample3.html
- http://localhost:8080/gadgets/files/container/sample4.html
- http://localhost:8080/gadgets/files/container/sample5.html
- http://localhost:8080/gadgets/files/container/sample6.html
- http://localhost:8080/gadgets/files/container/sample7.html
- http://localhost:8080/gadgets/files/container/sample-metadata.html
- http://localhost:8080/gadgets/files/container/rpctest_gadget.html
- http://localhost:8080/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml

4.3 Playing With Apache Shindig

This part is mainly to play with OpenSocial APIs.

4.3.1 OpenSocial REST

Apache Shindig implements the OpenSocial Restful Protocol based on AtomPub. It supports XML and JSON formats in addition to Atom format XML. The following REST samples will focus on the Read only operation (GET). All sample data are located in http://localhost:8080/gadgets/files/sampledata/canonicaldb.json.

- Getting all people connected to *john.doe*:
  - URL: http://localhost:8080/social/rest/people/john.doe/@all
  - Result:

```
{"entry":[{"id":"jane.doe","name":{"formatted":"Jane Doe","givenName":"Jane"
{"id":"george.doe","name":{"formatted":"George Doe","givenName":"George","fa
{"id":"maija.m","name":{"formatted":"Maija Meikäläinen","givenName":"Maija",
```

- Getting the profile of *john.doe*:
  - URL: http://localhost:8080/social/rest/people/john.doe/@self
  - Result:

```
{"entry":{"id":"john.doe","name":{"formatted":"John Doe","givenName":"John",
```

- Getting all the friends of *jane.doe*:
  - URL: http://localhost:8080/social/rest/people/jane.doe/@friends
  - Result:

```
{"entry":[{"id":"john.doe","name":{"formatted":"John Doe","givenName":"John"
```

- Getting the activities of *john.doe*:
  - URL: http://localhost:8080/social/rest/activities/john.doe/@self
  - Result:

```
{"entry":[{"title":"yellow","userId":"john.doe","id":"1","body":"what a colo
```

- Getting the activities of *john.doe* using a **security token**:

- URL: http://localhost:8080/social/rest/people/jane.doe/@friends?st=a:a:a:a:a:a:a
- Result:

```
{"entry":[{"title":"yellow","userId":"john.doe","id":"1","body":"what a colo
```

For more information on the URI templates, please read OpenSocial REST section in the overview page.

4.3.2 OpenSocial JSON-RPC

Apache Shindig also implements OpenSocial RPC Protocol. The following RPC samples are similar to the REST samples.

- Getting the profile of *john.doe*:
  - URL: http://localhost:8080/social/rpc?method=people.get&id=myself&params.userId=john.doe&params.groupId=@self
  - Result:

```
{"data":{"id":"john.doe","name":{"formatted":"John Doe","givenName":"John","
```

- Getting all the friends of *jane.doe*:
  - URL: http://localhost:8080/social/rpc?method=people.get&id=myself&params.userId=john.doe&params.groupId=@all
  - Result:

```
{"data":{"list":[{"id":"jane.doe","name":{"formatted":"Jane Doe","givenName"
{"id":"george.doe","name":{"formatted":"George Doe","givenName":"George","fa
{"id":"maija.m","name":{"formatted":"Maija Meikäläinen","givenName":"Maija",
```

For more information on the URI templates, please read OpenSocial JSON-RPC section in the overview page.

4.4 Go Further: Building a Container Supporting OpenSocial

You are now ready to create your first OpenSocial application!

4.4.1 Create Your First OpenSocial Gadget

Apache Shindig implements Gadgets Specification. Gadgets are web-based software components based on HTML, CSS, and JavaScript. Please read Getting Started: gadgets.* API to learn more.

Typically, an Opensocial gadget is similar to a Google Gadget since its should respect the Google Gadgets specifications. Here is a small snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="A Title">
    <Require feature="opensocial-0.8"/>
    ...
  </ModulePrefs>
  <Content type="html">
     <![CDATA[
     <script type="text/javascript">
       ...
       gadgets.util.registerOnLoadHandler(init);
     </script>
     <div id="id"/>
     ...
     ]]>
  </Content>
</Module>
```

This XML file describes the OpenSocial application.

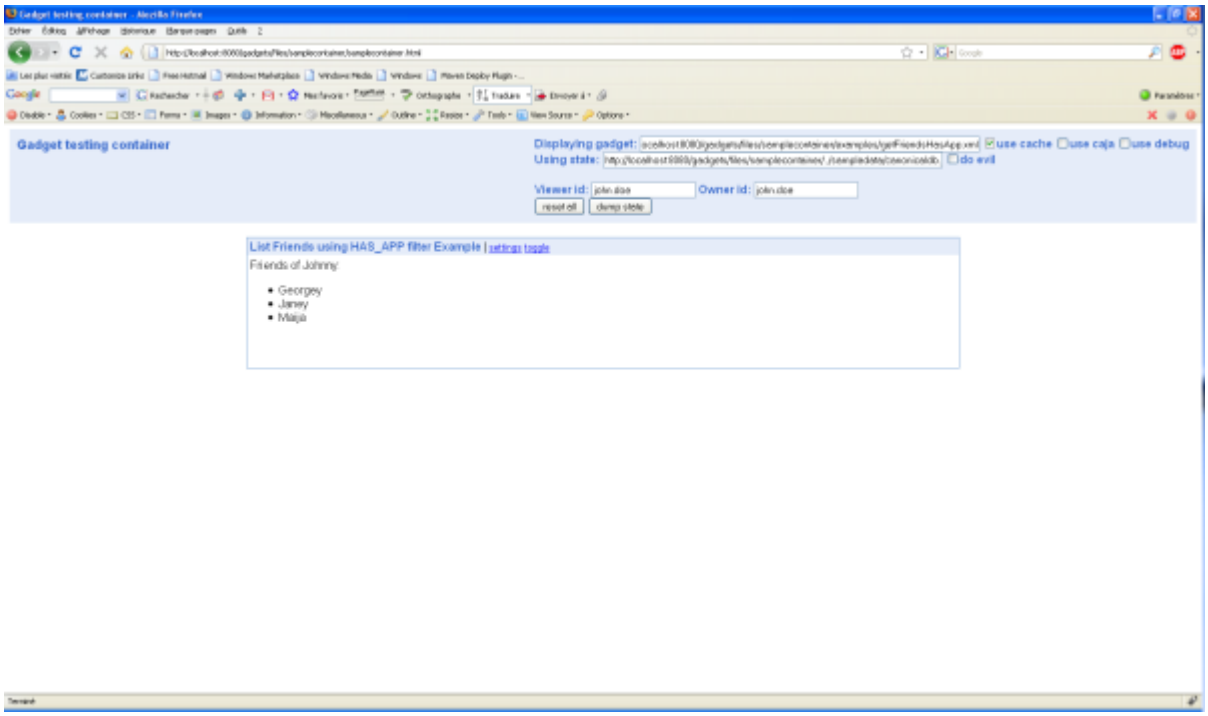
- The *ModulePrefs* section contains application metadata, such as title and required libraries. Also, you would notice the *require* element which includes the OpenSocial APIs as feature. Always declaring this requirement make your applications as an OpenSocial application. Others features supported by Apache Shindig are described here.
- The application contents live within the *Content* section. You should include additional JavaScript that contains the application logic and an handler to fire an *init()* function at launch time with the *gadgets.util.registerOnLoadHandler()* function. Finally, you will include an HTML section to display the result.

To access data from the Apache Shindig, you have to create a *DataRequest* object using the *opensocial.newDataRequest()* call.

Your first application will enumerate the friends currently belonging to the network of the user accessing the application. The following snippet is the code of http://localhost:8080/gadgets/files/samplecontainer/examples/getFriendsHasApp.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="List Friends using HAS_APP filter Example">
    <Require feature="opensocial-0.7"/>
  </ModulePrefs>
  <Content type="html">
  <![CDATA[
    <script type="text/javascript">
      /**
       * Request for friend information.
       */
      function getData() {
        var req = opensocial.newDataRequest();

        req.add(req.newFetchPersonRequest(opensocial.DataRequest.PersonId.OWNER), '
        var params = {};
        params[opensocial.DataRequest.PeopleRequestFields.MAX] = 50;
        params[opensocial.DataRequest.PeopleRequestFields.FILTER] = opensocial.Data
        params[opensocial.DataRequest.PeopleRequestFields.SORT_ORDER] = opensocial.
        req.add(req.newFetchPeopleRequest(opensocial.DataRequest.Group.OWNER_FRIEND
        req.send(onLoadFriends);
      };

      /**
       * Parses the response to the friend information request and generates
       * html to list the friends along with their display name.
       *
       * @param {Object} dataResponse Friend information that was requested.
       */
      function onLoadFriends(dataResponse) {
        var owner = dataResponse.get('owner').getData();
        var html = 'Friends of ' + owner.getDisplayName();
        html += ':<br><ul>';
        var ownerFriends = dataResponse.get('ownerFriends').getData();
        ownerFriends.each(function(person) {
          html += '<li>' + person.getDisplayName() + '</li>';
        });
        html += '</ul>';
        document.getElementById('message').innerHTML = html;
      };

      gadgets.util.registerOnLoadHandler(getData);
    </script>
    <div id="message"> </div>
  ]]>
  </Content>
</Module>
```

The following is the result of http://localhost:8080/gadgets/files/samplecontainer/examples/getFriendsHasApp.xml in the sample container http://localhost:8080/gadgets/files/samplecontainer/samplecontainer.html:



Get Friends

This is your first overview of the APIs that deals with people and relationships.

To go further, you could also try the opensocialdevapp http://osda.appspot.com/gadget/osda-0.8.xml, for instance:

http://localhost:8080/gadgets/ifr?url=http://osda.appspot.com/gadget/osda-0.8.xml&view=canvas

4.4.2 Create Your Own OpenSocial Back-end

Apache Shindig implements Opensocial Specification.

Typically, you need to implement some classes: *PersonService*, *AppDataService*, *ActivityService*, *MessagesService*, *AlbumService*, *GroupService*, *MediaItemService*.

In Java, theses classes are located in the org.apache.shindig.social.opensocial.spi package. Apache Shindig proposes a sample JPA implementation. Read Samples for Java for more information.

In PHP, theses classes are located in the social/opensocial/spi dir.

4.4.3 Resources

- OpenSocial Tutorial

5 Documentation Centre

5.1 Documentation Centre

This documentation centre is for developer who wants to use Apache Shindig. Apache Shindig is developped in two languages: Java and PHP and you need to read the specific documentation depending the language wanted. Also, you could read about which features Apache Shindig uses.

- Java
- PHP
- Features

5.2 Developer Tools

Here are some useful developer OpenSocial tools.

- OpenSocial Development Environment(OSDE)

5.3 Resources

- Apache Shindig resources
- Articles & Tutorials from OpenSocial Foundation Wiki

6 Java

6.1 Documentation Centre for Java Apache Shindig

This documentation centre is for developer who wants to work with the Java version of Apache Shindig.

- Building Java Apache Shindig
- Using samples
- Javadoc for 1.1.x
- Documentation for 1.1.x
- Javadoc for 1.0.x
- Documentation for 1.0.x

6.2 Java Tools

- OpenSocial RESTful Java client library

7 Building Java

7.1 Building and running Apache Shindig for Java

This is the Java steps on how to build and run Apache Shindig.

7.1.1 Prequisites before building Apache Shindig

In order to build Apache Shindig, you must have the following:

- Java (JDK/JRE) 1.5 or later installed on your system and the JAVA_HOME environment variable set.
  - See: http://java.sun.com/ for installation instructions.
- A Subversion client installed in order to checkout the code.
  - Instructions for downloading and installing Subversion can be found here: http://subversion.tigris.org/
- Apache Maven installed to perform the build.
  - Instructions for downloading and installing Maven can be found here: http://maven.apache.org/download.html

7.1.2 Getting the code

Create a subdirectory and checkout the Apache Shindig code from its Subversion repository

1. `mkdir ~/src/shindig` (or wherever you'd like to put it)
2. `cd ~/src/shindig`
3. `svn co http://svn.apache.org/repos/asf/shindig/trunk/ .`

7.1.3 Building and running the code with Maven

To build a Web Archive (WAR) file for the Gadget server and run tests, perform the following:

1. Make sure you have the prerequisites installed first.
2. `cd ~/src/shindig/`
3. `mvn`
4. Once the build successfully completes, you can install the built WAR files located in the /target subdirectory onto your JEE server.

To run the code and start a Jetty server that will run on at localhost:8080:

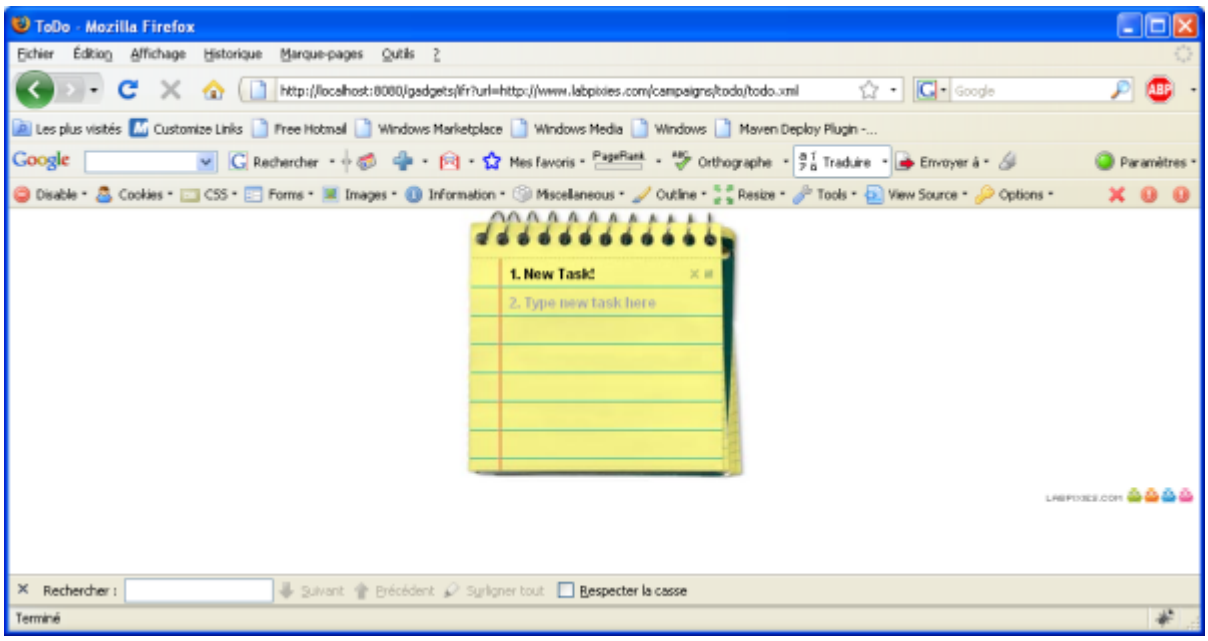
- `mvn -Prun`
- Open in you browser http://localhost:8080/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml

To run the Jetty server on a different port, use:

- `cd java/server`
- `mvn clean install jetty:run -DrunType=<full|gadgets|social> -Djetty.port=<port>`

Once you've either installed the WAR file on your JEE server, or are running locally using the Jetty server, you can test the Gadget server using:

- http://localhost:<port>/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml



7.1.4 Setting up an Eclipse project to build Apache Shindig

These steps, after completing the previous section, will allow you to build from within Eclipse using the Maven2 plugin. You should first install the Maven plugin, then create the new Java project.

- Create `~/.m2/settings.xml` consisting solely of

```
<settings> </settings>
```

- Install the Maven2 plugin
  1. Help -> Software Updates -> Find and Install
  2. Search for new features to install
  3. Create a new remote update site for the Maven 2 plugin
     4. Name: Maven2 - Sonatype
     5. URL: `http://m2eclipse.sonatype.org/update/`
  6. Select the site and click "Finish"
  7. There are optional dependencies on mylyn and subclipse. If you don't have these plugins, you can get them here. Otherwise, select only the Maven Integration plug-in.
  8. Complete the installation
- Setup new workspace and project
  Creating a new workspace eliminates the performance cost from existing projects and makes it easier to manage the code.

1. File -> Switch Workspace -> Other...
2. Select directory to store workspace
    3. Do not select a parent directory of the Apache Shindig source (e.g. ~/src/shindig) as Eclipse won't allow you to create the Java project.
    4. Something like `~/eclipse/workspaces/shindig` would work fine
5. File -> New -> Java Project
    1. Name the project. The instructions below will assume "SHINDIG".
    2. Select 'Create project from existing source' and navigate to `.../src/shindig/java`
    3. Click Finish
    4. If you see a dialog for "Open Associated Perspective", click Ok. Don't worry about the errors after loading as they will be fixed in the next step.
6. Right-click the project, select `Maven : Enable Dependency Management`
7. Right-click the project, select `Maven : Update Source Folders`
8. Optionally, if you would like to be able to browse or step into the code of your dependent jars when debugging, you need the source jars. Right-click the project, select `Maven : Download Sources` and Eclipse will automatically know about these sources when debugging. You can browse them under `Maven Dependencies` in your project.
9. If you'll be using AllTests to run tests or generate code coverage stats, adjust the project's output folders.
    1. Project -> Properties -> Java Build Path -> Source
    2. Locate and open `SHINDIG/gadgets/src/test/java`
    3. Select `Output Folder: (Default Output Folder)` and click Edit...
    4. Select `Specific Output Folder`
    5. Enter `target/test-classes` and click OK.
    6. Repeat for `SHINDIG/social-api/src/test/java`

7.1.5 Generating Code Coverage in Eclipse

To generate code coverage statistics inside of Eclipse, install the EclEmma plugin. Then

- Open `org.apache.shindig.gadgets.AllTests`
- Right-click in the class, and select `Coverage as -> JUnit Test`

7.1.6 Running inside Eclipse

To debug the server in Eclipse, follow the last two steps here (takes a few minutes to set up):

- "Using eclipse external tools"
- "Attaching to the server running in debug mode, using eclipse"

Note: You must have set up Eclipse to build the code or do `mvn package` yourself after making changes, but you won't need to restart Jetty to see your changes.

7.1.7 Running with Caja

Caja is an important part of OpenSocial that greatly enhances JavaScript security. Caja is managed in a separate open source project hosted by Google code projects. For more information on Caja, see: http://code.google.com/p/google-caja/wiki/CajaEasyIntro

1. Load this page: http://localhost:8080/gadgets/files/samplecontainer/samplecontainer.html
2. Point it to this gadget: http://localhost:8080/gadgets/files/samplecontainer/examples/SocialHelloWorld.xml

To see the cajoled code (Firefox only), right-click inside the iframe and do "This Frame -> View Frame Source"

7.1.8 Additional reading

Read java/README for original instructions on how to start up any of the java shindig servers.

Read javascript/README for instructions for using the Apache Shindig Gadget Container JavaScript to enable your page to render Gadgets using gmodules.com or a server started up as described above.

8 Samples

8.1 Samples for Java

There is a JPA implementation of SPI in our SVN /java/samples

You could also have a look to the SHINDIG-204 issue which provides an other (old) implementation and the Socialsite projectwhich implements the persistence for social data via JPA.

8.2 Resources

- Make Your Site an OpenSocial Container Using Shindig
- Shindig Spring Example

9 PHP

9.1 Documentation Centre for PHP Apache Shindig

This documentation centre is for developer who wants to work with the PHP version of Apache Shindig.

- Building PHP Apache Shindig

9.2 PHP Tools

- OpenSocial RESTful PHP client library
- Open Social Module for phpFox

10 Building PHP

10.1 Building and running Apache Shindig for PHP

This is the PHP steps on how to build and run Apache Shindig.

10.1.1 Prequisites before building Apache Shindig for PHP

In order to build and run Apache Shindig for PHP, you must have the following:

- A Subversion client installed in order to checkout the code.
  - Instructions for downloading and installing Subversion can be found here: http://subversion.tigris.org/
- Apache with mod_rewrite enabled.
  - PHP 5.2.x with the json, simplexml, mcrypt and curl extentions enabled.

10.1.2 Getting the code

Create a subdirectory in your web document root, e.g. /var/www/html and checkout the Apache Shindig code from its Subversion repository

1. `mkdir /var/www/html/shindig`
2. `cd /var/www/html/shindig`
3. `svn co http://svn.apache.org/repos/asf/shindig/trunk/ .`

10.1.3 Running Apache Shindig

With PHP There is no need to build anything - the source code is already built.

To run the code, you have several options:

10.1.3.1 a. Create a new virtual host

Point your apache to the php dir with a virtual host like:

```
<VirtualHost your_ip:your_port>

  ServerName your.host
  DocumentRoot /var/www/html/shindig/php
  ... other normal settings in vhosts...
</VirtualHost>
```

Restart apache, and point your browser to:

http://your.host/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml

you should see something like this.

10.1.3.2 b. Run with an existing host

If you cannot/don't want to create a virtual host, you can edit the file php/config.php and change the web_prefix setting to '/shindig/php'.

Then you can run the gadget by pointing your browser to:

http://your.host/shindig/php/gadgets/ifr?url=http://www.labpixies.com/campaigns/todo/todo.xml

10.1.4 Additional reading

Read php/README for original instructions on how to start up the php Apache Shindig server.

11 Features

11.1 Documentation Centre for the Apache Shindig's Features

This documentation centre is for developer who wants to use the Apache Shindig's features.

- Features List
- Using Features
- Create Your Own Feature
- JavascriptDoc for 1.1.x

11.2 Apache Shindig Features

Apache Shindig 1.1.x supports several features.

Feature	Description
analytics/feature.xml	Google Analytics
auth-refresh/feature.xml	To refresh the gadget security token
caja/feature.xml	Caja support
core/feature.xml	Core feature
core.io/feature.xml	Core IO feature to provide remote content retrieval facilities.
dynamic-height/feature.xml	To augment gadgets.window with functionality to change the height of a gadget dynamically.
flash/feature.xml	To embed Flash content into gadgets.
locked-domain/feature.xml	Locked domain.
minimessage/feature.xml	To create small dismissible messages in gadgets.
oauthpopup/feature.xml	To assist with management of the OAuth popup window.
opensocial-0.6/feature.xml	Opensocial-0.6.
opensocial-0.7/feature.xml	Opensocial-0.7.
opensocial-0.8/feature.xml	Opensocial-0.8.
opensocial-base/feature.xml	Opensocial base.
opensocial-current/feature.xml	Opensocial current.
opensocial-data/feature.xml	Opensocial data.
opensocial-data-context/feature.xml	Opensocial data-context.
opensocial-jsonrpc/feature.xml	Opensocial jsonrpc.
opensocial-reference/feature.xml	Opensocial reference.
opensocial-rest/feature.xml	Opensocial rest.
opensocial-templates/feature.xml	Opensocial templates.
osapi/feature.xml	Opensocial API.
pubsub/feature.xml	Gadget-side PubSub library for gadget-to-gadget communication.
rpc/feature.xml	RPC support.

setprefs/feature.xml	To augment gadgets.Prefs with functionality to store prefs dynamically.
settitle/feature.xml	To augment gadets.window with functionality to set the title of a gadget dynamically.
skins/feature.xml	Provide operations for getting display information about the currently shown skin.
tabs/feature.xml	Tab suuport.
views/feature.xml	Gadgets.views API spec
xmlutil/feature.xml	Provide xml utilities.

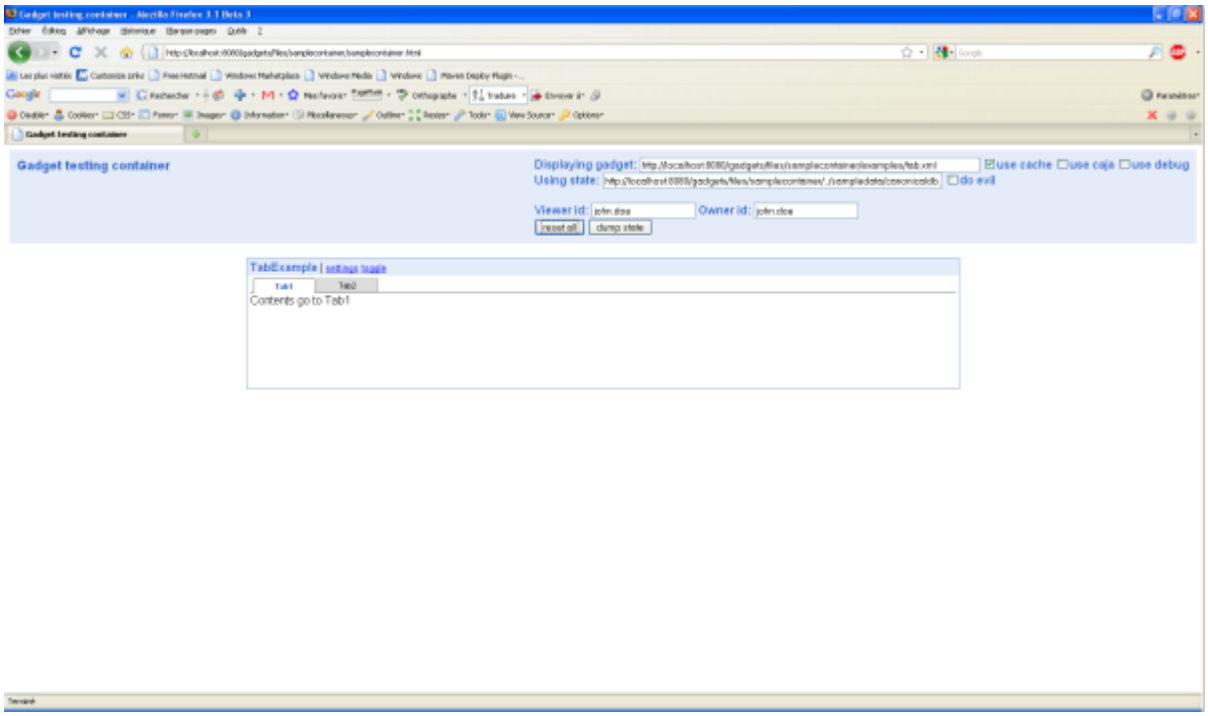
11.3 Using Features

To use an above feature in your gadget, you need to specify the feature as *<require/>* in *<ModulePrefs/>* and defining some Javascript codes to use it. Refer to the JavascriptDoc for 1.1.x to write your own Javascript.

For example, if you plan to use the tabs/feature.xml, your gadget will look like the following:

```
<Module>
  <ModulePrefs title="TabExample" description="Tabs Example">
    <Require feature="tabs"></Require>
  </ModulePrefs>
  <Content type="html">
    <![CDATA[
    <script type="text/javascript">
      function onLoad(){
        var tabset=new gadgets.TabSet();
        tabset.alignTabs('left');

        tabset.addTab("Tab1",{contentContainer:document.getElementById("tab1")});

        tabset.addTab("Tab2",{contentContainer:document.getElementById("tab2")});
      }

      gadgets.util.registerOnLoadHandler(onLoad);
    </script>

    <div id="tab1">
      Contents go to Tab1
    </div>

    <div id="tab2">
      Contents go to Tab2
    </div>
    ]]>
  </Content>
</Module>
```

You could test your code using the Gadget testing container at: http://localhost:8080/gadgets/files/samplecontainer/samplecontainer.html



11.4 Create Your Own Feature

TODO, see How to Contribute page

11.5 Resources

- The opensocial-resources project
- The gadget-resources project

12 Community Overview

12.1 The Apache Shindig Community

Apache Shindig, like any other opensource project, relies heavily on the efforts of the entire user community to be ever vigilant for improvements, logging of defects, communicating use-cases, generating documentation, and being wary of other users in need. This is a quick guide outlining what members of the Apache Shindig community may do to make the system work better for everyone.

12.1.1 Helping With Apache Shindig

12.1.1.1 Why Would I Want to Help?

There are several reasons these are good things.

- By answering other people's questions, you can learn more for yourself
- By submitting your own fixes, they get incorporated faster
- By reporting issues, you ensure that bugs don't get missed, or forgotten
- You are giving back to a community that has given you software for free

12.1.1.2 How to contribute?

As with any open source project, there are several ways you can help:

- Join the mailing list and answer other user's questions
- Report bugs, feature requests and other issues in the issue tracking application.
- Build Java Apache Shindig or Build PHP Apache Shindig for yourself, in order to fix bugs.
- Submit patches to reported issues (both those you find, or that others have filed)
- Help with the documentation by pointing out areas that are lacking or unclear, and if you are so inclined, submitting patches to correct it. You can quickly contribute rough thoughts to the wiki, or you can volunteer to help collate and organize information that is already there.

Your participation in the community is much appreciated!

12.1.1.3 Creating and submitting a patch

Note: Be sure to respect our Java code style! You could also read this thread http://markmail.org/message/t6b7jgzvfjj4fiqu.

For new people to the project, creating patches is the way to get started and build your reputation.

1. Start editing the code, since Subversion is being used no checkout is needed
2. Move to top level folder, e.g. cd ~/src/shindig
3. Generate diffs using svn di > fix-xxx-bug.patch
4. If needed, remove from the patch file any changes you do not want to submit until later
5. If you're new, create an account on https://issues.apache.org/jira/browse/SHINDIG
6. Create a new issue with the patch:
    1. Follow the "New" link: https://issues.apache.org/jira/secure/CreateIssue!default.jspa
    2. For the "Issue Type", select "Bug", "Improvement" or "New Feature"
    3. Click "Next>>"
    4. Enter summary / description and select the component
    5. Click "Create"

6. Click "Attach file" and select the mychanges.patch file
7. Check the "Grant license to ASF for inclusion in ASF works" option
8. Click on "Watching" and then click on "Start" watching to get updates

7. If your patch includes major changes, the Apache Shindig project uses codereview.appspot.com to review your patch. You need a Google Account to process. Go to the sign up form to create an account if you don't already have one. When creating a codereview issue add dev-remailer@shindig.apache.org as the reviewer and **DO NOT** add dev@.

The newly created issue will automatically be sent to dev@shindig.apache.org. You should also subscribe using dev-subscribe@shindig.apache.org to see all the feedback, in which case you'll get the updates on the issue without needing to "Watch" them individually.

12.2 How do I Join the Project?

Projects at Apache operate under a meritocracy, meaning those that the developers notice participating to a high extent will be invited to join the project as a committer.

This is as much based on personality and ability to work with other developers and the community as it is with proven technical ability. Being unhelpful to other users, or obviously looking to become a committer for bragging rights and nothing else is frowned upon, as is asking to be made a committer without having contributed sufficiently to be invited.

12.3 Developers Conventions

There are a number of conventions used in the project, which contributors and developers alike should follow for consistency's sake.

- Code Style And Code Conventions
- Jira Conventions
- SVN Conventions

12.4 Developers Documentation

- Apache Shindig Release Process

12.5 Apache Resources

- About the Apache Software Foundation
- Developer Resources
- Mailing List Stats (browse to find shindig lists)
- Apache Wiki

13 Getting Help

13.1 Getting Help

So something didn't work as you expected it to? You think that *Apache Shindig is broken*. What should you do?

Here's a list of actions that you can take:

13.1.1 Search the mailing list archives

Someone else might have experienced the same problem as you before. A list of mail-archives can be found on mailing list index page. Please search one of them before going any further.

13.1.2 Ask on the mailing list

Our community is very helpful, just ask it the right way. See the references section, at the end of this page, for info on how to do that. Subscribe to the list and describe your problem there. Don't expect to get an answer right away. Sometimes it takes a couple of days.

13.1.3 Submit an issue

If it turns out that there is indeed something wrong with Apache Shindig, you should report it to our issue management system JIRA.

First of all you need to create an account in JIRA. This is so that we can communicate with you while we work together on the issue. Go to the sign up form to create an account if you don't already have one.

13.1.3.1 How?

Just describing the problem is not enough. It takes a developer a lot of time to reproduce your problem. Issues that states problems without something usable to try out will be closed as incomplete.

Please attach a test case that we can download and run. We appreciate reports, but if you don't have something usable for us it's incredibly hard for us to manage the issues.

If you want to contribute to solve the problem, please read the creating and submitting a patch section.

13.1.4 Using IRC (Internet Relay Chat)

Where is it?

13.1.5 References

- How To Ask Questions The Smart Way
- How to Get Support from Open Source Mailing Lists

14 Code Conventions

14.1 Apache Shindig Code Style And Code Conventions

This document describes how developers and contributors should write code. The reasoning of these styles and conventions is mainly for consistency, readability and maintainability reasons.

14.1.1 Generic Code Style And Convention

All working files (java, xml, others) should respect the following conventions:

- **License Header**: Always add the current ASF license header in all versionned files.
- **Trailing Whitespaces**: Remove all trailing whitespaces. If your are an Eclipse user, you could use the Anyedit Eclipse Plugin.

and the following style:

- **Indentation**: **Never** use tabs!
- **Line wrapping**: Always use a 100-column line width.

Note: The specific styles and conventions, listed in the next sections, could override these generic rules.

14.1.2 Java

14.1.2.1 Java Code Style

The full Apache Shindig for Java is described in our Wiki.

14.1.2.2 Java Code Convention

ongoing

14.1.2.3 Javadoc Code Convention

Our Javadoc code convention is mainly:

- No author tag
- No version tag

ongoing

14.Eclipse 3.2+

Download `shindig-eclipse-codestyle_2.xml`.

After this, select Window > Preferences, and open up the configuration for Java > Code Style > Code Formatter. Click on the button labeled Import... and select the file you downloaded. Give the style a name, and click OK.

Similarly, you could import shindig.importorder and shindig-eclipse-codetemplate.xml

14.1.3 PHP

TODO, see How to Contribute page

14.1.4 JavaScript

TODO, see How to Contribute page

15 Jira Conventions

15.1 Apache Shindig JIRA Conventions

This document describes how Apache Shindig developers should use JIRA, our issue tracking.

15.1.1 When To Create a JIRA Issue?

This section discusses when to create a JIRA issue versus just committing a change in SVN.

- **Minor changes**, like code reformatting, documentation fixes, etc. that aren't going to impact other users can be committed without much issue.
- **Larger changes**, like bug fixes, API changes, significant refactoring, new classes, and pretty much any change of more than 100 lines, should have a JIRA ticket associated with it, or at least an email discussion.

15.1.2 How To Use Issue Details?

This section presents some conventions about the issue fields.

15.1.2.1 Priority

Committers has the responsibility to realign priority by editing the issue.

Reasoning: having a correct release note.

15.1.2.2 Assignee

Committers could assign an issue to a specific committer if he thinks it is the right committer.

15.1.2.3 Component/s

Committers has the responsibility to specify the correct the component by editing the issue.

Reasoning: having a correct release note.

15.1.2.4 Affects Version/s

By default, the Maven team considers that an issue, which affects a given version, affects also precedent versions, i.e. issue which affects Maven 2.0.9 will affect also 2.0, 2.0.1 ... 2.0.9. If it is a regression, the committers should specify the affected versions.

Reasoning: having a correct release note.

15.1.2.5 Fix Version/s

TO BE DISCUSSED

15.1.2.6 Time Tracking

The Apache Shindig team never uses it. Committers could do it, but like said, it will never be used.

15.1.3 Further Links

- JIRA Documentation
- What is an Issue?
- What is a project?

16 SVN Conventions

16.1 Apache Shindig SVN Conventions

This document describes how developers should use SVN, our SCM.

16.1.1 Subversion Configuration

Before committing files in subversion repository, you need to read the Committer Subversion Access document and you must set your svn client with this properties file: svn-props

16.1.2 Commit Message Template

Based on ASF committer's FAQ, the commits should have a message that follows this template:

```
[issue1, issue2] <<comment>>
Submitted by: (when it was a patch, put that persons name there)

o some comments
```

Where:

- **issue** can be omitted if there was no relevant JIRA issue, though it is strongly encouraged to create one for significant changes.
- **Submitted by** only needs to be specified when a patch is being applied for a non-committer.
- **comments** some words about the commits.

16.1.3 Apply User Patch

By default, the committer should apply the patch without any **major** modifications. In a second step, the committer could apply any changes as usual.

16.1.4 Edit Commit Message

If you want to edit a commit message, you could call:

```
svn pe svn:log --revprop -r XXX
```

where **XXX** is the wanted version

16.1.5 Other useful Subversion commands while developing

If you've done a chunk of work and you would like ditch your changes and start from scratch use this command to revert to the original checkout:

```
$ svn revert -R .
```

The `-R` argument means that the command will recurse down all directories and revert all changes.

Before committing code to the Subversion repository we always set the `svn:ignore` property on the directory to prevent some files and directories to be checked in. We always exclude the IDE project files and the `target/` directory. Instead of keeping all of the excludes in mind all the time it's useful to put them all in a file and reference the file with the `-F` option:

```
$ svn propset svn:ignore -F ~/bin/svnignore .
```

An example svnignore file:

```
target
*~
*.log
.classpath
.project
*.ipr
*.iws
*.iml
```

17 Shindig Release Process

17.1 Apache Shindig Release Process

This document describes how Apache Shindig's committers will make a release.

Notes:

1. This process is based on the Maven Release Process. Please read it firstly for the prerequisites and technical notes.
2. The items 2 to 5 should take you 1-2 hours, depending your machine and your Internet connection.
3. Some of the build steps require more than the default amount of maven memory. You should set MAVEN_OPTS to a larger value, -Xmx512m seems to be sufficient.

17.1.1 1. Release Discussions

The starting point will be a consensus in the developer community about the release and the JIRA issues to include (or not) in the next release. Typically, the discussion will be on the Dev List.

17.1.2 2. Prepare The Release

Duration: 20 min

Now that everybody in the community is happy to push a release, the release manager needs to verify the source code before to continue:

1. Make sure there are no snapshots (less the current version) in the POMs to be released.
2. Check that your POMs will not lose content when they are rewritten during the release process.
   A. `mvn release:prepare -DdryRun=true -Papache-release`
   B. Diff the original file pom.xml with the one called pom.xml.tag to see if the license or any other info has been removed. The only things that should be different between these files are the *version* and *scm* elements. Any other changes, you must backport yourself to the original pom.xml file and commit before proceeding with the release.
   C. Verify that all Apache Shindig modules will be touched, including `assembly` and `samples`.

Next, prepare the release by calling Maven:

```
mvn release:clean -Papache-release
mvn release:prepare -Papache-release
```

Notes:

1. *-Papache-release* is **very** important since `assembly` and `samples` modules are not included by default in the build (for productivity reasons).
2. Preparing the release will create a new tag in SVN, automatically checking in on your behalf. Verify the SVN logs in the Commit List or review the generated tag.
3. All license headers and legal files will be checked by RAT (Release Audit Tool).

17.1.3 3. Stage The Release

Duration: 30 min

Call Maven to publish the generated artifacts:

```
mvn release:perform -Papache-release
```

Note: All artifacts will be signed and pushed to https://repository.apache.org/.

Close the staging repository as described in the points 5 and 6 of the Maven Release Process.

17.1.4 4. Deploy The Documentation

Duration: 20 min

Call Maven to publish the generated technical site:

```
cd target/checkout
mvn site -Preporting
mvn site:deploy
```

Optional: Redeploy the current website:

```
cd trunk/site
mvn clean site
mvn site:deploy
```

Note: Wait for the sync before to see the changes in http://shindig.apache.org/.

17.1.5 5. Propose A Vote

Start a release vote on the Dev List. The vote must be aligned with the Apache vote process.

Typically, the mail should include the Jira release notes, the staging repository URL and the site URL, for instance:

```
To: "Shindig Developers List" <dev@shidnig.apache.org>
Subject: [VOTE] Release Apache Shindig version X.Y

Hi,

We solved N issues:
https://issues.apache.org/jira/secure/ReleaseNote.jspa?version=XXX&styleName=Html&p

There are still a couple of issues left in JIRA:
https://issues.apache.org/jira/secure/IssueNavigator.jspa?reset=true&pid=12310741&s

Staging repo:
https://repository.apache.org/content/repositories/shindig-staging-[YOUR REPOSITORY

Web site:
http://shindig.apache.org/

Vote open for 72 hours.

[ ] +1
[ ] +0
[ ] -1
```

17.1.6 6. Publish The Release

Promote the release as described in the points 10 of the Maven Release Process.

Also, update Jira to specify the release date.

17.1.7 7. Publish The Website

Update the download page and redeploy the website (need to sync):

```
cd trunk/site
mvn clean site
mvn site:deploy
```

17.1.8 8. Announce The Release

Create an announcement similar to:

```
From: YOUR_APACHE_USERNAME@apache.org
To: announce@apache.org, dev@shindig.apache.org
Subject: [ANN] Apache Shindig X.Y Released

The Apache Shindig team is proud to announce the release
of Apache Shindig, version X.Y-incubating.

Apache Shindig is a JavaScript container and
implementations of the backend APIs and proxy required for hosting
OpenSocial applications.

http://shindig.apache.org/

Enjoy,

-The Apache Shindig Team
```

17.1.9 9. Celebrate :o)

18 FAQ

18.1 Frequently Asked Questions

1. What is OpenSocial?
2. Which OpenSocial spec versions Apache Shindig implements?
3. Does Apache Shindig plan to start a new language implementation?
4. OpenSocial Jargon
5. Where are the opensocial/gadget XSDs?

What is OpenSocial?

OpenSocial is a set of common Application Programming Interfaces (APIs) for web-based social network applications, developed by Google and a number of other social networks. Applications implementing the OpenSocial APIs will be interoperable with any social network system that supports them. It was released November 1, 2007.

See resources page for more information.

[top]

Which OpenSocial spec versions Apache Shindig implements?

Apache Shindig implements OpenSocial API spec. 0.8.1 and 0.9.

[top]

Does Apache Shindig plan to start a new language implementation?

Apache Shindig project actually supports both Java and PHP version. Other Opensocial container implementations exist.

[top]

OpenSocial Jargon

Owner

The user who has installed the app

Viewer

The user who is using the app

Friends

Relationship between two users

OpenSocial App

A Gadget Specification compliant

App Data

Data stored by an app

Activity

Data stored by a user (what a user does)

Container

A website

[top]

Where are the opensocial/gadget XSDs?

No official XSDs exist yet but you could find unofficial XSDs in the Opensocial resources project:

- OpenSocial Gadgets API Specification v0.8
- OpenSocial Gadgets API Specification v0.9
- OpenSocial Restful Protocol v0.81
- OpenSocial RESTful Protocol Specification v0.9

Note: for performance reasons, Apache Shindig will not validate files against these XSDs at runtime.

[top]

19 Powered By

19.1 Powered by Apache Shindig

19.2 Software that uses Apache Shindig

- OpenSocial Pages
- Eureka Streams
- Partuza based on PHP Apache Shindig
- SocialSite based on Java Apache Shindig
- **Add more, see How to Contribute page**

19.3 Opensocial sites or projects using Apache Shindig

- LinkedIn
- hi5
- Yahoo!
- Orkut
- Zing
- **Add more, see How to Contribute page**

20 Resources

20.1 Articles on Apache Shindig

All articles on Apache Shindig are listed on our Wiki.

If you are writing an article on Apache Shindig we suggest contacting the developers on the mailing list as we would be happy to provide feedback to help ensure accuracy in your article. Just ping us on the dev mailing list to get in touch.

20.2 Resources

The following list is some pointers on related OpenSocial resources.

- OpenSocial Foundation
- OpenSocial Community Wiki
- OpenSocial project
- OpenSocial resources project
- Gadget resources project
- OpenSocial Templates
- OpenSocial Dev App
- OpenSocial Templates Developer App
- **Add more, see How to Contribute page**