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. RelationshipManager

The Service is used to manipulate user relationships.

2.2. SpaceIdentityProvider

The provider is to give identity space instances.

2.3. SpaceApplicationHandler

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

2.4. ExoPeopleService

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

2.5. Space Service

2.5.1. Description

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

2.5.2. 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>
    <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>
```

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:true, MembersPortlet:true`	The list of configurations for portlets used as portlet applications.

2.5.3. External plug-in configuration

```
<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' name 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.

2.6. Activity Manager

2.6.1. Description

The service is to manipulate space and user activities.

2.6.2. Component plug-in configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.manager.ActivityManager</target-component>
  <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>
</external-component-plugins>
```

In which,

Name	Set-method	Type	Description
OSHtmlSanitizerProcessor	addProcessorPlugin	org.exoplatform.social.core.processor.OSHtmlSanitizerProcessor	The plug-in filters invalid 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, p, ol, ul,li, br, img	To process and render html tags in the activity content (body).

2.6.3. External plug-in configuration

```
<external-component-plugins>
    <target-component>org.exoplatform.social.core.manager.ActivityManager</target-component>
    <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>
</external-component-plugins>
```

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).

2.7. Identity Manager

2.7.1. Description

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

2.7.2. 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	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.identity.provider.SpaceIdent`	Identity Provider instances for managing identities.

2.8. OpenSocial Rest Context Configuration

2.8.1. 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`.

2.8.2. 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.auth.RestPortalContainerNameC	The plugin provided 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

2.9. Spaces Template configuration

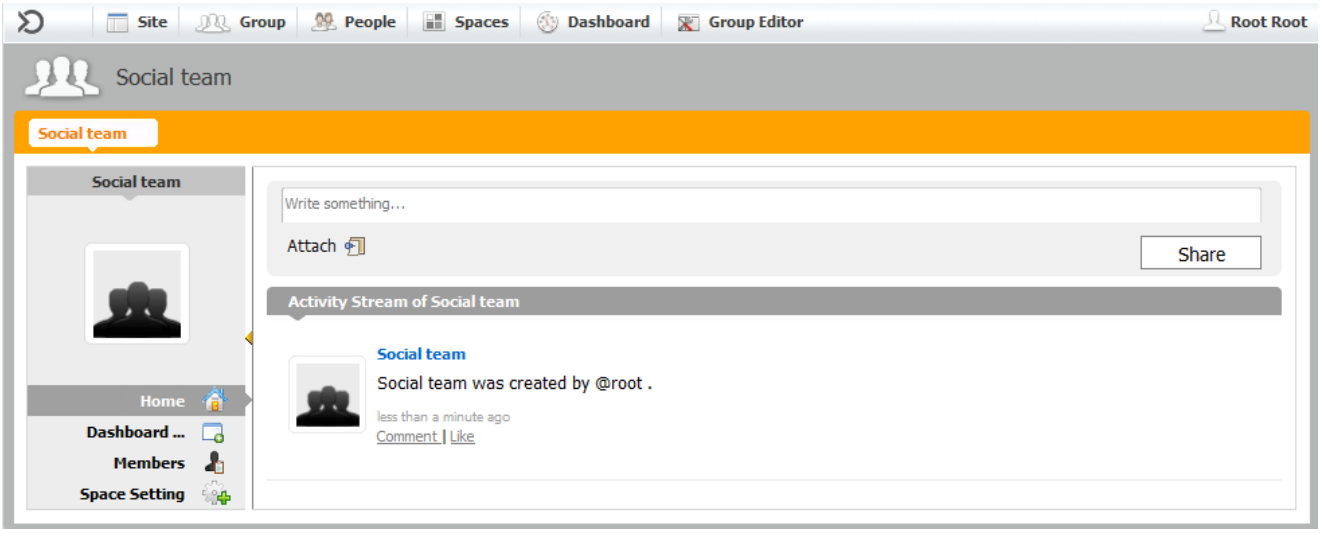
In the eXo Spaces, we may have two space types (classic and webos spaces). This is for the classic mode (it's the only one implemented now).

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.

2.10. 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

2.10.1. 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
```

2.10.2. 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 Social 1.1.0, the activity plugin feature was introduced to allow using activity composer extension and use 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 Social there's 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 release: 1.1.0-CR01.

```
<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-CR01</version>
  </parent>

  <groupId>org.exoplatform.social.samples</groupId>
  <artifactId>exo.social.samples.activity-plugin</artifactId>
  <packaging>jar</packaging>
  <version>1.1.0-CR01</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;

/**
 * Created by The eXo Platform SAS
 * Author : eXoPlatform
 *          exo@exoplatform.com
 * Aug 23, 2010
 */
@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 is also needed which will be explained later.

```
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;

/**
 * Created by The eXo Platform SAS
 * Author : eXoPlatform
 *          exo@exoplatform.com
 * Aug 19, 2010
 */
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;

/**
 * Created by The eXo Platform SAS
 * Author : eXoPlatform
 *          exo@exoplatform.com
 * Aug 23, 2010
 */
@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 activity builder?

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

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 ActivityStorageException	`activity`	void	Deletes a stored activity (id != null). (Since 1.1.1).
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).

Method	Param	Return	Description
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 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	`activity, identity` - a user who dislikes an activity	void	Removes an indentity who likes an activity, if this activity is liked, it

Method	Param	Return	Description
ActivityStorageException			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.
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,	`id` can be a social	null if nothing is found,	Gets the identity by

Method	Param	Return	Description
boolean loadProfile)	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	or the Identity object	loading id of the profile optionally.
deleteIdentity(Identity identity)	`identiy`	void	Deletes an identity.
addIdentityProvider(IdentityProvider idProvider)	`idProvider`	the id of provider	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 remoteId, boolean loadProfile)	`providerId` - referring to the name of the Identity provider, `remoteId` - the identifier that identify the identity in the specific identity provider, `loadProfile` - true when the profile is loaded	null if nothing is found, or the Identity object improves the performance by specifying what needs to be loaded	This function returns an Identity object that is 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(ProviderId,	`profileFilter`	List<Identity>	Gets the identities by a

Method	Param	Return	Description
profileFilter, long offset, long limit) throws Exception	`profileFilter, offset, limit`		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, String remoteId)	`providerId, remoteId`	boolean	Checks if the identity is 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 - the 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	`profile`	void	Updates the contact

Method	Param	Return	Description
p) throws Exception			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().
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	`relationship` - a relationship	void	Saves a relationship.

Method	Param	Return	Description
RelationshipStorageException			
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	`sender` - an identity	List<Relationship>	Gets all the pending relationship of sender.
getPendingRelationships(Identity sender, List<Identity> identities) throws Exception	`sender` - an identity, `identities` - a list of identity	List<Relationship>	Gets pending relationships of sender that match with identities.
getRequireValidationRelationships(Identity receiver) throws Exception	`receiver` - an identity	List<Relationship>	Gets list of required validation relationship of receiver.
getRequireValidationRelationships(Identity receiver, List<Identity> identities)	`receiver` - 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.

Method	Param	Return	Description
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 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` - 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.

Method	Param	Return	Description
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 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.

Method	Param	Return	Description
			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
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	`space, userId, isLeader`	void	Sets a member of a space as a manager.

Method	Param	Return	Description
SpaceException			
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.
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,	`space, userId`	boolen - true if that user	Checks if a user is in the

Method	Param	Return	Description
String userId) throws SpaceException		is in the pending list, otherwise, false	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.
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.

Method	Param	Return	Description
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 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`	void	Registers a space

Method	Param	Return	Description
listener			lifecycle listener.
unregisterSpaceLifeCycleListener(SpaceLifeCycleListener listener)	SpaceLifeCycleListener listener	void	Unregisters a space lifecycle listener.
setPortletsPrefsRequired(PortletPreferenceRequiredPlugin portletPrefsRequiredPlugin)	PortletPreferenceRequiredPlugin portletPrefsRequiredPlugin	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.

3.4. Java APIs sample code/ tutorial

3.4.1. Activity Stream

eXo Social provides users with a way to share status updates and users' activity information as well as spaces (also known as Activity Stream). With the API, you can customize the way to display activities or publish new ones. To manipulate activities, you will use the `AtivityManager`.To get an instance of this class, you will need to use the `PortalContainer`.

3.4.1.1. Introduction on some features of ActivityStream and ExoSocialActivity

```
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);

  // Root identity.
  private Identity rootIdentity;

  // 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 ROOT_NAME = "root";
  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 creates the identity "root".
    rootIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, ROOT_NAME);

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

  public void introduceActivityStreamAndExoSocialActivity {
    ExoSocialActivity activity = new ExoSocialActivityImpl();

    // 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 root.
    activity.setUserId(rootIdentity.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 ExoSocialActivity
    ExoSocialActivity rootActivity = new ExoSocialActivityImpl();;
    activity.setTitle(DEFAULT_ACTIVITY_TITLE);
    activityManager.saveActivity(rootIdentity, rootActivity);

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

    //Sets comments of root
    comment.setUserId(rootIdentity.getId());

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

3.4.1.2. 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.

3.4.1.2.1. 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);
    }
  }
}
```

3.4.1.3. 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);
    }
  }
}
```

3.4.1.3.1. Configure an activity processor

An activity processor is used to modify the content of activities before they are responded and rendered at a client's browser. For example, we will create an activity processor to replace all the texts representing the smile face ":-)" in the activity title by the smiley icons.

Firstly, we will 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, we have to 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.

3.4.1.3.2. 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 `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 Social project on Jira and Hudson will be automatically published to the `exosocial` space.

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}/social/activities/destroy/{activityId}.{format}	portalName activityId format	String String String: json or xml	Destroys activity and gets the json/xml format.
showLikes	{restContextName}/{portalName}/social/activities/{activityId}/likes/show.{format}	portalName activityId format	String String String: json or xml	Shows the list of likes by activityId and returns the json/xml format.
updateLike	{restContextName}/{portalName}/social/activities/{activityId}/likes/update.{format}	portalName activityId format	String String String: json or xml	Updates the list of likes by the json/xml format.
destroyLike	{restContextName}/{portalName}/social/activities/{activityId}/likes/destroy/{identityId}.{format}	portalName activityId identityId format	String String String String: json or xml	Destroys **like** by identityId and get the json/xml return format.
showComments	{restContextName}/{portalName}/social/activities/{activityId}/likes/show.{format}	portalName activityId format	String String String: json or xml	Shows the comment list by the json/xml format.
updateComment	{restContextName}/{portalName}/social/activities/{activityId}/likes/update.{format}	portalName activityId format	String String String: json or xml	Updates the comment by the json/xml format.

Name	Service URL Endpoint	Parameters	Expected Values	Description
destroyComment	{restContextName}/{portalName}/social/activities/{activityId}/comments/destroy/{commentId}.{format}	portalName activityId commentId format	String String String String: json or xml	Destroys comment and returns the 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}		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}/id	**Maven groupId**: org.exoplatform.social	Gets identityId by the username.

Name	Serive URL	Location	Description
		ArtifactId: exo.social.component.service	

- **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 **ArtifactId**: exo.social.component.service	Provides REST services for manipulating jobs 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	String	Shows mySpaceList by the json/xml	

Name	Service URL Endpoint	Parameters	Expected Values	Description	
		format	String: json or xml	format.	
showPendingSpaceList	{restContextName}/social/spaces/pendingSpaces/show.{format}	portalName format	String String: json or xml	Shows pendingSpaceList 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	String String (default	Returns the HTML page for displaying the information of the space. Two

Name	Service URL Endpoint	Parameters	Expected Values	Description
		spacePrettyName description	value: classic) String String	query parameters needed: `spaceName` and `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