

# JIDE Feed Reader Developer Guide

---

## Contents

<b>PURPOSE OF THIS DOCUMENT</b> .....	<b>1</b>
<b>FEATURES</b> .....	<b>2</b>
<b>HOW TO USE FEED READER</b> .....	<b>2</b>
<b>THE USER INTERFACE</b> .....	<b>3</b>
LAYOUT.....	3
TOOLBAR .....	3
HTML BROWSER.....	3
ICONS .....	4
<b>PERSISTENCE LAYER</b> .....	<b>5</b>
<b>USER PREFERENCE</b> .....	<b>5</b>
<b>FEED EVENTS</b> .....	<b>5</b>
<b>INTERNATIONALIZATION SUPPORT</b> .....	<b>6</b>
<b>DEPENDENCY</b> .....	<b>6</b>

## Purpose of This Document

Even if you never heard RSS before, you must hear “weblogs” or “blogs” which is very popular these days. RSS is the technology behind the “weblogs” that is used to publish frequently updated content. RSS is an acronym for Really Simple Syndication. In fact, RSS is used beyond “weblogs”. RSS basically makes it possible for people to keep up with their favorite web sites in an automated manner that’s easier than checking them manually.

RSS content can be read using software called a "feed reader" or an "aggregator." The user subscribes to a feed by entering the feed's link into the reader or by clicking an RSS icon in a browser that initiates the subscription process. The reader checks the user's subscribed feeds regularly for new content, downloading any updates that it finds.

There are many feed readers, commercial and open source. Most web browsers (IE, Firefox etc) and email clients (Outlook, Thunderbird etc) support RSS now. There are also many standalone feed readers.

Among all those feed readers, *JIDE RSS Feed Reader* serves a whole different purpose. Imagine you are a software provider providing a tool to millions of users. Every time you do a new release, you want to somehow notify your users. You can post news on your website but not many people check your website regularly. You can email a newsletter which means you need to have their email addresses. Even if you do, how often your newsletters will be filtered

by your users' junk email filter? Or you can have your own network protocol and let your tool talk to your server when it starts. If there is anything new, notify your user through a flashing button in the user interface. This works better than newsletter or web page, but do you realize RSS can be used for this exact purpose and you don't need to waste your precious time writing a new network protocol for it? That's what *JIDE RSS Feed Reader* for – to enable the usage of RSS inside your application. *JIDE RSS Feed Reader* is a Swing component which means you can embed this component inside your application so that your users will always be kept in sync with the content you want to them to know, from product releases, critical upgrade notification, company news, even product tips and tricks.

This developer guide is designed for developers who want to learn how to use *JIDE RSS Feed Reader* in their applications.

## Features

Here are the main features of *JIDE RSS Feed Reader*.

- ❖ Supports several feed formats such as RSS 0.9x, RSS 1.0 / RDF, RSS 2.0, Atom 0.3 and Atom 1.0. This is made possible by using an open source library called Informa (<http://informa.sourceforge.net/>).
- ❖ Supports subscription to any RSS sources (a.k.a. Channel). You can also customize it so that only a fixed number of RSS sources to be used.
- ❖ Supports operations on read/unread status of each feed item or all items in the same channel
- ❖ Supports automatic channel update
- ❖ Supports optional grouping of channels
- ❖ Built-in persistence of the channel list, read/unread status of feed items and user preference
- ❖ Supports customization to use any other html browsers as the embedded browser. Also allow user to launch native web browser to view the channel and feed items.

## How to use Feed Reader

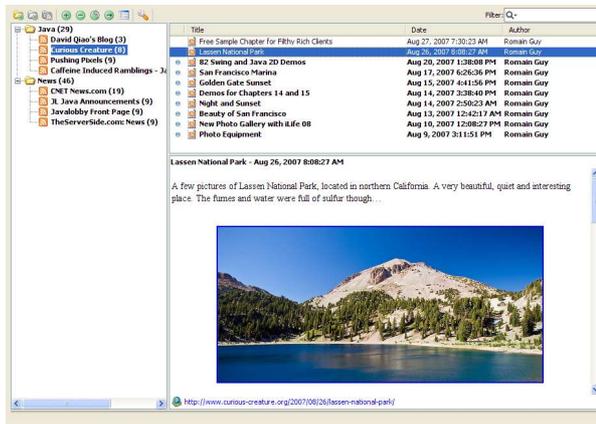
It is very easy to use Feed Reader. All you need to do is to create an instance like this and use it inside your application.

```
FeedReader feedReader = new FeedReader();
```

The constructor also can take a local folder or an instance of *FeedPersistenceLayer* as parameter. This will give you the option where to store the user data. By default, we will use *hsqldb* as a file based database to store the data. However you can always use your own way to persist the user data as long as you implement *FeedPersistenceLayer* interface and call *FeedReader#setFeedPersistenceLayer()* to use it.

*FeedReader* constructor also can take `String[]` to allow you to pass in an array of default channels. Those channels will be subscribed by default when user uses it for the first time.

## The User Interface



## Layout

There are four parts in *JIDE RSS Feed Reader*. The top portion is the tool bar component. It has “add category”, “remove category”, “category property”, “add channel”, “remove channel”, “reload channel”, “move channel”, “channel property” and “preference” buttons. Please note, this is the case when grouping enabled. If grouping is disabled, all group related buttons will not be visible. On the left portion, it’s the channel tree. It has all the subscribed channels. There are two portions on the right side. On the top-right portion, there is feed item table. On the bottom-right portion, there is an html browser.

This is just the default layout. You can override this method to decide the layout you want.

```
JComponent layoutComponents(Component toolbar, Component channelPane, Component itemPane, Component browserPane)
```

## ToolBar

The toolbar component is customizable too. You can override this method to do.

```
protected JComponent createToolBarComponent()
```

If you call `super.createToolBarComponent`, you will get a `JToolBar`. You can add a new button, remove an existing button etc. You can’t access directly to any of the buttons but you can look up by the name because each button has a unique name which is defined in the `feed.properties` file.

## HTML Browser

Another important portion that should be allowed to customize is the html browser. By default, we use `JEditorPane` to preview the RSS content which is html format. If you ever used

*JEditorPane* before, you know *JEditorPane* can only display very simple html content. For complex html content, we suggest you to use a commercial html browser component. But since our focus is not to make such a component, we allow you to hook up with whatever html browser component you selected. Those are the three methods you need to override in order to provide your own html browser.

```
protected Component createHtmlBrowser()
protected void clearHtmlBrowser()
protected void displayHtmlBrowser(String content)
```

## Icons

All the icons *JIDE Feed Reader* used can be customized. You can override this method to do it.

```
protected ImageIcon getImageIcon(String name)
```

*FeedIconsFactory* is the *IconsFactory* that provides all the default icons. Here is the list of the icons. The last column in the table below is the name that will be passed to *getImageIcon* method above.

### Icons in com.jidesoft.rss.FeedIconsFactory

Generated by JIDE Icons

1. If you cannot view the images in this page, make sure the file is at the same directory as *FeedIconsFactory.java*
2. To get a particular icon in your code, call *FeedIconsFactory.getImageIcon(FULL\_CONSTANT\_NAME)*. Replace *!* with the actual full constant name as in the table below

#### FeedIconsFactory

Name	Image	File Name	Full Constant Name
ADD_GROUP		icons/addGroup.png	FeedIconsFactory.ADD_GROUP
ADD_GROUP_DISABLED		icons/addGroup_disabled.png	FeedIconsFactory.ADD_GROUP_DISABLED
REMOVE_GROUP		icons/removeGroup.png	FeedIconsFactory.REMOVE_GROUP
REMOVE_GROUP_DISABLED		icons/removeGroup_disabled.png	FeedIconsFactory.REMOVE_GROUP_DISABLED
MOVE_CHANNEL		icons/moveChannel.png	FeedIconsFactory.MOVE_CHANNEL
MOVE_CHANNEL_DISABLED		icons/moveChannel_disabled.png	FeedIconsFactory.MOVE_CHANNEL_DISABLED
GROUP_PROPERTY		icons/group.png	FeedIconsFactory.GROUP_PROPERTY
GROUP_PROPERTY_DISABLED		icons/group_disabled.png	FeedIconsFactory.GROUP_PROPERTY_DISABLED
ADD		icons/add.png	FeedIconsFactory.ADD
ADD_DISABLED		icons/add_disabled.png	FeedIconsFactory.ADD_DISABLED
REMOVE		icons/remove.png	FeedIconsFactory.REMOVE
REMOVE_DISABLED		icons/remove_disabled.png	FeedIconsFactory.REMOVE_DISABLED
RESET		icons/reset.png	FeedIconsFactory.RESET
RESET_DISABLED		icons/reset_disabled.png	FeedIconsFactory.RESET_DISABLED
PROPERTY		icons/property.png	FeedIconsFactory.PROPERTY
PROPERTY_DISABLED		icons/property_disabled.png	FeedIconsFactory.PROPERTY_DISABLED
PREFERENCE		icons/preference.png	FeedIconsFactory.PREFERENCE
UNREAD		icons/unreadFlag.png	FeedIconsFactory.UNREAD
RSS		icons/rss.png	FeedIconsFactory.RSS
RSS_ITEM		icons/rssItem.png	FeedIconsFactory.RSS_ITEM
BROWSER		icons/browser.png	FeedIconsFactory.BROWSER
CONTENT		icons/contents.png	FeedIconsFactory.CONTENT
CLEAR		icons/clear.png	FeedIconsFactory.CLEAR
MAKE_READ		icons/read.png	FeedIconsFactory.MAKE_READ
MAKE_UNREAD		icons/unread.png	FeedIconsFactory.MAKE_UNREAD
TOGGLE		icons/toggle.png	FeedIconsFactory.TOGGLE

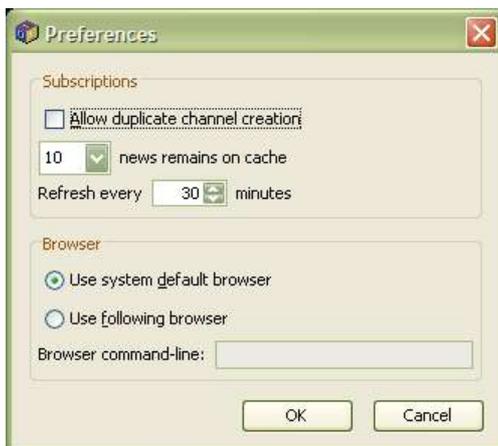
## Persistence Layer

As mentioned earlier, the persistence layer used by *JIDE Feed Reader* can be customized. The interface is called *PersistenceManagerIF* which is an interface defined by Informa. It has methods related to the categories, subscribed channels, read status of each feed item. As long as you implement those methods on the interface, *JIDE Feed Reader* will use it to save the information correctly.

By default, we have *FeedDatabasePersistenceManager* which uses *hsqldb* as the mechanism to persist the information. The only thing *FeedDatabasePersistenceManager* needs is a local folder. Two subfolders will be created. The “icon” subfolder is for channel icons. The “db” subfolder for *hsqldb* files. We recommend you use a hidden folder under *user.home* as the local folder so that the information is saved per user. Please note, each instance of *FeedReader* must have its own folder. It cannot be shared.

The user preference which can be customized using Preference button is stored in an object that implements *FeedPreference* interface. You can call *FeedReader#getFeedPreference* to get it. We will save it using *java.util.pref*. But you can always save it as part of your application preference by providing your own *FeedPreference*.

## User Preference



There are only a few options that are available in this component as you can see from the preference dialog above. All the preferences are saved as part of *FeedPersistenceLayer*. You can also change it yourself using *getFeedPersistenceLayer().getPreference()* to get an instance of *FeedPreference* and change the value on it.

## Feed Events

*FeedReader* will fire *FeedEvent* when something happened. You can listen to the event by adding a *FeedEventListener* using *addFeedEventListener*. You will receive an notification when

- ❖ A category is added

- ❖ An existing category is removed
- ❖ An existing category name is changed
- ❖ A channel is added
- ❖ An existing channel is removed
- ❖ A channel's title is changed
- ❖ Feed items from channel are cleared
- ❖ A channel is reloaded
- ❖ A feed item read status is changed
- ❖ User preference is changed
- ❖ There is a info or error message

You can choose to handle those events by looking at the event id. The last message event is mainly for the status bar if you have one. If you have an application level status bar, you can listen to the message event and display the message on the status bar.

## Internationalization Support

All Strings used in *JIDE Feed Reader* are contained in one properties file called `feed.properties` under `com/jidesoft/rss`. Some users contributed localized version of this file and we put those files inside `jide-properties.jar`. If you want to support languages other than those we provided, just extract this properties file, translated to the language you want, add the correct postfix and then jar it back into `jide-properties.jar`. You are welcome to send the translated properties file back to us if you want to share it.

## Dependency

JIDE Feed Reader product depends on JIDE Common Layer, JIDE Grids and JIDE Components. In addition, it depends on the following 3rd party jars.

1. `informa.jar` (<http://informa.sourceforge.net/>, LGPL). Informa project is a news aggregation library based on the Java Platform. Informa itself depends on
  - a) `commons-logging.jar`: <http://jakarta.apache.org/commons/logging/>, apache license
  - b) `jdom.jar`: <http://www.idom.org/>, apache license

2. `hsqldb.jar` (<http://hsqldb.org/>, BSD license) which is "Lightweight 100% Java SQL Database Engine". This dependency is optional. Only if you are using the default `FeedDatabasePersistenceManager` we provided, you will have this dependency. Many applications have their own way to persist the information. So if you implement your own implementation of `PersistenceManagerInterface`, you don't have to use `hsqldb`.

We used both libraries without any modification to them. So you can download from their website directly to get those jars.