eXo Social Reference Guide

Chapter 1. Applications

1.1. List of Portlets in Social

All the portlets are in the file *social-portlet.war*

Portlet name	Description
Members Portlet	To enable users to search for Space members or list Space members in the alphabet order.
My Spaces Portlet	To display the Space page where you can add new Spaces, search for Spaces or list Spaces in the alphabet order.
Space Activity Stream Portlet	To share Spaces activity information.
Invitations Portlet	To list all people that invite users.
Requests Portlet	To list all invitations that the user requests.
Invitation Spaces Portlet	To enable users to search for Spaces, and list spaces in the alphabet order and view all the Space invitations.
Pending Spaces Portlet	To display the **Space requests to join** page where you can search for Spaces, list spaces in the alphabet order and view all the pending requests to join Spaces.
Public Spaces Portlet	To display the **Public Spaces** page where you can search for Spaces, list Spaces in the alphabet order and view all available public Spaces to join.
User Activity Stream Portlet	To update and share user's activities and/or status.
People Portlet	To display the People page where you can find people, display people names in the alphabet order.
Connections Portlet	To display the Connections page where you can access your network, invitations to connect or connection requests.
User Profile Portlet	To display the user profile page where users can view and/or edit basic information, contacts, experience or change their avatar.

1.2. List of Gadgets in Social

All Social Gadgets are in *opensocial.war*.

1.2.1. Activity Stream

- **Description**: Manages activities of users. Some kinds of activities, such as update status, like/unlike activities, comment activities, delete activities and delete comments.
- **Used REST services**: ActivitiesRestServices

1.2.2. Social RSS Reader

- **Description**: Fetches and parses RSS information from a specific url.
- **Description of user preferences**: The number of RSS per page is 10 by default. Get RSS from input URL (Default URL is http://blog.exoplatform.org/feed/).
- **Used REST services**: N/A

1.2.3. My Connections

- **Description**: Displays a viewer's information and his connections' information.
- **Description of user preferences**: Connections displayed per page. The number of connections displayed is 5 by default.
- **Used REST services**: N/A

1.2.4. My Spaces

- **Description**: Displays all spaces that a user has the "member" role.
- **Used REST services**: SpacesRestService

Chapter 2. Configuration

2.1. Components

2.1.1. ActivityManager

Configuration name	Data type	Default Value	Description
allowedTags	`String list`	`b, i, a, span, em, strong, p, ol, ul, li, br, img`	html tags

2.1.2. SpaceService

Configuration name	Data type	Default Value	Description
space.homeNodeApp	`String`	`SpaceActivityStreamPortlet`	The home application for a space
space.apps	`String list`	`DashboardPortlet:true, SpaceSettingPortlet:false, MembersPortlet:true`	The space applications

Note

Deprecated: Use external-component-plugins instead: org.exoplatform.social.core.space.SpaceApplicationConfigPlugin.

2.1.3. IdentityManager

Configuration name	Data type	Default Value	Description
providers	`String`	`org.exoplatform.social.core.identity.provider.SpaceIdent`	The identity providers.

2.1.4. ProfileConfig

Configuration name	Data type	Default Value	Description
nodetype.emails	`String`	`exo:profileKeyValue`	
nodetype.phones	`String`	`exo:profileKeyValue`	
nodetype.ims	`String`	`exo:profileKeyValue`	
nodetype.urls	`String`	`exo:profileKeyValue`	

Configuration name	Data type	Default Value	Description
nodetype.address	`String`	`exo:profileAddress`	
nodetype.experiences	`String`	`exo:profileExperience`	
nodetype.education	`String`	`exo:profileEducation`	
forceMultiValue	`String`	`xxxxxxxxxxxx`	

2.1.5. ServiceProviderStore

Configuration name	Data type	Default Value	Description
sample-provider	properties-params		sample service provider

2.2. External component plugins

2.2.1. MentionsProcessor

Configuration name	Data type	Default Value	Description
priority	`String`	2	priority of this processor (lower are executed first)

2.2.2. PortletPreferenceRequiredPlugin

Configuration name	Data type	Default Value	Description
portletsPrefsRequired	`String list`	`SpaceActivityStreamPortlet, SpaceSettingPortlet, MembersPortlet`	The portlet name which requires space URL preference

2.2.3. SpaceApplicationConfigPlugin

Configuration name	Data type	Description
spaceHomeApplication	`SpaceApplicationConfigPlugin$SpaceApplicationConfig`	The space application for the space home node.
spaceApplicationList	`SpaceApplicationConfigPlugin`	space application list configuration

2.2.4. AddNodeTypePlugin

Configuration name	Data type	Default Value	Description
autoCreatedInNewRepository	String	jar:/conf/portal/core-nodetypes.xml	Node types configuration file.

2.3. RelationshipManager

The Service is used to manipulate user relationships.

2.4. SpaceIdentityProvider

The provider is to give identity space instances.

2.5. SpaceApplicationHandler

The service is to handle all events related to creating and managing all the application spaces.

2.6. ExoPeopleService

The service is to manipulate all data related to people in the Portal.

2.7. Space Service

2.7.1. Description

The service for spaces management includes creating spaces, and installing applications.

2.7.2. Components configuration

Component name	Description
SpaceServiceImpl	Implementation class of Space Service.

```
<component>
    <key>org.exoplatform.social.core.space.spi.SpaceService</key>
    <type>org.exoplatform.social.core.space.impl.SpaceServiceImpl</type>
    <!--Deprecated, Use external-component-plugins instead
    <init-params>
      <values-param>
        <name>space.homeNodeApp</name>
        <value>SpaceActivityStreamPortlet</value>
      </values-param>
      <values-param>
        <name>space.apps</name>
        <value>DashboardPortlet:true</value>
```

```
        <value>SpaceSettingPortlet:false</value>
        <value>MembersPortlet:true</value>
      </values-param>
    </init-params>
    -->
</component>
```

Configuration name	Data Type	Possible Value	Default Value	Description
SpaceActivityStreamPortlet	String	N/A	`SpaceActivityStreamPortlet`	The name of portlet displaying activities of spaces
space.apps	`String list`	`Porlets' name: true/false`	`DashboardPortlet:true, SpaceSettingPortlet:false, MembersPortlet:true`	The list of configurations for portlets used as portlet applications.

2.7.3. External plug-in configuration

2.7.3.1. PortletPreferenceRequiredPlugin

```
<external-component-plugins>
    <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
    <component-plugin>
      <name>portlets.prefs.required</name>
      <set-method>setPortletsPrefsRequired</set-method>
      <type>org.exoplatform.social.core.application.PortletPreferenceRequiredPlugin</type>
      <init-params>
        <values-param>
          <name>portletsPrefsRequired</name>
          <value>SpaceActivityStreamPortlet</value>
          <value>SpaceSettingPortlet</value>
          <value>MembersPortlet</value>
        </values-param>
      </init-params>
    </component-plugin>
</external-component-plugins>
```

In which:

Name	Set-method	Type	Description
PortletPreferenceRequiredPlugin	setPortletsPrefsRequired	org.exoplatform.social.core.application.PortletPreferenceRequiredPlugin	Configure Portlets' preference portlet names which will have portlet preference of space context.

- Init-params:

Name	Possible value	Default Value	Description
portletsPrefsRequired	`Portlet names`	`SpaceActivityStreamPortlet, SpaceSettingPortlet; MembersPortlet`	List of portlets which need to be saved and get the space context name.

2.7.3.2. SpaceApplicationConfigPlugin

Note

Since 1.2.0-GA

```
<external-component-plugins>
   <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
   <!-- Default applications to be installed when creating a new space -->
   <component-plugin>
      <name>Space Application Configuration</name>
      <set-method>setSpaceApplicationConfigPlugin</set-method>
      <type>org.exoplatform.social.core.space.SpaceApplicationConfigPlugin</type>
      <init-params>
        <object-param>
          <name>spaceHomeApplication</name>
          <description>Space Home Application</description>
          <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplication">
            <field name="portletApp"><string>social-portlet</string></field>
            <field name="portletName"><string>SpaceActivityStreamPortlet</string></field>
            <field name="appTitle"><string>Home</string></field>
            <!--<field name="icon"><string>SpaceHomeIcon</string></field>-->
          </object>
        </object-param>
        <object-param>
          <name>spaceApplicationListConfig</name>
          <description>space application list configuration</description>
          <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin">
            <field name="spaceApplicationList">
              <collection type="java.util.ArrayList">
                <value>
                  <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplicatior
                    <field name="portletApp"><string>dashboard</string></field>
                    <field name="portletName"><string>DashboardPortlet</string></field>
                    <field name="appTitle"><string>Dashboard</string></field>
                    <field name="removable"><boolean>true</boolean></field>
                    <field name="order"><int>1</int></field>
                    <field name="uri"><string>dashboard</string></field>
                    <!--<field name="icon"><string>SpaceDashboardIcon</string></field>-->
                  </object>
                </value>
                <value>
                  <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplicatior
                    <field name="portletApp"><string>social-portlet</string></field>
                    <field name="portletName"><string>SpaceSettingPortlet</string></field>
                    <field name="appTitle"><string>Space Settings</string></field>
                    <field name="removable"><boolean>false</boolean></field>
                    <field name="order"><int>2</int></field>
                    <field name="uri"><string>settings</string></field>
                    <!--<field name="icon"><string>SpaceSettingsIcon</string></field>-->
                  </object>
                </value>
                <value>
                  <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplicatior
                    <field name="portletApp"><string>social-portlet</string></field>
                    <field name="portletName"><string>MembersPortlet</string></field>
                    <field name="appTitle"><string>Members</string></field>
                    <field name="removable"><boolean>true</boolean></field>
                    <field name="order"><int>3</int></field>
                    <field name="uri"><string>members</string></field>
                    <!--<field name="icon"><string>SpaceMembersIcon</string></field>-->
                  </object>
                </value>
              </collection>
            </field>
          </object>
        </object-param>

      </init-params>
    </component-plugin>
 </external-component-plugins>
```

In which:

Name	Set-method	Type	Description
Space Application Configuration	setSpaceApplicationConfigPlugin	org.exoplatform.social.core.application.PortletPreferenceRequiredPlugin	Configure portlets preference required space applications to be installed when creating a new space.

2.8. Activity Manager

2.8.1. Description

The service is to manipulate space and user activities.

2.8.2. Component plug-in configuration

```
<component-plugin>
      <name>OSHtmlSanitizer</name>
      <set-method>addProcessorPlugin</set-method>
      <type>org.exoplatform.social.core.processor.OSHtmlSanitizerProcessor</type>
      <init-params>
        <values-param>
          <name>allowedTags</name>
          <value>b</value>
          <value>i</value>
          <value>a</value>
          <value>span</value>
          <value>em</value>
          <value>strong</value>
          <value>p</value>
          <value>ol</value>
          <value>ul</value>
          <value>li</value>
          <value>br</value>
          <value>img</value>
        </values-param>
      </init-params>
</component-plugin>
```

In which,

Name	Set-method	Type	Description
OSHtmlSanitizerProcessor	addProcessorPlugin	org.exoplatform.social.core.processor.OSHtmlSanitizerProcessor	The plugin filters valid html tags appearing in the Activity body (content).

- Init-params:

Name	Possible value	Default Value	Description
allowedTags	`html tags`	b, i, a, span, em, strong,	To process and render

Name	Possible value	Default Value	Description
		p, ol, ul,li, br, img	html tags in the activity content (body).

2.8.3. External plug-in configuration

```
    <component-plugin>
      <name>MentionsProcessor</name>
      <set-method>addProcessorPlugin</set-method>
      <type>org.exoplatform.social.core.processor.MentionsProcessor</type>
      <init-params>
        <value-param>
          <name>priority</name>
          <description>priority of this processor (lower number will be executed first)</description>
          <value>2</value>
        </value-param>
      </init-params>
    </component-plugin>
```

In which:

Name	Set-method	Type	Description
MentionsProcessor	addProcessorPlugin	org.exoplatform.social.core.processor.MentionsProcessor	A processor that substitutes @username expressions by a link on the user profile.

- Init-params:

Name	Possible value	Default Value	Description
priority	`priority number`	2	Priority of this processor (The lower level will be executed first).

2.9. Identity Manager

2.9.1. Description

The service is to manipulate the identity operations like creating, getting, deleting or finding a profile.

2.9.2. Component plug-in configuration

```
<component>
    <key>org.exoplatform.social.core.manager.IdentityManager</key>
    <type>org.exoplatform.social.core.manager.IdentityManager</type>
    <component-plugins>
      <component-plugin>
```

```
        <name>SpaceIdentityProvider plugin</name>
        <set-method>registerIdentityProviders</set-method>
        <type>org.exoplatform.social.core.identity.IdentityProviderPlugin</type>
        <init-params>
          <values-param>
          <name>providers</name>
            <description>Identity Providers</description>
            <value>org.exoplatform.social.core.identity.provider.SpaceIdentityProvider</value>
          </values-param>
        </init-params>
      </component-plugin>
    </component-plugins>
</component>
```

In which:

Name	Set-method	Type	Description
SpaceIdentityProvider plugin	registerIdentityProviders	org.exoplatform.social.core.identity.IdentityProviderPlugin	The plugin that provides identity for a space

- Init-params:

Name	Possible value	Default Value	Description
providers	`Every other identity providers`	`org.exoplatform.social.core.identity.provider.SpaceIdent`	Identity Provider instances for managing identities.

2.10. OpenSocial Rest Context Configuration

2.10.1. Description

The service is used to configure the portal container name when there is a OpenSocial REST API request. By configuring this service, we can make sure to to reach the right portal container. The default portal container is `portal`.

2.10.2. Component plug-in configuration

This should be used when there is a portal container different than the default `portal` one.

```
<external-component-plugins>
  <target-component>org.exoplatform.social.opensocial.auth.RestPortalContainerNameConfig</target-component>
  <component-plugin>
    <name>set portal container name used for REST service</name>
    <set-method>setRestContainerName</set-method>
    <type>org.exoplatform.social.opensocial.auth.RestPortalContainerNameConfig</type>
      <init-params>
      <value-param>
        <name>rest-container-name</name>
        <value>socialdemo</value>
      </value-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

Name	Set-method	Type	Description
set portal container name used for REST service	setRestContainerName	org.exoplatform.social.opensocial.plugin.RestProviderContainerNameC	The plugin provider to set rest portal container name for OpenSocial REST API request

- Init-params:

Name	Possible value	Default Value	Description
rest-container-name	`any valid portal container name`	`N/A`	Portal container name

2.11. Spaces Template configuration

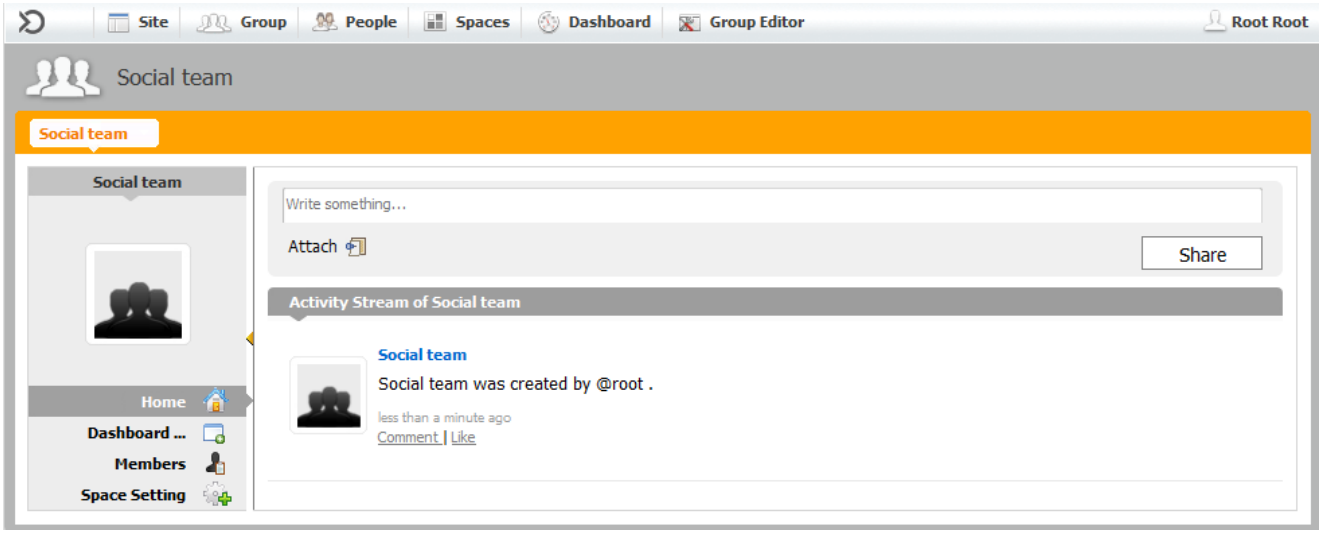
In the eXo Spaces, we may have two space types (classic and webos spaces). This is for the classic mode (it's the only one implemented now).

For the classic space, we can pre-configure the template, meaning that you can set up where your **menu** will be displayed or where your **application** will be displayed.

Here is an example of configuration file that displays the menu on the left. The Application will be inserted in the container with the id **Application**:

```
<page>
  <owner-type/>
  <owner-id/>
  <name/>
  <container id="SpacePage" template="system:/groovy/portal/webui/container/UITableColumnContainer.gtmpl">
    <container id="Menu" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
      <portlet-application>
        <portlet>
          <application-ref>space</application-ref>
          <portlet-ref>SpaceMenuPortlet</portlet-ref>
        </portlet>
        <access-permissions>*:/platform/users</access-permissions>
        <show-info-bar>false</show-info-bar>
      </portlet-application>
    </container>
    <container id="Application" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
    </container>
  </container>
</page>
```

In this example, the outer container contains two inner containers: one container has id as **Menu** for your Menu and another has id as **Application** containing your applications.



If you want to put your menu in right and your application in left, you can swap the declared position of two containers:

```
<page>
  <owner-type/>
  <owner-id/>
  <name/>
  <container id="SpacePage" template="system:/groovy/portal/webui/container/UITableColumnContainer.gtmpl">
    <container id="Application" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
    <container id="Menu" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
      <portlet-application>
        <portlet>
          <application-ref>space</application-ref>
          <portlet-ref>SpaceMenuPortlet</portlet-ref>
        </portlet>
        <access-permissions>*:/platform/users</access-permissions>
        <show-info-bar>false</show-info-bar>
      </portlet-application>
    </container>
    </container>
  </container>
</page>
```

Here is the configure file in FishEye: http://fisheye.exoplatform.org/browse/social/trunk/extension/war/src/main/webapp/WEB-INF/conf/portal/template/pages/sp

In your tomcat, this configuration file is at *$EXO*TOMCAT/webapps/social-ext/WEB-INF/conf/portal/template/pages/space/page.xml.

2.12. Configure the oauth 2 legged scenario

This is to explain how to configure the oAuth 2 legs scenario in openSocial.

For more information about this, visit the website: great article

2.12.1. Generate the certificates

To generate the key:

```
$ openssl req -newkey rsa:1024 -days 365 -nodes -x509 -keyout testkey.pem
     -out testkey.pem -subj '/CN=mytestkey'
$ openssl pkcs8 -in testkey.pem -out oauthkey.pem -topk8 -nocrypt -outform PEM
```

2.12.2. Configure the property file

Edit container.js and change the following parameter to point to your private key and your key name.

```
"gadgets.signingKeyFile" : "oauth.pem",
"gadgets.signingKeyName" : "oauthKey",
```

Chapter 3. Developers References

3.1. UI Extensions

3.1.1. About Activity Plugin

Since eXo Social 1.1.0-GA, the activity plugin feature was introduced to allow using the activity composer extension and the custom UI component for displaying activity by its type.

3.1.2. How to create activity plugin

You should have an idea about UI Extension Framework. If you have already worked with UI Extension Framework, it's really easy to create activity plugin. If no, you have chance to work with it now . You should have a look at UI Extension Framework.

3.1.2.1. Create a custom ui component for displaying the activity based on its type

Note

("Project Code can be downloadable here")

When an activity is displayed, UIActivityFactory will look for its registered custom actvity display by activity's type. If not found, UIDefaultActivity will be called for displaying that activity.

For example, in eXo Social, there is an activity of type "exosocial:spaces" created by SpaceActivityPublisher. And now we want to display it with out own ui component instead of default one.

First of all, create a sample project:

```
mvn archetype:generate
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] artifact org.apache.maven.plugins:maven-archetype-plugin: checking for updates from central
[INFO] snapshot org.apache.maven.plugins:maven-archetype-plugin:2.0-alpha-6-SNAPSHOT: checking for updates from
[INFO] snapshot org.apache.maven.archetype:maven-archetype:2.0-alpha-6-SNAPSHOT: checking for updates from centr
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Default Project
[INFO]    task-segment: [archetype:generate] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing archetype:generate
[INFO] No goals needed for project - skipping
[INFO] snapshot org.apache.maven.archetype:archetype-common:2.0-alpha-6-SNAPSHOT: checking for updates from cent
[INFO] snapshot org.apache.maven.archetype:archetype-common:2.0-alpha-6-SNAPSHOT: checking for updates from apac
[INFO] [archetype:generate {execution: default-cli}]
[INFO] Generating project in Interactive mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quick
Choose archetype:
1: remote -> docbkx-quickstart-archetype (null)
2: remote -> gquery-archetype (null)
3: remote -> gquery-plugin-archetype (null)
//....

285: remote -> wicket-scala-archetype (Basic setup for a project that combines Scala and Wicket,
                depending on the Wicket-Scala project. Includes an example Specs
                test.)
286: remote -> circumflex-archetype (null)
Choose a number: 76: 76
Choose version:
```

```
1: 1.0
2: 1.0-alpha-1
3: 1.0-alpha-2
4: 1.0-alpha-3
5: 1.0-alpha-4
6: 1.1
Choose a number: : 1
Define value for property 'groupId': : org.exoplatform.social.samples
Define value for property 'artifactId': : exo.social.samples.activity-plugin
Define value for property 'version': 1.0-SNAPSHOT: 1.0.0-SNAPSHOT
Define value for property 'package': org.exoplatform.social.samples: org.exoplatform.social.samples.activityPlug
Confirm properties configuration:
groupId: org.exoplatform.social.samples
artifactId: exo.social.samples.activity-plugin
version: 1.0.0-SNAPSHOT
package: org.exoplatform.social.samples.activityPlugin
Y: y
```

Edit the pom.xml file as following. We'll work with Social latest major release: 1.1.0-GA.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sch
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.exoplatform.social</groupId>
    <artifactId>social-project</artifactId>
    <version>1.1.0-GA</version>
  </parent>

  <groupId>org.exoplatform.social.samples</groupId>
  <artifactId>exo.social.samples.activity-plugin</artifactId>
  <packaging>jar</packaging>
  <version>1.1.0-GA</version>

  <name>exo.social.samples.activity-plugin</name>

  <build>
    <sourceDirectory>src/main/java</sourceDirectory>
    <outputDirectory>target/classes</outputDirectory>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.gtmpl</include>
        </includes>
      </resource>
      <resource>
        <directory>src/main/java</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
      </resource>
    </resources>
  </build>

  <dependencies>

    <dependency>
      <groupId>org.exoplatform.portal</groupId>
      <artifactId>exo.portal.webui.core</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.exoplatform.portal</groupId>
      <artifactId>exo.portal.webui.portal</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.core</artifactId>
      <scope>provided</scope>
    </dependency>
```

```
    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.webui</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.service</artifactId>
      <scope>provided</scope>
    </dependency>

  </dependencies>

</project>
```

To use custom UI component for displaying its activity, we need a class that must be a subclass of `BaseUIActivity`. We call this `UISpaceSimpleActivity`:

```
package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;

@ComponentConfig(
  lifecycle = UIFormLifecycle.class,
  template = "classpath:groovy/social/plugin/space/UISpaceSimpleActivity.gtmpl"
)
public class UISpaceSimpleActivity extends BaseUIActivity {

}
```

The template `UISpaceSimpleActivity.gtmpl` should be created under `main/resources/groovy/social/plugin/space`:

```
<div>This is a space activity ui component displayed for type "exosocial:spaces"</div>
```

An activity builder as explained later is also needed.

```
package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.core.activity.model.Activity;
import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.social.webui.activity.BaseUIActivityBuilder;

public class SimpleSpaceUIActivityBuilder extends BaseUIActivityBuilder {

  @Override
  protected void extendUIActivity(BaseUIActivity uiActivity, Activity activity) {
    // TODO Auto-generated method stub

  }

}
```

We're done. Now create `configuration.xml` under conf/portal:

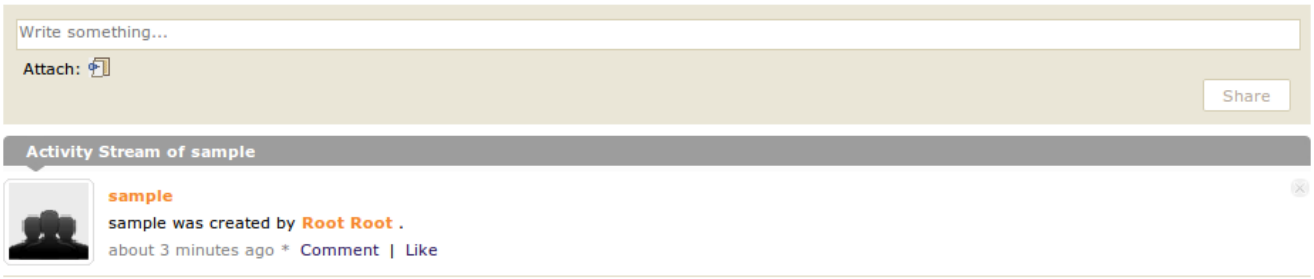
```
<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSche
  <external-component-plugins>
    <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
    <component-plugin>
      <name>add.action</name>
      <set-method>registerUIExtensionPlugin</set-method>
      <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
```

```
      <init-params>
        <object-param>
          <name>Simple Space Activity</name>
          <object type="org.exoplatform.social.webui.activity.UIActivityExtension">
            <field name="type"><string>org.exoplatform.social.webui.activity.BaseUIActivity</string></field>
            <field name="name"><string>exosocial:spaces</string></field>
            <field name="component"><string>org.exoplatform.social.samples.activityplugin.UISpaceSimpleActivity<
            <field name="activityBuiderClass"><string>org.exoplatform.social.samples.activityplugin.SimpleSpaceU
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</configuration>
```

Note that `exosocial:spaces`, its value have to match the activity's type you want to display with your ui component.

Assume that we have already built Social project with version 1.1.0-CR01 or above. If you don't know how to, have a look at Building from Sources with Social 1.1.0-CR01. Build sample project and copy jar file to /tomcat/lib. Run Social, create a space and access it, you can see:

space's activity of type "exosocial:spaces" is displayed by default in Social:



With our custom ui component for displaying activity of type: "exosocial:spaces":

1.1.1 Make the custom ui activity display have the look, feel and function like default one.

When displaying an activity, we should make sure the look and feel of custom ui component is consistent and matched other activities and have the functions of like, comments, too. So we're going to create another ui component to display, we call it: `UISpaceLookAndFeelActivity`:

```
package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.config.annotation.EventConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;

@ComponentConfig(
  lifecycle = UIFormLifecycle.class,
  template = "classpath:groovy/social/plugin/space/UISpaceLookAndFeelActivity.gtmpl",
  events = {
    @EventConfig(listeners = BaseUIActivity.ToggleDisplayLikesActionListener.class),
    @EventConfig(listeners = BaseUIActivity.ToggleDisplayCommentFormActionListener.class),
```

```
    @EventConfig(listeners = BaseUIActivity.LikeActivityActionListener.class),
    @EventConfig(listeners = BaseUIActivity.SetCommentListStatusActionListener.class),
    @EventConfig(listeners = BaseUIActivity.PostCommentActionListener.class),
    @EventConfig(listeners = BaseUIActivity.DeleteActivityActionListener.class, confirm = "UIActivity.msg.Are_Yo
    @EventConfig(listeners = BaseUIActivity.DeleteCommentActionListener.class, confirm = "UIActivity.msg.Are_You
  }
)
public class UISpaceLookAndFeelActivity extends BaseUIActivity {

}
```

Now create `UISpaceLookAndFeelActivity` template. The easiest way is to copy the content of `UIDefaultActivity.gtmpl` to this template file:

```
<%
  import org.exoplatform.portal.webui.util.Util;
  import org.exoplatform.webui.form.UIFormTextAreaInput;

  def pcontext = Util.getPortalRequestContext();
  def jsManager = pcontext.getJavascriptManager();
  def labelActivityHasBeenDeleted = _ctx.appRes("UIActivity.label.Activity_Has_Been_Deleted");
  def activity = uicomponent.getActivity();
  def activityDeletable = uicomponent.isActivityDeletable();
%>

<% if (activity) { //process if not null

  jsManager.importJavascript("eXo.social.Util", "/social-resources/javascript");
  jsManager.importJavascript("eXo.social.PortalHttpRequest", "/social-resources/javascript");
  jsManager.importJavascript("eXo.social.webui.UIForm", "/social-resources/javascript");
  jsManager.importJavascript("eXo.social.webui.UIActivity", "/social-resources/javascript");

  def labelComment = _ctx.appRes("UIActivity.label.Comment");
  def labelLike = _ctx.appRes("UIActivity.label.Like");
  def labelUnlike = _ctx.appRes("UIActivity.label.Unlike");
  def labelSource = _ctx.appRes("UIActivity.label.Source");
  def inputWriteAComment = _ctx.appRes("UIActivity.input.Write_A_Comment");
  def labelShowAllComments = _ctx.appRes("UIActivity.label.Show_All_Comments");
  def labelHideAllComments = _ctx.appRes("UIActivity.label.Hide_All_Comments");
  def labelOnePersonLikeThis = _ctx.appRes("UIActivity.label.One_Person_Like_This");
  def labelPeopleLikeThis = _ctx.appRes("UIActivity.label.People_Like_This");
  def labelYouLikeThis = _ctx.appRes("UIActivity.label.You_Like_This");
  def labelYouAndOnePersonLikeThis = _ctx.appRes("UIActivity.label.You_And_One_Person_Like_This");
  def labelYouAndPeopleLikeThis = _ctx.appRes("UIActivity.label.You_And_People_Like_This");

  def likeActivityAction = uicomponent.event("LikeActivity", "true");
  def unlikeActivityAction = uicomponent.event("LikeActivity", "false");

  def commentList = uicomponent.getComments();
  def allComments = uicomponent.getAllComments();
  if (allComments) {
    labelShowAllComments = labelShowAllComments.replace("{0}", allComments.size() + "");
    labelHideAllComments = labelHideAllComments.replace("{0}", allComments.size() + "");

  }
  def displayedIdentityLikes = uicomponent.getDisplayedIdentityLikes();
  def identityLikesNum = 0;
  def labelLikes = null;
  def toggleDisplayLikesAction = uicomponent.event("ToggleDisplayLikes");
  def startTag = "<a onclick={{{"}}}$toggleDisplayLikesAction{{{"}}} id={{{"}}}ToggleDisplayListPeopleLikes${act
  def endTag = "</a>";
  if (displayedIdentityLikes != null) {
    identityLikesNum = displayedIdentityLikes.length;
  }
  def commentListStatus = uicomponent.getCommentListStatus();
  def commentFormDisplayed = uicomponent.isCommentFormDisplayed();
  def likesDisplayed = uicomponent.isLikesDisplayed();
  //params for init UIActivity javascript object
  def params = """
    {activityId: '${activity.id}',
     inputWriteAComment: '$inputWriteAComment',
     commentMinCharactersAllowed: ${uicomponent.getCommentMinCharactersAllowed()},
     commentMaxCharactersAllowed: ${uicomponent.getCommentMaxCharactersAllowed()},
     commentFormDisplayed: $commentFormDisplayed,
     commentFormFocused: ${uicomponent.isCommentFormFocused()}
```

```
    }
  """
  jsManager.addOnLoadJavascript("initUIActivity${activity.id}");
  //make sures commentFormFocused is set to false to prevent any refresh to focus, only focus after post a comme
  uicomponent.setCommentFormFocused(false);
  def activityUserName, activityUserProfileUri, activityImageSource, activityContentBody, activityPostedTime;
  def commentFormBlockClass = "", listPeopleLikeBlockClass = "", listPeopleBGClass = "";
  if (!commentFormDisplayed) {
    commentFormBlockClass = "DisplayNone";
  }

  if (!likesDisplayed) {
    listPeopleLikeBlockClass = "DisplayNone";
  }

  if (uicomponent.isLiked()) {
    if (identityLikesNum > 1) {
      labelLikes = labelYouAndPeopleLikeThis.replace("{start}", startTag).replace("{end}", endTag).replace("{0}"
    } else if (identityLikesNum == 1) {
      labelLikes = labelYouAndOnePersonLikeThis.replace("{start}", startTag).replace("{end}", endTag);
    } else {
      labelLikes = labelYouLikeThis;
    }
  } else {
    if (identityLikesNum > 1) {
        labelLikes = labelPeopleLikeThis.replace("{start}", startTag).replace("{end}", endTag).replace("{0}", id
    } else if (identityLikesNum == 1) {
        labelLikes = labelOnePersonLikeThis.replace("{start}", startTag).replace("{end}", endTag);
    }
  }

  if (!labelLikes) {
   //hides diplayPeopleBG
   listPeopleBGClass = "DisplayNone";
  }

  activityContentTitle = activity.title;
  activityPostedTime = uicomponent.getPostedTimeString(activity.postedTime);
  activityUserName = uicomponent.getUserFullName(activity.userId);
  activityUserProfileUri = uicomponent.getUserProfileUri(activity.userId);

  activityImageSource = uicomponent.getUserAvatarImageSource(activity.userId);
  if (!activityImageSource)  {
    activityImageSource = "/social-resources/skin/ShareImages/SpaceImages/SpaceLogoDefault_61x61.gif";
  }

%>

<div class="UIActivity">
  <script type="text/javascript">
    function initUIActivity${activity.id}() {
      new eXo.social.webui.UIActivity($params);
    }
  </script>

  <% uiform.begin() %>
  <div class="NormalBox clearfix">
    <a class="Avatar" title="$activityUserName" href="$activityUserProfileUri">
      <img title="$activityUserName" src="$activityImageSource" alt="$activityUserName" height="54px" width="56p
    </a>
    <div class="ContentBox" id="ContextBox${activity.id}">
      <div class="TitleContent clearfix">
        <div class="Text">
          <a title="$activityUserName" href="$activityUserProfileUri">$activityUserName</a>
        </div>
      <% if (activityDeletable) {%>
        <div onclick="<%= uicomponent.event("DeleteActivity", uicomponent.getId(), ""); %>" class="CloseContentB
      <%}%>
      </div>
      <div class="Content">
        $activityContentTitle (from custom ui component)<br>
      </div>
      <div class="LinkCM">
        <span class="DateTime">$activityPostedTime *</span>
      <% def toggleDisplayCommentAction = uicomponent.event('ToggleDisplayCommentForm', null, false);
         def commentLink = "";
      %>
        <a class="LinkCM $commentLink" onclick="$toggleDisplayCommentAction" id="CommentLink${activity.id}" href
          $labelComment
```

```
      </a> |
    <% if (uicomponent.isLiked()) { %>
      <a onclick="$unlikeActivityAction" class="LinkCM" id="UnLikeLink${activity.id}" href="#unlike">
        $labelUnlike
      </a>
    <% } else { %>
      <a onclick="$likeActivityAction" class="LinkCM" id="LikeLink${activity.id}" href="#like">
        $labelLike
      </a>
    <% }%>
    </div>
  </div>

  <div class="ListPeopleLikeBG $listPeopleBGClass">
    <div class="ListPeopleLike">
      <div class="ListPeopleContent">
      <% if (!labelLikes) labelLikes = ""; %>
        <div class="Title">$labelLikes</div>
        <div class="$listPeopleLikeBlockClass">
        <%
        //def personLikeFullName, personLikeProfileUri, personLikeAvatarImageSource;

        displayedIdentityLikes.each({
          personLikeFullName = uicomponent.getUserFullName(it);
          personLikeProfileUri = uicomponent.getUserProfileUri(it);
          personLikeAvatarImageSource = uicomponent.getUserAvatarImageSource(it);
          if (!personLikeAvatarImageSource) {
            personLikeAvatarImageSource = "/social-resources/skin/ShareImages/activity/AvatarPeople.gif";
          }
          %>
            <a class="AvatarPeopleBG" title="$personLikeFullName" href="$personLikeProfileUri">
              <img width="47px" height="47px" src="$personLikeAvatarImageSource" alt="$personLikeFullName" tit
            </a>
         <% })%>
        </div>
        <div class="ClearLeft">
          <span></span>
        </div>
      </div>
    </div>
  </div>

  <div class="CommentListInfo">
  <% if (uicomponent.commentListToggleable()) {
  def showAllCommentsAction = uicomponent.event("SetCommentListStatus", "all");
  def hideAllCommentsAction = uicomponent.event("SetCommentListStatus", "none");
  %>
    <div class="CommentBlock">
      <div class="CommentContent">
        <div class="CommentBorder">
        <% if (commentListStatus.getStatus().equals("latest") || commentListStatus.getStatus().equals("none"))
          <a onclick="$showAllCommentsAction" href="#show-all-comments">
            $labelShowAllComments
          </a>
        <% } else if (commentListStatus.getStatus().equals("all")) { %>
          <a onclick="$hideAllCommentsAction" href="#hide-all-comments">
            $labelHideAllComments
          </a>
        <% } %>
        </div>
      </div>
    </div>
  <% } %>
  </div>
<% if (allComments.size() > 0) { %>
  <div class="DownIconCM"><span></span></div>
<% }%>

<%
def commenterFullName, commenterProfileUri, commenterImageSource, commentMessage, commentPostedTime;
def first = true, commentContentClass;
commentList.each({
  if (first & !uicomponent.commentListToggleable()) {
    commentContentClass = "CommentContent";
    first = false;
  } else {
    commentContentClass = "";
  }
  commenterFullName = uicomponent.getUserFullName(it.userId);
```

```
    commenterProfileUri = uicomponent.getUserProfileUri(it.userId);
    commenterImageSource = uicomponent.getUserAvatarImageSource(it.userId);
    if (!commenterImageSource) {
      commenterImageSource = "/social-resources/skin/ShareImages/activity/AvatarPeople.gif";
    }
    commentMessage = it.title;
    commentPostedTime = uicomponent.getPostedTimeString(it.postedTime);
  %>
    <div id="CommentBlock${activity.id}" class="CommentBox clearfix">
      <a class="AvatarCM" title="$commenterFullName" href="$commenterProfileUri">
        <img src="$commenterImageSource" alt="$commenterFullName" height="36px" width="38px">
      </a>
      <div class="ContentBox">
        <div class="Content">
          <a href="$commenterProfileUri"><span class="Commenter">$commenterFullName<span></a><br />
            $commentMessage
          <br/>
        </div>
        <div class="LinkCM">
          <span class="DateTime">$commentPostedTime</span>
        </div>
      </div>
    <%
      if (uicomponent.isCommentDeletable(it.userId)) {
    %>
      <div onclick="<%= uicomponent.event("DeleteComment", uicomponent.id, it.id); %>" class="CloseCMContentHili
    <% } %>
    </div>
  <% }) %>

    <div class="CommentBox $commentFormBlockClass clearfix" id="CommentFormBlock${activity.id}">
      <% uicomponent.renderChild(UIFormTextAreaInput.class); %>
      <input type="button" onclick="<%= uicomponent.event("PostComment") %>" value="$labelComment" class="Commer
    </div>

  </div>
  <% uiform.end() %>
</div>
<% } else { %> <!-- activity deleted -->
<div class="UIActivity Deleted">$labelActivityHasBeenDeleted</div>
<% }%>
```

And you should make needed modifications for this template. We make a small modification here:

```
<div class="Content">
  $activityContentTitle (from custom ui component)<br>
</div>
```

That's all. We need to reconfigure `configuration.xml`:

```
<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSche
  <external-component-plugins>
    <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
    <component-plugin>
      <name>add.action</name>
      <set-method>registerUIExtensionPlugin</set-method>
      <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
      <init-params>
        <object-param>
          <name>Look And Feel Space Activity</name>
          <object type="org.exoplatform.social.webui.activity.UIActivityExtension">
            <field name="type"><string>org.exoplatform.social.webui.activity.BaseUIActivity</string></field>
            <field name="name"><string>exosocial:spaces</string></field>
            <field name="component"><string>org.exoplatform.social.samples.activityplugin.UISpaceLookAndFeelActi
            <field name="activityBuiderClass"><string>org.exoplatform.social.samples.activityplugin.SimpleSpaceU
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</configuration>
```

Rebuild the sample project, copy jar to tomcat/lib. Re-run server and see the result:

Note

Currently, we have to copy and paste in template file. This way we have full control of the ui but it's not good when there's changes in UIDefaultActivity. We'll improve this soon so that no copy and pasted needed.

3.1.2.1.1. What is ActivityBuilder?

The ActivityBuilder is one class which is used to get values of **ExoSocialActivity** to set to UIActivity for displaying. eXo Social provides the BaseUIActivityBuilder class for developers to extend and customize their own activity builder easily and properly.

For example, to write your own UILinkActivityBuilder, you just need to extend BaseUIActivityBuilder and then customize attributes and behaviors of the activity builder as below.

```
public class UILinkActivityBuilder extends BaseUIActivityBuilder {
  private static final Log LOG = ExoLogger.getLogger(UILinkActivityBuilder.class);
  @Override
  protected void extendUIActivity(BaseUIActivity uiActivity, ExoSocialActivity activity) {
    UILinkActivity uiLinkActivity = (UILinkActivity) uiActivity;
    Map<String, String> templateParams = activity.getTemplateParams();
    uiLinkActivity.setLinkSource(templateParams.get(UILinkActivityComposer.LINK_PARAM));
    uiLinkActivity.setLinkTitle(templateParams.get(UILinkActivityComposer.TITLE_PARAM));
    uiLinkActivity.setLinkImage(templateParams.get(UILinkActivityComposer.IMAGE_PARAM));
    uiLinkActivity.setLinkDescription(templateParams.get(UILinkActivityComposer.DESCRIPTION_PARAM));
    uiLinkActivity.setLinkComment(templateParams.get(UILinkActivityComposer.COMMENT_PARAM));
  }
}
```

Note

To learn more about the ActivityBuilder, refer to the BaseUIActivity class.

3.1.2.2. Create a composer extension for composing activity on the UI composer and display it on activity stream.

The UIActivityComposer is an extended class of UIContainer that is used to display inputs for users to create their own activities. To write your own activity composer, it is recommended that you use the UIActivityComposer available in eXo Social.

For example, to create an input component for inserting video links into your activity, do the following steps:

1. Write UIVideoActivityComposer which extends UIActivityComposer. The UIActivityComposer allows you to input extended activities (for example, adding videos, links, or documents) on the UI composer.

```
package org.exoplatform.social.plugin.videolink;

import org.exoplatform.social.plugin.videolink.util.VideoEmbedTool;
@ComponentConfig(
        template = "classpath:groovy/social/plugin/videolink/UIVideoActivityComposer.gtmpl",
        events = {
                @EventConfig(listeners = UIVideoActivityComposer.SearchVideo.class),
                @EventConfig(listeners = UIVideoActivityComposer.SelectVideoFromResultList.class),
                @EventConfig(listeners = UIVideoActivityComposer.AttachActionListener.class),
                @EventConfig(listeners = UIVideoActivityComposer.ChangeLinkContentActionListener.class),
```

```
                @EventConfig(listeners = UIActivityComposer.CloseActionListener.class),
                @EventConfig(listeners = UIActivityComposer.SubmitContentActionListener.class),
                @EventConfig(listeners = UIActivityComposer.ActivateActionListener.class)
        }
)

public class UIVideoActivityComposer extends UIActivityComposer {

  public static final String LINK_PARAM = "link";
  public static final String IMAGE_PARAM = "image";
  public static final String TITLE_PARAM = "title";
  public static final String HTML_PARAM = "htmlembed";
  public static final String COMMENT_PARAM = "comment";

  private static final String HTTP = "http://";
  private static final String HTTPS = "https://";
  private JSONObject videoJson;
  private boolean linkInfoDisplayed_ = false;
  private Map<String, String> templateParams;

  /**
   * The constructor.
   */
  public UIVideoActivityComposer() {
    setReadyForPostingActivity(false);
    addChild(new UIFormStringInput("InputLink", "InputLink", null));
  }
  /**
   * Set the link info to be displayed.
   *
   * @param displayed
   */
  public void setLinkInfoDisplayed(boolean displayed) {
    linkInfoDisplayed_ = displayed;
  }

  /**
   * Set the template params.
   *
   * @param templateParams
   */
  public void setTemplateParams(Map<String, String> templateParams) {
    this.templateParams = templateParams;
  }

  /**
   * Get the template params.
   */
  public Map<String, String> getTemplateParams() {
    return templateParams;
  }

  /**
   * Clear the video json.
   */
  public void clearVideoJson() {
    videoJson = null;
  }

  /**
   * Get the video json.
   */
  public JSONObject getVideoJson() {
    return videoJson;
  }

  /**
   * Set the link.

   * @param url
   * @throws Exception
   */
  private void setLink(String url) throws Exception {
    if (!(url.contains(HTTP) || url.contains(HTTPS))) {
      url = HTTP + url;
    }

    videoJson = VideoEmbedTool.getoembedData(url);
    templateParams = new HashMap<String, String>();
```

```
    templateParams.put(LINK_PARAM, url);
    templateParams.put(TITLE_PARAM, videoJson.getString(VideoEmbedTool.OEMBED_TITLE));
    templateParams.put(HTML_PARAM, videoJson.getString(VideoEmbedTool.OEMBED_HTML));
    setLinkInfoDisplayed(true);
  }

  static public class AttachActionListener extends EventListener<UIVideoActivityComposer> {

    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
      WebuiRequestContext requestContext = event.getRequestContext();
      UIVideoActivityComposer uiComposerLinkExtension = event.getSource();
      String url = requestContext.getRequestParameter(OBJECTID);
      try {
        uiComposerLinkExtension.setLink(url.trim());
      } catch (Exception e) {
        uiComposerLinkExtension.setReadyForPostingActivity(false);
        return;
      }
      requestContext.addUIComponentToUpdateByAjax(uiComposerLinkExtension);
      event.getSource().setReadyForPostingActivity(true);
    }
  }

  static public class ChangeLinkContentActionListener extends EventListener<UIVideoActivityComposer> {
    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
      WebuiRequestContext requestContext = event.getRequestContext();
      UIVideoActivityComposer uiComposerLinkExtension = event.getSource();

      Map<String, String> tempParams = new HashMap<String, String>();

      uiComposerLinkExtension.setTemplateParams(tempParams);
      requestContext.addUIComponentToUpdateByAjax(uiComposerLinkExtension);
      UIComponent uiParent = uiComposerLinkExtension.getParent();
      if (uiParent != null) {
        uiParent.broadcast(event, event.getExecutionPhase());
      }
    }
  }

  public static class SelectVideoFromResultList extends EventListener<UIVideoActivityComposer>{
    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
      WebuiRequestContext requestContext = event.getRequestContext();
      UIVideoActivityComposer uiComposerLinkExtension = event.getSource();

    }
  }

  public static class SearchVideo extends EventListener<UIVideoActivityComposer>{

    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
      WebuiRequestContext requestContext = event.getRequestContext();
      UIVideoActivityComposer uiComposerLinkExtension = event.getSource();


    }
  }

  @Override
  public void onPostActivity(PostContext postContext, UIComponent source,
                             WebuiRequestContext requestContext, String postedMessage) throws Exception {

    templateParams.put(COMMENT_PARAM, postedMessage);
    setTemplateParams(templateParams);
    if (templateParams.size() == 0) {
      uiApplication.addMessage(new ApplicationMessage("UIComposer.msg.error.Empty_Message",
              null,
              ApplicationMessage.WARNING));
      return;
    }
    String title = "Shared a video: <a href={{{"}}}${" + LINK_PARAM + "}{{{"}}}>${" + TITLE_PARAM + "} </a>";
    ExoSocialActivity activity = new ExoSocialActivityImpl(userIdentity.getId(),
            UIVideoActivity.ACTIVITY_TYPE,
            title,
            null);
    activity.setTemplateParams(templateParams);
```

```
    if (postContext == UIComposer.PostContext.SPACE) {

      UIActivitiesContainer activitiesContainer = uiDisplaySpaceActivities.getActivitiesLoader().getActivitiesCo
      activitiesContainer.addActivity(activity);
      requestContext.addUIComponentToUpdateByAjax(activitiesContainer);
      requestContext.addUIComponentToUpdateByAjax(uiComposer);
    } else if (postContext == PostContext.USER) {
      UIUserActivitiesDisplay uiUserActivitiesDisplay = (UIUserActivitiesDisplay) getActivityDisplay();
      String ownerName = uiUserActivitiesDisplay.getOwnerName();
      Identity ownerIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME,
              ownerName, false);

      activityManager.saveActivity(ownerIdentity, activity);

      if (uiUserActivitiesDisplay.getSelectedDisplayMode() == UIUserActivitiesDisplay.DisplayMode.MY_STATUS) {
        UIActivitiesContainer activitiesContainer = uiUserActivitiesDisplay.getActivitiesLoader().getActivitiesC
        if (activitiesContainer.getChildren().size() == 1) {
          uiUserActivitiesDisplay.setSelectedDisplayMode(UIUserActivitiesDisplay.DisplayMode.MY_STATUS);
        } else {
          activitiesContainer.addActivity(activity);
          requestContext.addUIComponentToUpdateByAjax(activitiesContainer);
          requestContext.addUIComponentToUpdateByAjax(uiComposer);
        }
      } else{
        uiUserActivitiesDisplay.setSelectedDisplayMode(UIUserActivitiesDisplay.DisplayMode.MY_STATUS);
      }
    }
  }
}
```

2. Use the BaseUIActivity class to write and customize the UIActivity display as below:

```
package org.exoplatform.social.plugin.videolink;
import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;
import org.exoplatform.webui.config.annotation.EventConfig;
@ComponentConfig(lifecycle = UIFormLifecycle.class, template = "classpath:groovy/social/plugin/videolink/UIVideo
  @EventConfig(listeners = BaseUIActivity.ToggleDisplayLikesActionListener.class),
  @EventConfig(listeners = BaseUIActivity.ToggleDisplayCommentFormActionListener.class),
  @EventConfig(listeners = BaseUIActivity.LikeActivityActionListener.class),
  @EventConfig(listeners = BaseUIActivity.SetCommentListStatusActionListener.class),
  @EventConfig(listeners = BaseUIActivity.PostCommentActionListener.class),
  @EventConfig(listeners = BaseUIActivity.DeleteActivityActionListener.class, confirm = "UIActivity.msg.Are_You_
  @EventConfig(listeners = BaseUIActivity.DeleteCommentActionListener.class, confirm = "UIActivity.msg.Are_You_S
 }
)
public class UIVideoActivity extends BaseUIActivity {
  public static final String ACTIVITY_TYPE = "VIDEO_ACTIVITY";
  private String linkSource = "";
  private String linkTitle = "";
  private String linkHTML = "";
  private String linkComment = "";

  /**
   * Get the link comment.
   */
  public String getLinkComment() {
    return linkComment;
  }

  /**
   * Set the link comment.
   *
   * @param linkComment
   */
  public void setLinkComment(String linkComment) {
    this.linkComment = linkComment;
  }

  /**
   * Get the link html.
   */
  public String getLinkHTML() {
    return linkHTML;
```

```
  }

  /**
   * Set the link html.
   *
   * @param linkHTML
   */
  public void setLinkHTML(String linkHTML) {
    this.linkHTML = linkHTML;
  }

  /**
   * Get the link source.
   */
  public String getLinkSource() {
    return linkSource;
  }

  /**
   * Set the link source.
   *
   * @param linkSource
   */
  public void setLinkSource(String linkSource) {
    this.linkSource = linkSource;
  }

  /**
   * Get the link title.
   */
  public String getLinkTitle() {
    return linkTitle;
  }

  /**
   * Set the link title.
   *
   * @param linkTitle
   */
  public void setLinkTitle(String linkTitle) {
    this.linkTitle = linkTitle;
  }
}
```

3. Use the UIVideoActivityBuilder class to get values of **ExoSocialActivity** that are set to UIVideoActivity for displaying.

```
package org.exoplatform.social.plugin.videolink;
import java.util.Map;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;

public class UIVideoActivityBuilder extends BaseUIActivityBuilder {
  private static final Log LOG = ExoLogger.getLogger(UIVideoActivityBuilder.class);
  @Override
  protected void extendUIActivity(BaseUIActivity uiActivity, ExoSocialActivity activity) {
  UIVideoActivity uiVideoActivity = (UIVideoActivity) uiActivity;
  Map<String, String> templateParams = activity.getTemplateParams();
    uiVideoActivity.setLinkSource(templateParams.get(UIVideoActivityComposer.LINK_PARAM));
    uiVideoActivity.setLinkTitle(templateParams.get(UIVideoActivityComposer.TITLE_PARAM));
    uiVideoActivity.setLinkImage(templateParams.get(UIVideoActivityComposer.IMAGE_PARAM));
    uiVideoActivity.setLinkHTML(templateParams.get(UIVideoActivityComposer.HTML_PARAM));
    uiVideoActivity.setLinkComment(templateParams.get(UIVideoActivityComposer.COMMENT_PARAM));
  }
}
```

Note

For the full code, visit YouTube Composer Plugin.

3.2. Overridable Components

There are 2 components in Social that can be overridden: Space Application Handler & Space Service

- **Space Application Handler**

```
<component>
    <key>org.exoplatform.social.core.space.spi.SpaceApplicationHandler</key>
    <type>org.exoplatform.social.core.space.impl.DefaultSpaceApplicationHandler</type>
  </component>
```

- **Space Service**

```
<component>
    <key>org.exoplatform.social.core.space.spi.SpaceService</key>
    <type>org.exoplatform.social.core.space.impl.SpaceServiceImpl</type>
    <init-params>
      <!-- Configure the applications to install in a space -->
      <values-param>
        <name>space.homeNodeApp</name>
        <value>SpaceActivityStreamPortlet</value>
      </values-param>
      <!-- Configure removable application or not <value>Application:removable</value> -->
      <values-param>
        <name>space.apps</name>
        <value>DashboardPortlet:true</value>
        <value>SpaceSettingPortlet:false</value>
        <value>MembersPortlet:true</value>
      </values-param>
    </init-params>
  </component>
```

3.3. Public Java APIs

3.3.1. ActivityManager

Method	Param	Return	Description
saveActivity(Identity owner, ExoSocialActivity activity) throws ActivityStorageException	`owner` - the owner of activity stream, `activity` - the activity which needs to be saved	ExoSocialActivity	Saves an activity to the stream of an owner.Note that the Activity.userId will be set to the owner's identity if it has not already set.
getActivity(String activityId) throws ActivityStorageException	`activityId` - the id of activity	ExoSocialActivity	Gets an activity by its id.
deleteActivity(String activityId) throws ActivityStorageException	`activityId` - the id of activity	void	Deletes an activity by its id.
deleteActivity(ExoSocialActivity activity) throws	`activity`	void	Deletes a stored activity (id != null). (Since 1.1.1).

Method	Param	Return	Description
ActivityStorageException			
deleteComment(String activityId, String commentId) throws ActivityStorageException	`activityId` - the id of activity, `commentId` - the id of comment	void	Deletes a comment by its id.
getActivities(Identity identity) throws ActivityStorageException	`identity`	List<ExoSocialActivity>	Gets the latest activities by an identity with the default limit of 20 latest activities.
getActivities(Identity identity, long start, long limit) throws ActivityStorageException	`identity`, `start` , `limit`	List<ExoSocialActivity>	Gets the latest activities by an identity, specifying **start** that is an offset index and **limit**.
getActivitiesOfConnections(Identity ownerIdentity) throws ActivityStorageException	`ownerIdentity`	List<ExoSocialActivity>	Gets activities of connections from an identity. The activities are returned as a list that is sorted descending by activity posted time.. (Since 1.1.1).
getActivitiesOfConnections(Identity ownerIdentity, int offset, int limit) throws ActivityStorageException;	`ownerIdentity`, `offset`, `limit`	List<ExoSocialActivity>	Gets the activities of connections from an identity by specifying offset and limit. The activities are returned as a list that is sorted starting from the most recent activity.(Since 1.2.0-GA).
getActivitiesOfUserSpaces(Identity ownerIdentity)	`ownerIdentity`	List<ExoSocialActivity>	Gets the activities from all spaces of a user. By default, the activity list is composed of all spaces' activities. Each activity list of the space contains maximum 20 activities and are sorted by time. (Since 1.1.1).
getActivityFeed(Identity identity) throws ActivityStorageException	`identity`	List<ExoSocialActivity>	Gets the activity feed of an identity. This feed is the combination of all the activities of his own activities, his connections' activities and spaces' activities which are returned as a list that is sorted starting

Method	Param	Return	Description
			from the most recent activity.(Since 1.1.2).
saveActivity(ExoSocialActivity activity) throws ActivityStorageException	`activity` - the activity to save	ExoSocialActivity	Saves an activity into the stream for the activity's userId. The userId must be set and this field is used to indicate the owner stream.
saveComment(ExoSocialActivity activity, ExoSocialActivity comment) throws ActivityStorageException	`activity, comment`	void	Saves a new comment or updates an existing comment that is an instance of activity with mandatory fields: userId, title.
saveLike(ExoSocialActivity activity, Identity identity) throws ActivityStorageException	`activity, identity`	void	Saves an identity who likes an activity.
removeLike(ExoSocialActivity activity, Identity identity) throws ActivityStorageException	`activity, identity` - a user who dislikes an activity	void	Removes an indentity who likes an activity, if this activity is liked, it will be removed.
getComments(ExoSocialActivity activity) throws ActivityStorageException	`activity`	List<ExoSocialActivity>	Gets the comment list of an activity.
recordActivity(Identity owner, String type, String title) throws ActivityStorageException	`owner, type, title`	ExoSocialActivity	Records an activity. (Since 1.2.0-GA).
recordActivity(Identity owner, ExoSocialActivity activity) throws Exception	`owner, activity`	ExoSocialActivity	Saves an activity. Deprecated: use ActivityManager#saveActivity(org.exo org.exoplatform.social.core.activity.mo instead. It will be removed by 1.3.x.
recordActivity(Identity owner, String type, String title, String body) throws ActivityStorageException	`owner` - the owner of the target stream for this activity, `type` - the type of an activity which will be used to render a custom ui, `title` - the title, `body` - the body	ExoSocialActivity	Records an activity.
addProcessor(ActivityProcessor processor)	`processor`	void	Adds a new processor.

Method	Param	Return	Description
addProcessorPlugin(BaseActivityProcessorPlugin plugin)	plugin	void	Adds a new processor plugin.
getActivitiesCount(Identity owner) throws ActivityStorageException	`owner`	int	Gets the number of activities from a stream owner.
processActivitiy(ExoSocialActivity activity)	`activity`	void	Passes an activity through the chain of processors.

3.3.2. IdentityManager

Method	Param	Return	Description
registerIdentityProviders(IdentityProviderPlugin plugin)	plugin	void	Registers one or more IdentityProvider through an IdentityProviderPlugin.
getIdentity(String id)	`id` can be a social GlobalId or a raw identity such as in Identity.getId()	Identity - null if nothing is found, or the Identity object	Gets the identity by id and loads his profile.
getIdentity(String id, boolean loadProfile)	`id` can be a social GlobalId or a raw identity such as in Identity.getId(), `loadProfile` - the value is true if the profile is loaded and false if not loaded	null if nothing is found, or the Identity object	Gets the identity by loading id of the profile optionally.
deleteIdentity(Identity identity)	`identiy`	void	Deletes an identity.
addIdentityProvider(IdentityProvider idProvider)	`idProvider`	void	Adds the identity provider.
getOrCreateIdentity(String providerId, String remoteId)	`providerId` - the id of provider, `remoteId` - the remote id	Identity	Gets the identity by remoteId. If the provider can not find any identity by remoteId, the return value is null. If no identity found by identity provider and that identity is still stored on JCR, that stored identity will be deleted and the return value is null.
getOrCreateIdentity(String providerId, String	`providerId` - referring to the name of the Identity	null if nothing is found, or the Identity object	This function returns an Identity object that is

Method	Param	Return	Description
remoteId, boolean loadProfile)	provider, `remoteId` - the identifier that identify the identity in the specific identity provider, `loadProfile` - true when the profile is loaded	improves the performance by specifying what needs to be loaded	specific to a special type. For example, if the type is Linked'In, the identifier will be the URL of profile or if it is a CS contact manager, it will be the UID of the contact. A new identity is created if it does not exist.
getIdentitiesByProfileFilter(String providerId, ProfileFilter profileFilter) throws Exception	`providerId` - the id of provider, `profileFilter` - the filter of provider	Identity	Gets the identities by a profile filter.
getIdentitiesByProfileFilter(String providerId, ProfileFilter profileFilter, long offset, long limit) throws Exception	`providerId`, `profileFilter`, `offset`, `limit`	List<Identity>	Gets the identities by a profile filter.
getIdentitiesByProfileFilter(ProfileFilter profileFilter) throws Exception	`profileFilter` - the profile filter	List<Identity>	Get the identities by profile filter.
getIdentitiesByProfileFilter(ProfileFilter profileFilter, long offset, long limit) throws Exception	`profileFilter`, `offset`, `limit`	List<Identity>	Gets the identities by a profile filter.
getIdentitiesFilterByAlphaBet(String providerId, ProfileFilter profileFilter) throws Exception	`providerId` - the id of provider, `profileFilter` - the profile filter	List<Identity>	Gets the identities filter by alphabet.
getIdentitiesFilterByAlphaBet(String providerId, ProfileFilter profileFilter, long offset,long limit) throws Exception	`providerId`, `profileFilter`, `offset`, `limit`	List<Identity>	Gets the identities fileter by alphabet with offset and limit.
getIdentitiesFilterByAlphaBet(ProfileFilter profileFilter) throws Exception	`profileFilter` - the profile filter	List<Identity>	Gets the identities filter by alphabet.
getIdentity(String providerId, String remoteId, boolean loadProfile)	`providerId`, `remoteId`, `loadProfile`	Identity	Gets the identity.
getIdentitiesCount(String providerId)	`providerId`	long	Gets the number of indentities.
identityExisted(String	`providerId`, `remoteId`	boolean	Checks if the identity is

Method	Param	Return	Description
providerId, String remoteId)			already existed or not.
saveIdentity(Identity identity)	`identity` - the identity	void	Saves the identity.
saveProfile(Profile profile)	`profile`	void	Saves a profile.
addOrModifyProfileProperties(Profile profile) throws Exception	`profile`	void	Adds or modifies properties of profile. Profile parameter is a lightweight that contains only the property that you want to add or modify. NOTE: The method will not delete the properties of an old profile when the param profile does not have those keys.
updateAvatar(Profile p)	`profile`	void	Updates the avatar.
updateBasicInfo(Profile p) throws Exception	`profile`	void	Updates the basic infomation.
updateContactSection(Profile p) throws Exception	`profile`	void	Updates the contact section of the profile.
updateExperienceSection(Profile p)throws Exception	`profile`	void	Updates the experience section of the profile.
updateHeaderSection(Profile p) throws Exception	`profile`	void	Updates the header section of the profile.
getIdentities(String providerId) throws Exception	`providerId` - the id of provider	List<Identity>	Gets the identities by the provider id.
getIdentities(String providerId, boolean loadProfile)	`providerId` - the id of provider, `loadProfile` - the loaded profile.	List<Identity>	Gets the identities by the provider id. If loadProvider is true, loading the profie will be perfomed.
getConnections(Identity ownerIdentity) throws Exception	`ownerIdentity`	List<Identity>	Gets connections of an identity. (Since 1.1.1).
getIdentityStorage()	`N/A`	IdentityStorage	Gets the identity storage.
getStorage()	`N/A`	IdentityStorage	Gets the storage. Deprecated: should use method getIdentityStorage().

Method	Param	Return	Description
registerProfileListener(ProfileListener listener)	listener	void	Registers the profile listener.
unregisterProfileListener(ProfileListener listener)	listener	void	Unregisters the profile listener.
addProfileListener(ProfileListenerPlugin plugin)	plugin	void	Registers a profile listener component plug-in.

3.3.3. RelationshipManager

Method	Param	Return	Description
getRelationshipById(String id) throws Exception	`id`	Relationship	Gets relationship the by id. Deprecated: Use get(String) instead. It will be removed at 1.2.x
invite(Identity sender, Identity receiver) throws RelationshipStorageException	`sender receiver`	Relationship	Creates a connection invitation between two identities.
saveRelationship(Relationship relationship) throws RelationshipStorageException	`relationship` - a relationship	void	Saves a relationship.
confirm(Relationship relationship) throws RelationshipStorageException	`relationship` - a pending relationship	void	Marks a relationship as confirmed.
deny(Relationship relationship) throws RelationshipStorageException	`relationship` - a pending relationship	void	Denies a relationship.
remove(Relationship relationship) throws RelationshipStorageException	`relationship` - a pending relationship	void	Removes a relationship.
ignore(Relationship relationship) throws RelationshipStorageException	`relationship` - a pending relationship	void	Marks a relationship as ignored
getPendingRelationships(Identity sender) throws Exception	`identity` - an identity	List<Relationship>	Gets all the pending relationship of sender.
getPendingRelationships(Identity sender, List<Identity> identities) throws Exception	`identity` - an identity, `identities` - a list of identity	List<Relationship>	Gets pending relationships of sender that match with identities.
getRequireValidationRelationships(Identity receiver) throws	`identity` - identity	List<Relationship>	Gets list of required validation relationship of

Method	Param	Return	Description
Exception			receiver.
getRequireValidationRelationships(Identity receiver, List<Identity> identities)	`identity` - an identity, `identities` - a list of identity	List<Relationship>	Gets list of required validation relationship of receiver that match with identities.
getConfirmedRelationships(Identity identity)	`identity` - an identity	List<Relationship>	Gets list of confirmed relationship of identity.
getConfirmedRelationships(Identity identity, List<Identity> identities)	`identity` - an identity, `identities` - a list of identity	List<Relationship>	Gets list of confirmed relationship of identity that match with identities.
getAllRelationships(Identity identity)	`identity` - an identity	List<Relationship>	Returns all the relationship of a given identity with other identity.
getAllRelationships(Identity identity, List<Identity> identities)	`identity` - an identity, `identities` - a list of identity	List<Relationship>	Returns all the relationship of a given identity with other identity in identities.
getAllRelationships(Identity identity)	`identity` - an identity	List<Relationship>	Returns all the relationship of a given identity with other identity.
getAllRelationships(Identity identity, Relationship.Type type, List<Identity> identities)	`identity` - an identity, `type` - a Relationship.Type, `identities` - a list of identity <Relationship>	Returns all the relationship of a given identity with other identity in identities in type.	
getRelationship(Identity identity1, Identity identity2)	`identity1` and `identity2` - identities	Relationship	Gets the relationship of two identities.

3.3.4. SpaceService

Method	Param	Return	Description
getAllSpaces() throws SpaceException	`N/A`	List<Space> - list of spaces in Social	Gets all spaces in Social.
getSpaceByDisplayName(String spaceDisplayName) throws SpaceException	`spaceDisplayName`	Space	Gets a space by its display name. (Since 1.2.0-GA).
getSpaceByName(String spaceName) throws SpaceException	`spaceName`	Space	Gets a space by its name. Deprecated: Use SpaceService#getSpaceByPrettyName

Method	Param	Return	Description
			instead. It will be removed at 1.3.x
getSpaceByPrettyName(String spaceName) throws SpaceException	`spaceName`	Space	Gets a space by its name. (Since 1.2.0-GA).
getSpacesByFirstCharacterOfName(String firstCharacterOfName) throws SpaceException	`firstCharacterOfName`	List<Space> - all spaces which have first character of name matched the input string.	Gets all spaces has the name starting with the input character.
getSpacesBySearchCondition(String condition) throws Exception	condition(String - the input condition	List<Space> - a list of spaces	Gets all spaces which has the name or the description that matches the input condition.
getSpaceById(String spaceId) throws SpaceException	`spaceId` - Id of that space	Space - space with the id specified	Gets a space by its id.
getSpaceByUrl(String spaceUrl) throws SpaceException	`spaceUrl` - url of space	Space - the space with string url specified	Gets a space by its url.
getSpaces(String userId) throws SpaceException	`userId` - Id of the user	List<Space> - all spaces of a user in which the user is a member	Gets spaces of a user in which that user is a member.
getAccessibleSpaces(String userId) throws SpaceException	`userId`	List<Space> - list of spaces	Gets spaces of a user which that user has the access permission
getEditableSpaces(String userId) throws SpaceException	`userId`	List<Space> - list of space	Gets spaces of a user which that user has the edit permission.
getInvitedSpaces(String userId) throws SpaceException	`userId`	List<Space> - spaces list of all user's invited spaces	Gets a user's invited spaces and that user can accept or deny the request.
getPublicSpaces(String userId) throws SpaceException	`userId - Id of user`	List<Space> - spaces list in which the user can request to join	Gets a user's public spaces and that user can request to join.
getPendingSpaces(String userId) throws SpaceException	`userId`	List<Space> - spaces list in which the user can revoke that request	Gets a user's pending spaces and that the user can revoke that request.
createSpace(Space space, String creator) throws SpaceException	`space, creator`	Space - the created space	Creates a new space by creating a new group.This new group will be under **/Spaces** node.This is shorthand for calling

Method	Param	Return	Description
			createSpace(space, creator, null).
createSpace(Space space, String creator, String groupId) throws SpaceException	`space, creator, groupId` - if groupId == null: create a new space by creating a new group	space	Creates a new space from an existing group.
saveSpace(Space space, boolean isNew) throws SpaceException	`space` - space is saved, `isNew` - true if creating a new space, otherwise, update an existing space.	void	Saves a new space or updates a space.
deleteSpace(Space space) throws SpaceException	`space` - the space is deleted	void	Deletes a space. When deleting a space, all of its page navigations and its group will be deleted.
deleteSpace(String spaceId) throws SpaceException	`spaceId`	void	Deletes a space by its id.
initApp(Space space) throws SpaceException	`space`	void	Initializes default applications in a space. Deprecated: Use **initApps**(Space) instead.
initApps(Space space) throws SpaceException	`space`	void	Initializes default applications in a space. Set **space.homeNodeApp** from the configuration file to be the root page of that space node. When removing a space, make sure to call **deInitApps**(Space) and then deleteSpace(Space) or **deleteSpace**(String)
deInitApps(Space space) throws SpaceException	`space`	void	De-initializes the applications of a space.Make sure to call this method before **deleteSpace**(Space) or **deleteSpace**(String). Otherwise, the space is deleted but its pages and navigation still exists.
addMember(Space space, String userId) throws SpaceException	`space, userId`	void	Adds a user to a space, the user will get the "member" role in a space

Method	Param	Return	Description
addMember(String spaceId, String userId) throws SpaceException	`spaceId, userId`	void	Adds a user to a space, the user will get the "member" role in a space
removeMember(Space space, String userId) throws SpaceException	`space, userId`	void	Removes a member from a space. If the member is the only leader of that space, the member removed is not allowed and throws **SpaceException** with Code = USER*ONLY*LEADER.
removeMember?(String spaceId, String userId) throws SpaceException	`spaceId, userId`	void	Removes a member from a space.
getMembers(Space space) throws SpaceException	`space`	List<String> - a list of the space members	Gets a list of the space members from a space.
getMembers(String spaceId) throws SpaceException	`spaceId`	List<String> - a list of the space members	Gets a list of the space members from a space.
setLeader(Space space, String userId, boolean isLeader) throws SpaceException	`space, userId, isLeader`	void	Sets a member of a space as a manager.
setLeader(String spaceId, String userId, boolean isLeader) throws SpaceException	`spaceId, userId, isLeader`	void	Sets a member of a space as a manager.
isLeader(Space space, String userId) throws SpaceException	`space, userId`	boolean - true if that the user is a leader, otherwise, false	Checks whether a user is a space's leader or not.
isLeader(String spaceId, String userId) throws SpaceException	`spaceId, userId`	boolean - true if that user is a leader, otherwise, false	Checks whether a user is a space's leader or not.
isOnlyLeader(Space space, String userId) throws SpaceException	`space, userId`	boolean - true if that user is the only leader of the space, otherwise, false	Checks whether a user is the only leader of a space or not.
isOnlyLeader(String spaceId, String userId) throws SpaceException	`spaceId, userId`	boolean	Checks whether a user is the only leader of a space or not.
isMember(Space space, String userId) throws SpaceException	`space, userId`	boolean - true if that user is a member, otherwise, false	Checks whether a user is a space's member or not.

Method	Param	Return	Description
isMember(String spaceId, String userId) throws SpaceException	`spaceId, userId,`	boolean - true if that user is a member, otherwise,false	Checks whether a user is a space's member or not.
hasAccessPermission(Space space, String userId) throws SpaceException	`space, userId`	boolean - true If the user is root or the space's member.	Checks if a user can access a space or not.
hasAccessPermission(String spaceId, String userId) throws SpaceException	`spaceId, userId`	boolean - true If the user is root or the space's member	Checks if a user can access a space or not.
hasEditPermission(Space space, String userId) throws SpaceException	`space, userId`	boolen - true If the user is root or the space's manager	Checks if a user can have the edit permission of a space or not.
hasEditPermission(String spaceId, String userId) throws SpaceException	`spaceId, userId`	boolen - true If user is root or the space's manager	Checks if a user can have the edit permission of a space or not.
isInvited(Space space, String userId) throws SpaceException	`space, userId`	boolen - true if that user is in the invited list, otherwise, false	Checks if a user is in the invited list of a space or not.
isInvited(String spaceId, String userId) throws SpaceException	`spaceId, userId`	boolen - true if the user is in the invited list, otherwise, false	Checks if a user is in the invited list of a space or not.
isPending(Space space, String userId) throws SpaceException	`space, userId`	boolen - true if that user is in the pending list, otherwise, false	Checks if a user is in the pending list of a space or not.
installApplication(String spaceId, String appId) throws SpaceException	`spaceId, appId`	void	Installs an application to a space.
installApplication(Space space, String appId) throws SpaceException	`space, appId`	void	Installs an application to a space
activateApplication(Space space, String appId) throws SpaceException	`space, appId`	void	Activates an installed application in a space.
activateApplication(String spaceId, String appId) throws SpaceException	`spaceId, appId`	void	Activates an installed application in a space.
deactivateApplication(Space space, String appId) throws SpaceException	`space, appId`	void	Deactivates an installed application in a space.
deactivateApplication(String spaceId, String appId) throws SpaceException	`spaceId, appId`	void	Deactivates an installed application in a space.

Method	Param	Return	Description
removeApplication(Space space, String appId, String appName) throws SpaceException	`space, appId, appName`	void	Removes an installed application from a space.
removeApplication(String spaceId, String appId, String appName) throws SpaceException	`space, appId, appName`	void	Removes an installed application from a space.
requestJoin(Space space, String userId) throws SpaceException	`space, userid`	void	Requests a user to join a space, adds that user to the pending list of the space.
requestJoin(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Requests a user to join a space, adds that user to the pending list of the space.
revokeRequestJoin(Space space, String userId) throws SpaceException	`space, userid`	void	Revokes a join request after users request to join a group and is in the pending status.
revokeRequestJoin(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Revokes a request to join a space.
inviteMember(Space space, String userId) throws SpaceException	`space, userid`	void	Invites a userId to become a member of a space.
inviteMember(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Invites a userId to a be member of a space.
revokeInvitation(Space space, String userId) throws SpaceException	`space, userid`	void	Revokes an invitation. Removes a user from the invited member list of the space.
revokeInvitation(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Revokes an invitation. Removes a user from the invited member list of the space.
acceptInvitation(Space space, String userId) throws SpaceException	`space, userid`	void	Accepts an invitation and moves a user from the invited list to the member list.
acceptInvitation(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Accepts an invitation and moves a user from the invited list to the member

Method	Param	Return	Description
			list.
denyInvitation(Space space, String userId) throws SpaceException	`space, userid`	void	Denies an invitation and removes a user from the invited list.
denyInvitation(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Denies an invitation and removes a user from the invited list.
validateRequest(Space space, String userId) throws SpaceException	`space, userid`	void	Validates a request and moves a user from the pending list to the member list.
validateRequest(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Validates a request and moves a user from the pending list to the member list.
declineRequest(Space space, String userId) throws SpaceException	`space, userid`	void	Declines a request and removes a user from the pending list.
declineRequest(String spaceId, String userId) throws SpaceException	`spaceId, userid`	void	Declines a request and removes a user from the pending list.
registerSpaceLifeCycleListener(SpaceLifeCycleListener listener	listener(SpaceLifeCycleListener	void	Registers a space lifecycle listener.
unregisterSpaceLifeCycleListener(SpaceLifeCycleListener listener)	Listener(SpaceLifeCycleListener	void	Unregisters a space lifecycle listener.
setPortletsPrefsRequired(PortletPreferencesRequiredPlugin portletPrefsRequiredPlugin)	portletPreferencesRequiredPlugin	void	Sets the portlet preferences got from the plug-in configuration.
getPortletsPrefsRequired()	`N/A`	String	Gets the portlet preferences required to use in creating the portlet application.
setSpaceApplicationConfigPlugin(SpaceApplicationConfigPlugin spaceApplicationConfigPlugin)	SpaceApplicationConfigPlugin	void	Sets configuration plugin for initializing applications when creating a new space. (Since 1.2.0-GA.
getSpaceApplicationConfigPlugin()	N/A	SpaceApplicationConfigPlugin	Gets the configuration of applications to be initialized when creating a new space. (Since 1.2.0-GA).

3.4. Java APIs sample code/ tutorial

3.4.1. Activity Stream

eXo Social provides users with a way to share their activity information (also known as Activity Stream) and collaborate in spaces (also known as group work). With the API, you can customize the way to display activities or publish new ones. To manipulate activities, you need to use the `AtivityManager` service.

3.4.1.1. Publish an activity

There are two types of activities: activities for a user and activities for a space. The following examples will show you how to publish an activity for each type.

3.4.1.1.1. Publish an activity for a user

```
package org.exoplatform.publish.user;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;

public class PublishActivityForUser {
  // Exo Log.
  private final Log LOG = ExoLogger.getLogger(PublishActivityForUser.class);

  // Portal container.
  private PortalContainer container;

  // identityManager manages identities.
  private IdentityManager identityManager;

  // activityManager manages activities.
  private ActivityManager activityManager;

  private final static String DEFAULT_USER_NAME = "zun";
  private final static String DEFAULT_ACTIVITY_TITLE = "Hello World!";

  /**
   * Constructor.
   */
  public PublishActivityForUser() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets identityManager to handle an identity operation.
    identityManager = (IdentityManager) container.getComponentInstance(IdentityManager.class);

    // Gets activityManager to handle an activity operation.
    activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);
  }

  public void createActivityForUser() {
    try {
      // Gets an existing identity or creates a new one.
      Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEFAULT_USE

      // Creates a new activity for this user.
      ExoSocialActivity activity = new ExoSocialActivityImpl();
      activity.setUserId(userIdentity.getId());
      activity.setTitle(DEFAULT_ACTIVITY_TITLE);
```

```
      // Saves an activity into JCR by using ActivityManager.
      activityManager.saveActivity(activity);
    } catch (Exception e) {
      LOG.error("can not save activity.", e);
    }
  }
}
```

3.4.1.2. Publish an activity for a space

```
package org.exoplatform.publish.space;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;

import org.exoplatform.social.core.space.model.Space;
import org.exoplatform.social.core.space.SpaceException;
import org.exoplatform.social.core.space.spi.SpaceService;
import org.exoplatform.social.core.identity.provider.SpaceIdentityProvider;

public class PublishActivityForSpace {
  // Exo Log.
  private final Log LOG = ExoLogger.getLogger(PublishActivityForSpace.class);

  // Portal container.
  private PortalContainer container;

  // identityManager manages identities.
  private IdentityManager identityManager;

  // activityManager manages activities.
  private ActivityManager activityManager;

  // spaceService manages spaces.
  private SpaceService spaceService;

  private final static String DEFAULT_NAME_SPACE = "mySpace";
  private final static String DEFAULT_USER_NAME = "zun";
  private final static String DEFAULT_ACTIVITY_TITLE = "An activity for space";

  /**
   * Constructor method.
   */
  public PublishActivityForSpace() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets identityManager to manage identities.
    identityManager = (IdentityManager) container.getComponentInstance(IdentityManager.class);

    // Gets activityManager to manage activities.
    activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);

    // Gets spaceService to handle the operation of a space.
    spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);
  }

  public void createActivityForSpace() {
    try {
      // make sure that a space with the name "mySpace" is created.
      Space space = spaceService.getSpaceByDisplayName(DEFAULT_NAME_SPACE);
      if (space != null) {
        // Gets spaceIdentity if it already exists. If not, a new one is created.
        Identity spaceIdentity = identityManager.getOrCreateIdentity(SpaceIdentityProvider.NAME, DEFAULT_NAME_SP
        // Gets an identity if it already exists. If not, a new one is created.
```

```
        Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEFAULT_U
        // Creates a new activity for this space.
        ExoSocialActivity activity =  new ExoSocialActivityImpl();
        activity.setUserId(userIdentity.getId());
        activity.setTitle(DEFAULT_ACTIVITY_TITLE);
        activityManager.saveActivity(spaceIdentity, activity);
      }
    } catch (SpaceException e) {
      LOG.error("Can not save activity.", e);
    } catch (Exception e) {
      LOG.error("Can not save activity.", e);
    }
  }
}
```

3.4.1.2.1. Configure an activity processor

An activity processor is used to modify the content of activities before they are returned from activity manager. For example, create an activity processor to replace all the texts representing the smile face ":-)" in the activity title by the smiley icons, do as follows:

Firstly, create the `SmileyProcessor` class by extending the `BaseActivityProcessorPlugin`.

```
package org.exoplatform.social.core.activitystream;

import org.exoplatform.social.core.BaseActivityProcessorPlugin;
import org.exoplatform.container.xml.InitParams;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;

public class SmileyProcessor extends BaseActivityProcessorPlugin {
  private String smiley;

  public SmileyProcessor(InitParams params) {
    super(params);
    this.smiley = "<img src={{{"}}}http://www.tombraider4u.com/pictures/smiley.gif{{{"}}}/>";
  }

  @Override
  public void processActivity(ExoSocialActivity activity) {
    String title = activity.getTitle();
    activity.setTitle(title.replaceAll(":-)", this.smiley));
  }
}
```

Then, register this processor by editing the *configuration.xml* file:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.manager.ActivityManager</target-component>
  <component-plugin>
    <name>SmileyProcessor</name>
    <set-method>addProcessorPlugin</set-method>
    <type>org.exoplatform.social.core.activitystream.SmileyProcessor</type>
    <description/>
    <init-params>
      <values-param>
        <name>priority</name>
        <value>1</value>
      </values-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

"init-params" contains all the key-value data which a processor will use to initialize. In the above configuration,

the **priority** value indicates the order in which this processor is executed. If the value is 1, this processor will be used before all the remaining processors with the lower priority.

3.4.1.2.2. Publish an RSS feed with feedmash

It is really easy to publish an RSS feed to a space's activity stream. eXo Social provides `FeedmashJobPlugin` to publish RSS feeds. As you can see in the project "`exo.social.extras.feedmash`", there are the `JiraFeedConsumer` and `HudsonFeedConsumer` samples to post eXo Social project's feeds (jira and hudson) to a pre-defined space named `exosocial` in a specific portal container named`socialdemo` as in the configuration file:

```
<external-component-plugins>
   <target-component>org.exoplatform.services.scheduler.JobSchedulerService</target-component>
    <component-plugin>
      <name>RepubSocialJiraActivityJob</name>
      <set-method>addPeriodJob</set-method>
      <type>org.exoplatform.social.extras.feedmash.FeedmashJobPlugin</type>
      <description/>
      <init-params>
        <properties-param>
          <name>mash.info</name>
          <property name="feedURL" value="http://jira.exoplatform.org/plugins/servlet/streams?key=SOC"/>
          <property name="categoryMatch" value="resolved|created"/>
          <property name="targetActivityStream" value="space:exosocial"/>
          <property name="portalContainer" value="socialdemo"/>
        </properties-param>
        <properties-param>
          <name>job.info</name>
          <description>save the monitor data  periodically</description>
          <property name="jobName" value="JIRAFeedConsumer"/>
          <property name="groupName" value="Feedmash"/>
          <property name="job" value="org.exoplatform.social.feedmash.JiraFeedConsumer"/>
          <property name="repeatCount" value="0"/>
          <property name="period" value="60000"/>
          <property name="startTime" value="+45"/>
          <property name="endTime" value=""/>
        </properties-param>
      </init-params>
    </component-plugin>
    <component-plugin>
      <name>WatchSocialBuildStatus</name>
      <set-method>addPeriodJob</set-method>
      <type>org.exoplatform.social.extras.feedmash.FeedmashJobPlugin</type>
      <description/>
      <init-params>
        <properties-param>
          <name>mash.info</name>
           <property name="feedURL" value="http://builder.exoplatform.org/hudson/view/social/job/social-trunk-ci
           <property name="targetActivityStream" value="space:exosocial"/>
           <property name="portalContainer" value="socialdemo"/>
        </properties-param>
        <properties-param>
          <name>job.info</name>
          <description>save the monitor data  periodically</description>
          <property name="jobName" value="HudsonFeedConsumer"/>
          <property name="groupName" value="Feedmash"/>
          <property name="job" value="org.exoplatform.social.feedmash.HudsonFeedConsumer"/>
          <property name="repeatCount" value="0"/>
          <property name="period" value="60000"/>
          <property name="startTime" value="+100"/>
          <property name="endTime" value=""/>
        </properties-param>
      </init-params>
    </component-plugin>
 </external-component-plugins>
```

Run eXo Social with the URL: http://localhost:8080/socialdemo, then log in and create a space named "exosocial". After creating the "exosocial" space, all the feeds of the eXo Social project on Jira and Hudson

will be automatically published to the `exosocial` space.

3.4.1.3. Sample Code

See the following code snippet to know more details how to publish an activity and add comments to an activity:

```
package org.exoplatform.social.introduction.activitystreamandexosocialactivity;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.activity.model.ActivityStream;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;

public class IntroduceActivityStreamAndExoSocialActivity {
  // Exo Log.
  private final Log LOG = ExoLogger.getLogger(IntroduceActivityStreamAndExoSocialActivity.class);

  // Demo identity.
  private Identity demoIdentity;

  // John identity.
  private Identity johnIdentity;

  // identityManager manages identities.
  private IdentityManager identityManager;

  // activityManager manages activities.
  private ActivityManager activityManager;

  // Portal container.
  private PortalContainer container;

  private final static String DEMO_NAME = "demo";
  private final static String JOHN_NAME = "john";
  private final static String DEFAULT_ACTIVITY_TITLE = "blabla";
  private final static String DEFAULT_COMMENT_TITLE = "comment blah blah";

  /**
   * Constructor.
   */
  public IntroduceActivityStreamAndExoSocialActivity() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets IdentityManager to handle an identity operation.
    identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class);

    // Gets ActivityManager to handle activity operation.
    ActivityManager activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.cla

   // Gets or create demo's identity
   demoIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEMO_NAME, false);

    // Gets or creates the identity "john".
    johnIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, JOHN_NAME, false);
  }

   /**
   * Posts activity to activity stream
   */
   public void introduceActivityStreamAndExoSocialActivity {

    // Sets a string that specifies the primary text of an activity. This field is REQUIRED by ActivityManager.
    activity.setTitle(DEFAULT_ACTIVITY_TITLE);

    // Sets this activity for demo.
   activity.setUserId(demoIdentity.getId());
```

```
    // Saves the activity.
    activityManager.saveActivity(johnIdentity, activity);

    // Gets activity stream.
    ActivityStream activityStream = activity.getActivityStream();

    // Type of the activity stream. It can be organization or space.
    LOG.info("activity stream type: " + activityStream.getType());

    LOG.info("activity stream id: " + activityStream.getId());
    LOG.info("activity stream pretty id: " + activityStream.getPrettyId());
    LOG.info("activity stream perma link: " + activityStream.getPermaLink());

    LOG.info("activity stream id: " + activity.getStreamId());
    LOG.info("activity stream owner: " + activity.getStreamOwner());

    // Comment in Social
    ExoSocialActivity demoActivity = new ExoSocialActivityImpl();;
    activity.setTitle(DEFAULT_ACTIVITY_TITLE);
    activityManager.saveActivity(demoIdentity, demoActivity);

    ExoSocialActivity comment = new ExoSocialActivityImpl();;
    comment.setTitle(DEFAULT_COMMENT_TITLE);

    //Sets comment of demo
   comment.setUserId(demoIdentity.getId());

    //Saves a comment.
    activityManager.saveComment(activity, comment);
  }
}
```

3.4.2. OpenSocial

eXo Social supports the OpenSocial standard. So you can integrate OpenSocial gadgets in your dashboard and use the RPC or REST service to view or publish the social data. With the support for the OpenSocial standard, eXo Social provides a framework for developers to build gadgets that can display and mash-up activity information for contacts, social networks, applications and services.

3.4.2.1. Gadget

Gadgets are web-based software components based on HTML, CSS, JavaScript, defined using a declarative XML syntax. They allow developers to easily write useful social applications that work on the social networks supporting OpenSocial APIs without modification. See the following links for detailed information:

- Gadgets Specification
- OpenSocial Core Gadget Specification v0.8
- Gadgets APIs

To know how to create an OpenSocial gadget, see here

Note

OpenSocial gadgets will work out of the box on Dashboard. Morever, OpenSocial gadgets will be wrapped by GadgetWrapperPortlet so they can work in any pages on eXo Social, too. At present, eXo Social is supporting OpenSocial version 0.8.

3.4.2.1.1. Supported APIs

eXo Social leverages Apache Shindig - an OpenSocial reference implementation to provide and extend OpenSocial APIs which is compatible with the common OpenSocial APIs which is supported by other big social networks like Ning, Hi5, Orkut and more.

To get more details about Supported APIs, refer to Opensocial Specs.

3.4.2.1.1.1. REST/RPC API

Suppose that you are running local host at port 8080 (http://localhost:8080/), then the path of the API will be:

- REST API:

http://localhost:8080/social/social/rest

- RPC API:

http://localhost:8080/social/social/rpc

To learn what you can do with this APIs, have a look at the specification. If you are developing in Java, you can use the opensocial-java-client.

3.4.2.1.1.2. Configure the security

If you are using OpenSocial, you need to configure the OAuth authentication. With the case of eXo Platform, you need to edit the file: gatein/conf/portal/portal/configuration.xml and add the following configuration:

```
<component>
        <key>org.exoplatform.social.opensocial.oauth.ServiceProviderStore</key>
        <type>org.exoplatform.social.opensocial.oauth.ServiceProviderStore</type>
                        <init-params>
                <properties-param>
                        <name>grails-book-flow</name>
                        <description>consmer key and secret for sample oauth provider. </description>
                        <property name="consumerKey" value="YOUR_KEY_HERE"/>
                        <property name="sharedSecret" value="YOUR_SECRET_KEY_HERE"/>
                                </properties-param>
                    </init-params>
</component>
```

The consumerKey and sharedSecret are the key that need to be shared with the application that is doing the request.

3.4.2.1.1.3. Publish an activity into a space

eXo Social adds this functionality that is not available in the standard OpenSocial APIs. You can publish activities into a space using the OpenSocial APIs.

Instead of publishing your activities to the group @self as usual, publish them to the group "space:spaceID" or "space:spacePrettyName".

Use the OpenSocial java library and groovy, your code will look like this:

```
def client = getOpenSocialClient()

//we create our new activity
```

```
      Activity activity = new Activity()
          activity.body = "xx puchased the book xxx"
      activity.title = "BookFlow Puchased"

      //We prepare the request that will create the activity
      Request request = ActivitiesService.createActivity(activity);
      //We specify that the creation of this new activity is for the space bookflow
      request.groupId = "space:bookflow";

      client.send(request);
```

In the example above, the groupId is set to "space:bookflow" and bookflow is the name of the space.

3.4.2.2. Tutorial

- See Grails + eXo Social tutorial

3.4.3. People

eXo Social provides a way to add profile information, connections and connections between users called People. With the eXo People API, profile information and relationship can be easily managed and customized.

3.4.3.1. Identity

The identity allows identifying a unique social object. Social objects can be persons, groups, applications, or whatever is related to social interactions like connecting, publishing an activity stream or holding a profile.

3.4.3.1.1. IdentityProvider

eXo Social provides IdentityProvider as a identiy mechanism for the third parties to extend Social's identity system without any limitation. Here is an example:

```
class SampleIdentityProvider extends OrganizationIdentityProvider{
  public SampleIdentityProvider(OrganizationService organizationService) {
    super(organizationService);
  }

  @Override
  public void populateProfile(Profile profile, User user) {
    profile.setProperty(Profile.FIRST_NAME, "this is first name");
    profile.setProperty(Profile.LAST_NAME, "this is last name");
    profile.setProperty(Profile.USERNAME, "this is user name");
    profile.setProperty(Profile.URL, "/path/to/profile/");
  }
}
```

In this example, the SampleIdentityProvider class extends OrganizationIdentityProvider which is the IdentityProvider used to connect to the portal user's base. In this class, the populateProfile method is overridden and some dummy data are added to the profile fields.

3.4.3.1.2. IdentityManager

The IdentityManager is the service used to manipulate the identity operations like creating, getting, deleting or finding a profile. You can get the IdentityManager via the ExoContainer. The following code will show how to get an IdentityManager instance and create a basic identity instance:

```
import org.exoplatform.container.ExoContainer;
import org.exoplatform.container.ExoContainerContext;
```

```
import org.exoplatform.social.core.identity.IdentityManager;
import org.exoplatform.social.core.identity.impl.organization.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;

//......

String containerName = "portal";
String username = "zun";

//get container to get other registered components
ExoContainer container = ExoContainerContext.getContainerByName(containerName);

//get IdentityManager to handle identity operation
IdentityManager identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class);

//get ActivityManager to handle activity operation
ActivityManager activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);

//create new user with name Zun
Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, username);
```

3.4.3.2. ProfileListener

The People API provides notification interfaces which you can implement to create your own handlers for notification when having profile modifications by extending the ProfileListenerPlugin class or relationship changes by extending RelationshipListenerPlugin.

- Firstly, create the ProfileLoggerListener class to log all profile modifications of the systems. The abstract class ProfileListenerPlugin provides the interface to implement this method.

```
import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;
import org.exoplatform.social.core.identity.lifecycle.ProfileListenerPlugin;
import org.exoplatform.social.core.identity.spi.ProfileLifeCycleEvent;


public class ProfileLoggerListener extends ProfileListenerPlugin{
  private static final Log logger = ExoLogger.getExoLogger(ProfileLoggerListener.class);
  @Override
  public void avatarUpdated(ProfileLifeCycleEvent event) {
    logger.info("@" + event.getUsername() + " profile has updated his basic profile info.");
  }

  @Override
  public void basicInfoUpdated(ProfileLifeCycleEvent event) {
    logger.info("@" + event.getUsername() + " profile has updated his basic profile info.");
  }

  @Override
  public void contactSectionUpdated(ProfileLifeCycleEvent event) {
    logger.info("@" + event.getUsername() + " profile has updated his contact info.");
  }

  @Override
  public void experienceSectionUpdated(ProfileLifeCycleEvent event) {
    logger.info("@" + event.getUsername() + " profile has an updated experience section.");
  }

  @Override
  public void headerSectionUpdated(ProfileLifeCycleEvent event) {
    logger.info("@" + event.getUsername() + " has updated his header info.");
  }
}
```

- Then, add some configurations for this class to the configuration.xml:

```
<external-component-plugins>
```

```
  <target-component>org.exoplatform.social.core.identity.IdentityManager</target-component>
  <component-plugin>
    <name>ProfileLoggerListener</name>
    <set-method>addProfileListener</set-method>
    <type>path.to.ProfileLoggerListener</type>
  </component-plugin>
</external-component-plugins>
```

3.4.3.3. Connections

Similarly, you can apply the above steps to implement the RelationshipListenerPlugin for relationship notifications.

Relationship is the bridge between two identities in eXo Social. There are many types of relationships defined in the Relationship class. With these types, a user can invite another user, confirm invitations or remove relationship.

3.4.3.3.1. Users connection

The following code will show how to invite a user to connect to another :

```
import org.exoplatform.container.ExoContainer;
import org.exoplatform.container.ExoContainerContext;
import org.exoplatform.social.core.identity.IdentityManager;
import org.exoplatform.social.core.identity.impl.organization.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.relationship.Relationship;
import org.exoplatform.social.core.relationship.RelationshipManager;

public void inviteUser() throws Exception {
  String containerName = "portal";
  ExoContainer container = ExoContainerContext.getContainerByName(containerName);

  IdentityManager identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class

  Identity invitedIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, "Hoat");
  String invitedUserId = invitedIdentity.getId();

  String currUserId = "Zun";
  Identity currentIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME,  currUserId)

  RelationshipManager relationshipManager = (RelationshipManager) container.getComponentInstanceOfType(Relations
  Relationship relationship = relationshipManager.invite( currentIdentity, invitedIdentity);
}
```

3.4.3.3.2. RelationshipListener

The following example will show you how to implement one of these interfaces and how to configure this plugin. The following step is similar to the Profile notifications implementation.

- Firstly, create the RelationshipLoggerListener class:

```
import org.exoplatform.social.core.relationship.Relationship;
import org.exoplatform.social.core.relationship.lifecycle.RelationshipListenerPlugin;
import org.exoplatform.social.relationship.spi.RelationshipEvent;

public class RelationshipLoggerListener extends RelationshipListenerPlugin{
  private static final Log logger = ExoLogger.getExoLogger(RelationshipLoggerListener.class);

  @Override
  public void confirmed(RelationshipEvent event) {
    String[] names = getUserNamesFromEvent(event);
    logger.info(names[0] +" confirmed the invitation of "+ names[1]);
  }
```

```
  @Override
  public void ignored(RelationshipEvent event) {
    String[] names = getUserNamesFromEvent(event);
    logger.info(names[0] +" ignored the invitation of "+ names[1]);
  }

  @Override
  public void removed(RelationshipEvent event) {
    String[] names = getUserNamesFromEvent(event);
    logger.info(names[0] +" removed the relationship with "+ names[1]);
  }

  private String[] getUserNamesFromEvent(RelationshipEvent event){
    Relationship relationship = event.getPayload();

    Identity id1 = relationship.getIdentity1();
    Identity id2 = relationship.getIdentity2();

    String user1 = "@" + id1.getRemoteId();
    String user2 = "@" + id2.getRemoteId();

    return new String[]{user1, user2 };
  }
}
```

- Then, add some configurations for this class to the configuration.xml:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.relationship.RelationshipManager</target-component>
  <component-plugin>
    <name>RelationshipLoggerListener</name>
    <set-method>addListenerPlugin</set-method>
    <type>classpath.of.your.RelationshipLoggerListener</type>
  </component-plugin>
</external-component-plugins>
```

3.4.4. Spaces

eXo Social provides a way to create groups and to share data and applications by spaces. A space has its own activity stream in which applications or members can publish information. In each space, members use applications together.

To manipulate the spaces, you will use the SpaceService. To get an instance of this class, you need to get the current PortalContainer instance.

3.4.4.1. Spaces Management

The following example shows how to create a space:

```
package org.exoplatform.social.sample;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.application.impl.DefaultSpaceApplicationHandler;
import org.exoplatform.social.space.Space;
import org.exoplatform.social.space.SpaceException;
import org.exoplatform.social.space.SpaceService;

public class SpaceCreationSample {

  public void createSpace() throws SpaceException {
    String spaceName = "mySpace";
    String creator = "jeremi";
    PortalContainer container = PortalContainer.getInstance();
    SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);
```

```
    // We verify if there is no space already created
    Space space = spaceService.getSpaceByDisplayName(spaceName);
    if (space == null) {
      space = new Space();
      space.setDisplayName(spaceName);
      space.setRegistration(Space.OPEN);
      space.setDescription("space description");
      //DefaultSpaceApplicationHander is the default implementation of SpaceApplicationHandler. You can create y
      space.setType(DefaultSpaceApplicationHandler.NAME);
      //We create the space
      space = spaceService.createSpace(space, creator);
      //We initialize the applications
      spaceService.initApps(space);

    }
  }
}
```

3.4.4.2. Space's applications management

3.4.4.2.1. Add an application to a space

You can add portlet or gadget applications into spaces. Once added, all members of the space can use that application. The following code shows how to add an application to a space :

```
public void addApplicationToSpace() throws SpaceException {
  //Your portlet name
  String appId = "SocialBannerPortlet";
  String spaceId = "zunSpace";

  //get container to get other registered components
  PortalContainer container = PortalContainer.getInstance();

  //get space service for installing operations
  SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);

  //install application for the space
  spaceService.installApplication(spaceId, appId);

  //we must activate installed application to be able to use it
  spaceService.activateApplication(spaceId, appId);
}
```

Note

appId is the portlet or gadget name as defined in portlet.xml or gadget.xml in the web-app. With eXo Social, you can find it in social-portlet.war and social.war.

3.4.4.2.2. Remove an application from a space

You can remove portlet or gadget applications from a space. The members of this space cannot see them anymore. The following code shows how to remove an application from a space :

```
public void removeApplicationFromSpace() throws SpaceException {
  //Your portlet name
  String appId = "SocialBannerPortlet";
  String appName = "SocialBannerPortlet1"
  String spaceId = "zunSpace";

  //get container to get other registered components
  PortalContainer container = PortalContainer.getInstance();

  //get space service for installing operations
  SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);
```

```
  //install application for the space
  spaceService.removeApplication(spaceId, appId, appName);
}
```

3.4.4.3. Space's members management

The SpaceService allows you to manage the spaces' members. Here is the way to add a new member to a space:

```
String spacePrettyName = "mySpace";

PortalContainer container = PortalContainer.getInstance();
SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);

Space space = service.getSpaceByPrettyName(spacePrettyName);
if (space != null) {
    spaceService.addMember(space, "mary");
}
```

3.4.4.4. Listener to a space lifecycle

To receive notifications of what are happening in spaces, you need to extend SpaceListenerPlugin and register it. Every method takes a SpaceLifeCycleEvent object as a parameter that contains the information about the event and its context.

The available events are:

- spaceCreated: This event is called right after a space is created successfully with its applications.
- spaceRemoved: This event is called right after the space is removed. It means all the applications of the space are removed, its group and group navigation are removed.
- applicationAdded: This event is called right after an application is added (installed) to the space.
- applicationActivated: This event is called right after an application is activated in the space.
- applicationDeactivated: This event is called right after an application is deactivated.
- applicationRemoved: This event is called right after an application is removed from the space.
- joined: This event is called right after a user joins the space.
- left: This event is called right after a space member leaves its space.
- grantedLead: This event is called right after a user is grated the space's manager role.
- revokedLead: This event is called right after the space's manager is revoked his role to be a space member.

```
import org.exoplatform.social.space.lifecycle.SpaceListenerPlugin;

public class MySpaceListenerPlugin extends SpaceListenerPlugin {
...
}
```

As an example, see the SpaceActivityPublisher that publishes an activity based on an event that happened in a space.

To register your listener, configure it as a plugin to the SpaceService like this :

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
  <component-plugin>
    <name>SpaceActivityPublisher</name>
    <set-method>addSpaceListener</set-method>
    <type>org.mycompany.MySpaceListenerPlugin</type>
  </component-plugin>
</external-component-plugins>
```

3.4.5. Space widget tutorial

The eXo Social widget enables developers to add capabilities of eXo Social to external applications. Since the widget is hosted on your eXo Social server, it can display personalized information. An activity stream of the most recent user actions will display to the group's members.

There are two options of the eXo Social widget that provide different levels of integration and information.

The basic version of this widget is an iFrame. The more advanced version is a button you can insert in a page; this will display a small pop-up with information about the space.

3.4.5.1. Basic version

To insert the basic version, you need to have an iFrame to insert on your site.

```
<iframe  scrolling="no" height="180" frameborder="no" width="220" src="http://URL_OF_YOUR_EXO_INSTALLATION.COM/r
spaceName=NAME_OF_YOUR_SPACE&description=DESCRIPTION_OF_THE_SPACE"></iframe>
```

To install this version in your application, replace all the uppercase text below:

- URL_OF_YOUR_EXO_INSTALLATION.COM: This is the URL of your eXo Social installation. If you are testing on your local computer, the URL may be *localhost:8080_*
- NAME_OF_YOUR_SPACE: This is the title of your space in eXo Social. In the URL, it is necessary to avoid special characters. For the space name, you can only use alphanumeric characters and "_", ".", "-" or ".".
- DESCRIPTION_OF_THE_SPACE: This will be displayed in the list of spaces.

3.4.5.2. Advanced version

To install an advanced version of the widget, you need to insert a code snippet in your page. This includes some HTMLs plus some JavaScript. The necessary CSS will be added dynamically.

Next, insert the following code at the position you want the button to be displayed:

```
<div class="exoSpacesContainer"><a href="javascript:void(0);" id="exoSpacesLink" class="exoSpacesLink" target="_
<script src="/socialWidgetResources/javascript/space.js"></script>
<script>spaces.createPopup("exoSpacesLink", "MyAppName - my social object", "my cool description");</script>
```

The important function here is:

```
spaces.createPopup(link, spaceName, description)
```

- link: link is the ID or the HTMLElement where the pop-up will be placed. If you copy and paste the code snippet provided above, you do not need to change this value.
- spaceName: This is the name of space. It is also used to identify the space. We recommend you use the following format: "MyAppName - my social object"
- description: This is the description of your space that will be displayed within eXo Spaces. It is used when a new space is created.

3.4.5.3. Configure

- serverURL (Default: *"http://127.0.0.1:8080"*): The address of your eXo installation. To change it, use *spaces.setServerURL(...)*.
- spaceServicePath (Default: *"/rest/private/spaces/"*): The path to the spaces service. It is rare you have to change it; but if needed, use *spaces.setSpaceServicePath(...)*.
- portalName (Default: *"socialdemo"*): The name of portal you are using. To change it, use *spaces.setPortalName(...)*.

If you want to change any part of this configuration, the best way is to change before creating the pop-up. For example:

```
<div class="exoSpacesContainer"><a href="#" id="exoSpacesLink" class="exoSpacesLink" target="_blank">Space</a></
<script src="/socialWidgetResources/javascript/space.js"></script>
<script>
  spaces.setServerURL("http://192.168.2.100:8080");
  spaces.createPopup("exoSpacesLink", "My cool new space", "my cool description");
</script>
```

You can see an example of integration at: http://localhost:8080/socialWidgetResources/test.html

3.4.6. How to extend the activities rendering

3.4.6.1. Objective

In this article you will learn how you can implement a preprocessor for activities rendering.

3.4.6.2. Requirements

You need to be familiar with the eXo Kernel component model and its xml configuration. You also need be able to program in java.

3.4.6.3. Why would you need to do this ?

An simplistic activity is made of simple text. We have created an extension point at the level of activities renderting for two cases :

- support more html tags
- support @mentions

But you may want to support a special syntax like :

- #hashtags to feel like twitter
- smileys to look like skype
- Markdown to be as cool as Buzz ;-)

You can imagine more sophisticated processing. And not only text-related because a processor actually has full access to the Activity. So you can very well process based on the owner, the app, mediaitem, etc..

3.4.6.4. Writing an ActivityProcessor

eXo Social let's you pre-process an activity before it is returned by the ActivityManager. To do this you simply need to implement the interface ActivityProcessor :

```
/**
 * An activity processor is responsible to pre-process an activity before it is returned by the {@link ActivityM
 */
public interface ActivityProcessor {

  /**
   * Process an activity
   * @param activity the activity. It can be modified
   */
  void processActivity(Activity activity);

  /**
   * Priority of this processor.
   * All activity processors will be executed in ascending priority order
   * @return
   */
  int getPriority();

}
```

For example let's implement a SmileyProcessor that will replace text smileys by icons :

```
public class SmileyProcessor implements ActivityProcessor {

  String smiley =  "<img src='/images/smiley.gif'/>";

  public void processActivity(Activity activity) {
    String title = activity.getTitle();
     activity.setTitle(title.replaceAll(":-)", smiley));
  }

  public int getPriority() {
    return 100;
  }

}
```

3.4.6.5. Configuring the processor

Now that you have a nice procesor you need to hook it to the system. At runtime, processors can be attached to the ActivityManager via the method ActivityManageraddProcessor(ActivityProcessor processor).

But there is also a component plugin hook for it : public void addProcessorPlugin(BaseActivityProcessorPlugin plugin)

So to make your processor easy to hook, you simply need to let him extend the BaseActivityProcessorPlugin.

```
public class SmileyProcessor extends BaseActivityProcessorPlugin {

  String smiley =  "<img src='/images/smiley.gif'/>";

  public SmileyProcessor(InitParams params) {
    super(params);
  }

  public void processActivity(Activity activity) {
    String title = activity.getTitle();
      activity.setTitle(title.replaceAll(":-)", smiley));
  }
}
```

It will have the additional benefit to make the priority field configurable so you don't need to implement getPriorty().

Then your processor can be configured as a component plugi like this :

```
  <external-component-plugins>
    <target-component>org.exoplatform.social.core.activitystream.ActivityManager</target-component>
    <component-plugin>
      <name>SmileyProcessor</name>
      <set-method>addProcessorPlugin</set-method>
      <type>org.example.SmileyProcessor</type>
      <init-params>
        <value-param>
          <name>priority</name>
          <description>priority of this processor (lower are executed first)</description>
          <value>2</value>
        </value-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
```

Restart, place the smiley gif on the server and you should see something like that :

3.5. Public REST APIs

3.5.1. Activities REST service

Name	Service URL	Location	Description
ActivitiesRestService	{restContextName}/{portalName}/social/activities	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.component.service	Provides REST services for activity applications: like/unlike; comment; delete activity.

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
destroyActivity	{restContextName}/{	portalName	String	Destroys activity and gets the json/xml format.
		activityId	String	
		format	String: json or xml	
showLikes	{restContextName}/{	portalName	String	Shows {format} of likes by activityId and returns the json/xml format.
		activityId	String	
		format	String: json or xml	
updateLike	{restContextName}/{	portalName	String	Updates {format} of likes by the json/xml format.
		activityId	String	
		format	String: json or xml	
destroyLike	{restContextName}/{	portalName	String	Destroys {format} identityId and get the json/xml return format.
		activityId	String	
		identityId	String	
		format	String: json or xml	
showComments	{restContextName}/{	portalName	String	Shows {format} comment list by the json/xml format.
		activityId	String	
		format	String: json or xml	
updateComment	{restContextName}/{	portalName	String	Updates {format} comment by the json/xml format.
		activityId	String	
		format	String: json or xml	
destroyComment	{restContextName}/{	portalName	String	Destroys {format} and returns the

Name	Service URL Endpoint	Parameters	Expected Values	Description
		activityId commentId format	String String String: json or xml	json/xml format.

Example:

http://localhost:8080/rest-socialdemo/socialdemo/social/activities/s08d397dg6/likes/destroy/abc.json

3.5.2. Apps REST service

Name	Service URL	Location	Description
AppsRestService	{restContextName}/social/apps/	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.component.service	Provides REST services for the application registry gadget: shows application list

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
showApps	{restContextName}/social/apps/show.{format}	format	String: json or xml	Shows applications by the json/xml format.

Example:

http://localhost:8080/rest-socialdemo/social/apps/show.json

3.5.3. Identity REST service

Name	Serive URL	Location	Description
IdentityRestService	restContextName}/{portalName}/social/identity/{username}	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.component.service	Gets identityId by the username.

Name	Serive URL	Location	Description

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
UserId getId	{restContextName}/{portalName}/social/identity/{username}/id/show.json	username portalName	String String	Gets the identity by username and returns by the json format.

Example:

http://localhost:8080/rest-socialdemo/socialdemo/social/identity/john/id/show.json

3.5.4. Linkshare REST service

Name	Service URL	Location	Description
LinkshareRestService	{restContextName}/social/linkshare	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.component.service	Gets information from a provided link.

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
getLink	{restContextName}/social/linkshare/show.{format}	format	String: json or xml	Gets the link content by posting a linkShare request.

Example:

http://localhost:8080/rest-socialdemo/social/linkshare/show.json

3.5.5. People Rest Service

Name	Service URL	Location	Description
PeopleRestService	{restContextName}/social/people	**Maven groupId**: org.exoplatform.social	Provides REST services for manipulating jobs

Name	Service URL	Location	Description
		ArtifactId: exo.social.component.service	related to people.

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
suggestUsernames	{restContextName}/social/people/suggest.{format}	nameToSearch currentUser typeOfRelation spaceURL format	String String String String String: json or xml	Gets and returns a list of users's name that matches the input string for suggesting.

Example: http://localhost:8080/rest-socialdemo/social/people/suggest.json

3.5.6. Spaces REST service

Name	Service URL	Location	Description
SpacesRestService	{restContextName}/{portalName}/social/spaces	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.component.service	Provides REST services for space gadget to display users' spaces and pending spaces.

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description	
showMySpaceList	{restContextName}/social/spaces/mySpaces/show.{format}	portalName format	String String: json or xml	Shows mySpaceList by the json/xml format.	
showPendingSpaceList	{restContextName}/social/spaces/pendingSpaces/show.{format}	portalName	String	Shows pendingSpaceList	

Name	Service URL Endpoint	Parameters	Expected Values	Description	
		format	String: json or xml	by the json/xml format.	
suggestSpacenames	{restContextName}/social/spaces/spaceNames/suggest.{format}	portalName conditionToSearch typeOfRelation currentUser format	String String String String String: json or xml	{format} returns space's names that match the input string for suggesting.	

Example:

http://localhost:8080/rest-socialdemo/social/spaces/mySpaces/show.xml

3.5.7. Widget Rest Service

Name	Service URL	Location	Description
WidgetRestService	{restContextName}/spaces/{containerName}	**Maven groupId**: org.exoplatform.social **ArtifactId**: exo.social.extras.widget.rest	Provides REST services for creating spaces or getting spaces'information.

- **API**:

Name	Service URL Endpoint	Parameters	Expected Values	Description
spaceInfo	{restContextName}/spaces/{containerName}/space_info	containerName portalName spacePrettyName description	String String (default value: classic) String String	Returns the HTML page for displaying the information of the space. Two query parameters needed: `spaceName` and `description`

Name	Service URL Endpoint	Parameters	Expected Values	Description

Example:

http://localhost:8080/rest-socialdemo/spaces/socialdemo/space_info?name=Social&description=Social

3.5.8. Location

- **Maven groupId**: org.exoplatform.social
- **ArtifactId**: exo.social.component.service

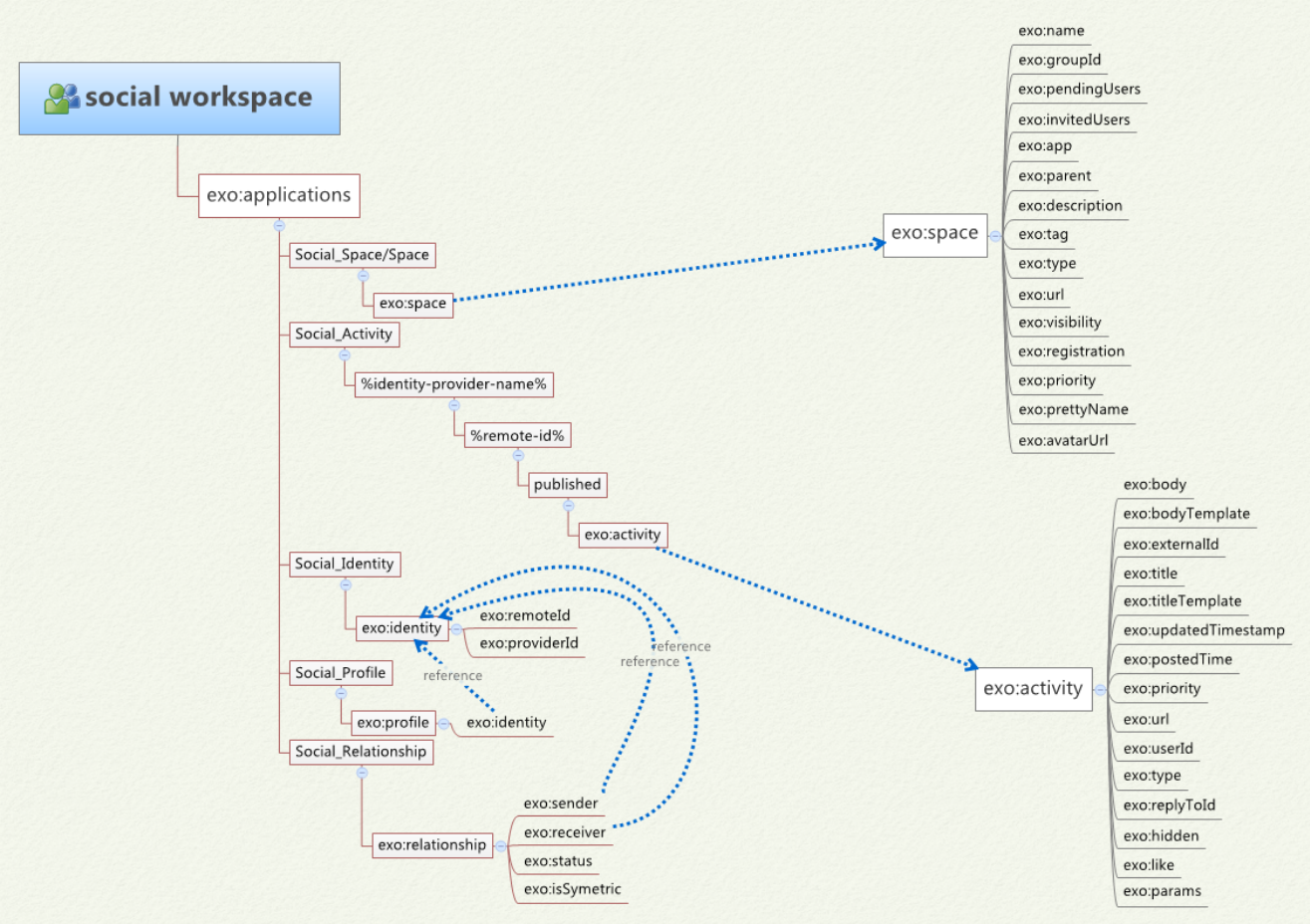
3.6. Public Javascript APIs

All references for Opensocial Javascript APIs can be viewed here

3.7. Social JCR Structure

3.7.1. Overview

As any other eXo's products, eXo Social fully complies the JCR standard to store data (identity, profile, activity, space and relationship). The Social JCR structure is organized to conform the data storage for the individual purpose of eXo Social. With this structure, it is easy for us to manage and access the data properly.



The root node of Social workspace is *exo:applications* which includes child nodes, such as *Social_Space/Space*, ,*Social_Activity*, *Social_Identity*, *Social_Profile*, *Social_Relationship*.

3.7.2. Social_Space/Space

Social_Space/Space is used to store data of eXo Social spaces. It includes nodes of the *exo:space* type. This node type is defined as follows:

```
<nodeType hasOrderableChildNodes="false" isMixin="false" name="exo:space" primaryItemName="">
  <supertypes>
    <supertype>nt:base</supertype>
  </supertypes>
  <propertyDefinitions>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:name" onParentVersion
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:groupId" onParentVers
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="true" name="exo:pendingUsers" onParent
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="true" name="exo:invitedUsers" onParent
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:app" onParentVersion=
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:parent" onParentVersi
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:description" onParent
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:tag" onParentVersion=
      <valueConstraints/>
```

```
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:type" onParentVersion
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:url" onParentVersion=
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:visibility" onParentV
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:registration" onParen
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:priority" onParentVer
         <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:prettyName" onParentV
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:avatarUrl" onParentVe
         <valueConstraints/>
      </propertyDefinition>
    </propertyDefinitions>
    <childNodeDefinitions>
      <childNodeDefinition autoCreated="false" defaultPrimaryType="nt:file" mandatory="false" name="*" onParentV
        <requiredPrimaryTypes>
          <requiredPrimaryType>nt:file</requiredPrimaryType>
        </requiredPrimaryTypes>
      </childNodeDefinition>
    </childNodeDefinitions>
  </nodeType>
```

The *exo:space* node type has the following properties:

Properties	**requiredType**	**Description**
exo:name	String	The space name.
exo:groupId	String	The group associated with a corresponding space.
exo:pendingUsers	String	The string contains a list of usernames which are pending for validation to join a space.
exo:invitedUsers	String	The string contains a list of usernames which are invited to join a space.
exo:app	String	The string contains a list of applications with portlet id, application name, and its state (installed, active, deactive).
exo:parent	String	The parent group of a space.
exo:description	String	The description of a space.
exo:tag	String	The string contains a list of keywords to tag a space.
exo:type	String	The type of space is used to run in the classic or webOS mode.
exo:url	String	The link to access a space.

Properties	requiredType	Description
exo:visibility	String	The string contains three values: public, private, and hidden.
exo:registration	String	The string contains three values: open, validation, and close.
exo:priority	String	The string is used to sort spaces in a spaces list. It contains three values: 1, 2 and 3. The smaller value has the higher priority.
exo:prettyName	String	The pretty name of a space.
exo:avatarUrl	String	The link to the avatar of a space.

3.7.3. Social_Activity

Social_Activity contains all the child nodes with the path name of the identity provider name. The identity provider name contains all the child nodes that have the path name of the remote id. The remote id path has one node with the path name as "published" that includes all the nodes of the *exo:activity* type. This node type stores data about the activity associated with a remote id. It is defined as follows:

```
<nodeType hasOrderableChildNodes="false" isMixin="false" name="exo:activity" primaryItemName="">
    <supertypes>
      <supertype>nt:unstructured</supertype>
    </supertypes>
    <propertyDefinitions>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:body" onParentVersion
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:bodyTemplate" onParen
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:externalId" onParentV
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:title" onParentVersion
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:titleTemplate" onPare
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:updatedTimestamp" onPa
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:postedTime" onParentVe
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:priority" onParentVer
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:url" onParentVersion=
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:userId" onParentVersi
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:type" onParentVersion
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:replyToId" onParentVe
        <valueConstraints/>
      </propertyDefinition>
```

```
      <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:hidden" onParentVersi
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="true" name="exo:like" onParentVersion=
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="false" multiple="true" name="exo:params" onParentVersio
        <valueConstraints/>
      </propertyDefinition>
    </propertyDefinitions>
  </nodeType>
```

The *exo:activity* node type has the following properties:

Properties	**requiredType**	**Description**
exo:body	String	The string specifies the body template message ID in the gadget spec. The body is an optional extended version of an activity.
exo:bodyTemplate	String	The body template is used to replace the parameters in the body.
exo:externalId	String	An optional string ID is generated by the posting application.
exo:title	String	The string specifies the primary text of an activity.
exo:titleTemplate	String	The template for the activity title.
exo:updatedTime	Long	The string specifies the time at which this activity is updated in milliseconds since the epoch.
exo:postedTime	Long	The string specifies the time at which this activity took place in milliseconds since the epoch.
exo:priority	Long	A number between 0 and 1 represents the relative priority of this activity in relation to other activities from the same source.
exo:url	String	The string specifies the stream's URL.
exo:userId	String	The string ID of the user who this activity is for.
exo:type	String	The application id which creates an activity.
exo:replyToId	String	The string contains the identity id which the activity uses.
exo:hidden	Boolen	The boolen denotes the activity display. If "true", the activity is not displayed.
exo:like	String	The list of identity ids who like the

Properties	requiredType	Description
		activity.
exo:params	String	A map of custom key/value pairs is associated with this activity.

3.7.4. Social_Identity

Social_Identity includes nodes of the *exo:identity* type. This node type stores data about any identities with removeId and providerId. This node type is defined as follows:

```
<nodeType hasOrderableChildNodes="false" isMixin="false" name="exo:identity" primaryItemName="">
    <supertypes>
      <supertype>nt:base</supertype>
    </supertypes>
    <propertyDefinitions>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:remoteId" onParentVers
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:providerId" onParentVe
        <valueConstraints/>
      </propertyDefinition>
    </propertyDefinitions>
  </nodeType>
```

The *exo:identity* node type has the following properties:

Properties	requiredType	Description
exo:remoteId	String	The local id from a providerId.
exo:providerId	String	The providerId is considered as a namespace for remoteId.

3.7.5. Social_Profile

The *Social_Profile*includes nodes of the *exo:profile* type. This node type stores data about identities. Each identiy has its own profile that stores information of this identity. This node type is defined as follows:

```
  <nodeType hasOrderableChildNodes="false" isMixin="false" name="exo:profile" primaryItemName="">
    <supertypes>
      <supertype>nt:unstructured</supertype>
    </supertypes>
    <propertyDefinitions>
      <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:identity" onParentVers
        <valueConstraints/>
      </propertyDefinition>
    </propertyDefinitions>
    <childNodeDefinitions>
        <childNodeDefinition autoCreated="false" defaultPrimaryType="" mandatory="false" name="*" onParentVersio
          <requiredPrimaryTypes>
            <requiredPrimaryType>nt:unstructured</requiredPrimaryType>
          </requiredPrimaryTypes>
        </childNodeDefinition>
    </childNodeDefinitions>
  </nodeType>
```

The node type has the following properties:

Properties	requiredType	Description
exo:identity	REFERENCE	The reference to the exo:identity node.

3.7.6. Social_Relationship

The *Social_Relationship*includes nodes of the *exo:relationship* type. This node type stores information about the connection type between identities. It is defined as follows:

```
<nodeType hasOrderableChildNodes="false" isMixin="false" name="exo:relationship" primaryItemName="">
  <supertypes>
    <supertype>nt:base</supertype>
  </supertypes>
  <propertyDefinitions>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:sender" onParentVersio
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:receiver" onParentVers
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="exo:status" onParentVersio
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="exo:isSymetric" onParentV
      <valueConstraints/>
    </propertyDefinition>
  </propertyDefinitions>
</nodeType>
```

This node type has the following properties:

Properties	requiredType	Description
exo:sender	REFERENCE	The sender identity. Its value is exo:identiy
exo:receiver	REFERENCE	The receiver identity. Its value is exo:identity
exo:status	String	The string has three values: PENDING, CONFIRMED, and IGNORED.
exo:isSymetric	boolean	Denotes if the connection is one way or two way.