
Table of Contents

Foreword	ii
1. About, Prereqs, and Dependencies	3
1.1. About	3
View mode application flow	3
Caching XML Data	5
Preferences Mode Application Flow	5
1.2. Prerequisites	7
1.3. Dependencies	7
Compile/Test Time	7
RunTime	7
JavaScript	8
2. Configuration, Compilation, and Deployment	9
2.1. Configuration	9
2.2. Compilation	9
2.3. Deploying the portlet to Tomcat	10
Installing the Portlet (uPortal only)	10
3. Data Formats	11
3.1. Feed Lists	11
4. Customizing the look of the Portlet	15
4.1. CSS	15
4.2. JSP	15

Foreword



The Internet Frameworks Services division of Duke University [<http://www.duke.edu/>]'s Office Of Information Technology (OIT) [<http://oit.duke.edu/>] has created a number of portlets used in the student portal, DukePass. We have released the source code for these portlets under the JA-SIG License [<http://www.uportal.org/license.html>].

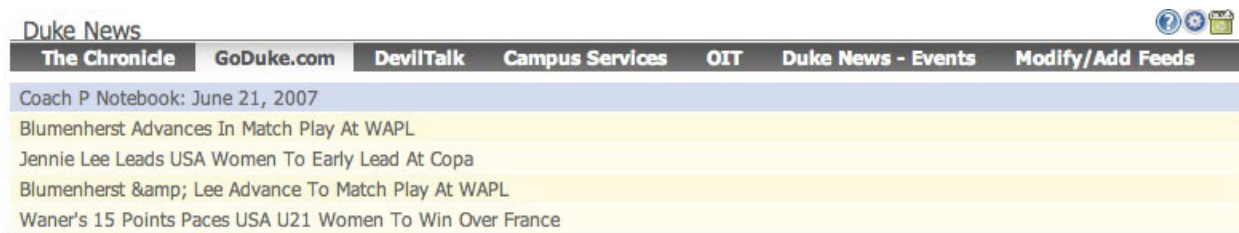
We encourage anyone with an interest to download, install, and use these portlets. Our portal installation uses uPortal version 2.5.3, but we have tried wherever possible to create JSR-168 compliant portlets that should run in any portlet container.

1. About, Prereqs, and Dependencies

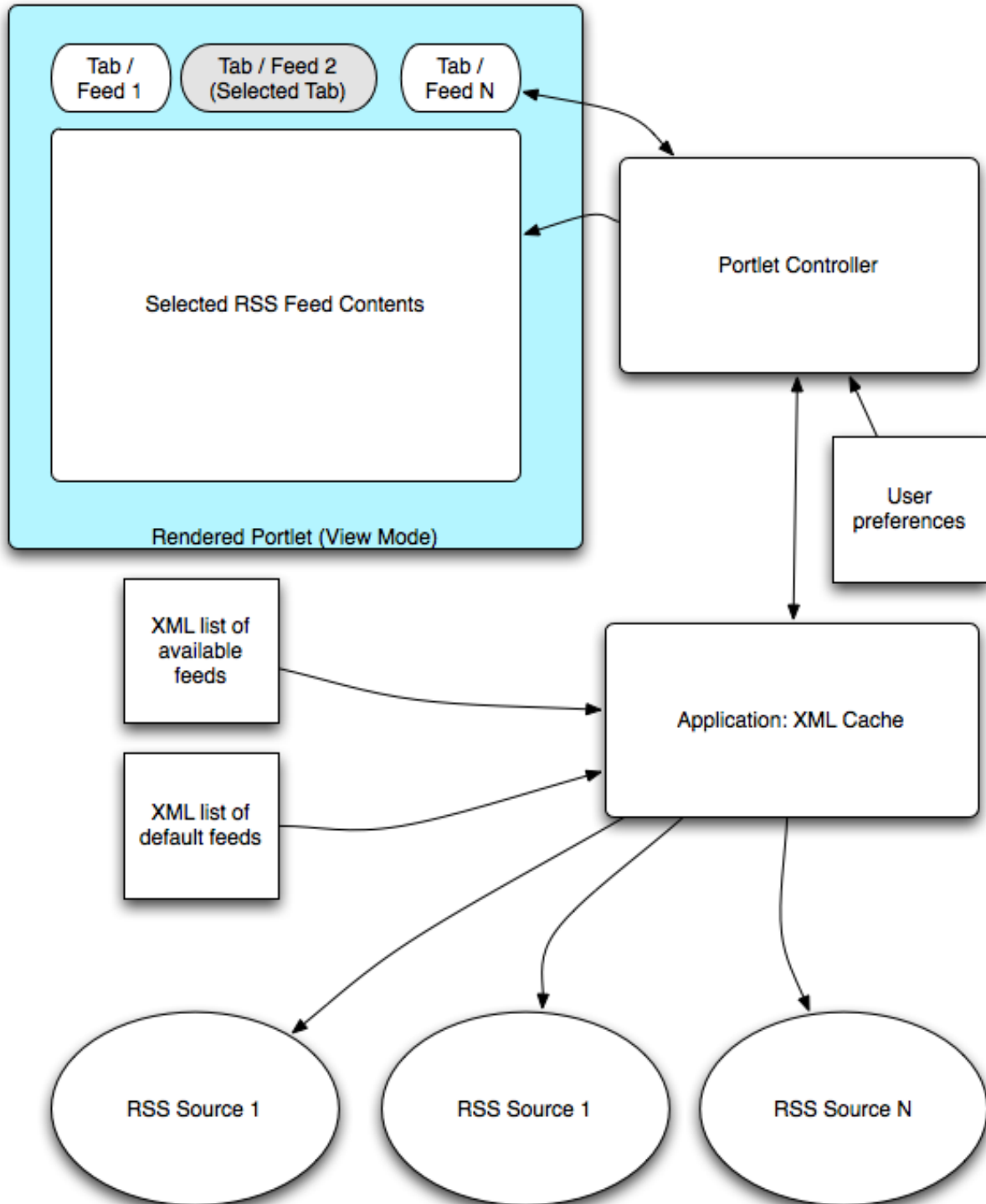
1.1. About

The Tabbed RSS Portlet aggregates multiple RSS feeds into a single portlet, displaying one feed per tab. It allows for portlets to offer a canned list of RSS feeds for a user to select from, with some available by default. It also allows the user to enter arbitrary RSS feeds.

On its own, this portlet is inefficient - it relies on the presence of a cached source of the data for the RSS feeds, otherwise this portlet would call the RSS source on every rendering of an individual portlet. At Duke, we accomplish this with a very lightweight servlet called 'getrss,' which acts as an intermediary cache allowing feed results to be shared across any portal request for the data.



View mode application flow



Upon first rendering of the Tabbed RSS portlet, the portlet controller will fetch the list of available RSS feeds. Each

of the feeds in this list labeled as a default feed will be added to the model as a tab. The first tab's RSS content will be fetched from the internet. A model containing the tabs and the contents of the selected RSS feed are sent to the view. The view will render each feed as a tab, and the contents of the selected tab will be shown in the main body of the portlet.

When a different tab is selected, the portlet fetches the RSS content for the selected feed, and returns the model (a list of tabs and contents for the selected feed) with the new feed's content replacing the previous feed's content.

Over time, a user may customize the list of feeds that is selected for display in the tabs. These may be selected from a canned list of available feeds, or a user may enter arbitrary RSS URLs for consumption.

In general, the rendering logic is this:

1. If the user has not customized their list of feeds, display the default list of feeds made available to the portlet in the portlet's feed XML file. These will be the tabs displayed.
2. If the user has customized their list of feeds, look up the ID numbers for any selected feeds in the portlet's feed XML file, and add any arbitrary RSS urls that they have added. These will be the tabs displayed.
3. Grab the RSS contents for the currently selected feed and return those results in the model.

Caching XML Data

There are several varieties of XML data that the Tabbed RSS Portlet consumes. The first is the list of RSS feeds that will be offered to the user, and displayed by default. Each RSS source produces XML as the contents of the feed, and for performance improvements and compliance with the terms of service, it's better if the RSS source isn't contacted each time the RSS feed needs to be rendered.

At Duke, we use an intermediary XML cache that runs as a separate servlet to maintain frequently used XML data. In order to enable the use of this cache, we prefix all RSS URL requests with a URL that will cause the cache to intercept the request and serve the result from the cache if available, and fetch the content and place it in the cache if not.

As an example, an RSS url is: <http://www.tuaw.com/rss.xml>

A proxy for us is located at: <http://example.duke.edu/getxml>

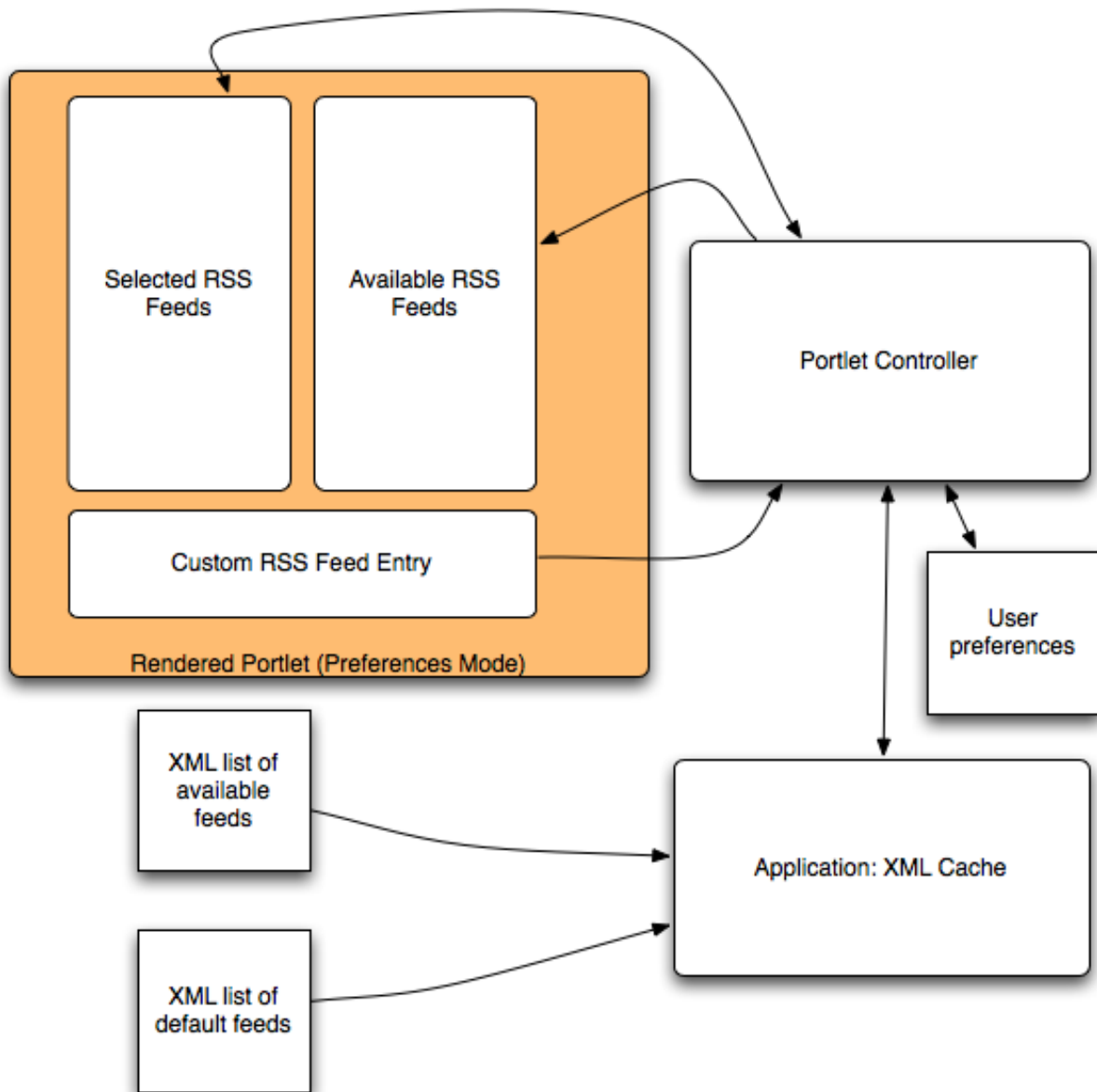
The proxy is expecting a single argument, named uri, so we send requests of the format

<http://example.duke.edu/getxml?uri=http://www.tuaw.com/rss.xml>

The RSS URL acts as the key into the cache. So if any portal user has requested the TUAW rss feed within the allowed cache time, the results will be very quickly returned from a local cache. If the feed isn't available from the cache, tuaw.com will be contacted, the RSS feed retrieved, stored in the cache, and returned to the portal.

Similarly, our list of feeds is maintained in an xml file, feeds.xml. This could be obtained by going to <http://example.duke.edu/portal/support/rss/feeds.xml>, or we could grab a version from the cache. In this case, Apache will probably serve the small static file locally much faster than a cache, so we opt to avoid the cache and get the file directly from the server.

Preferences Mode Application Flow



When the user enters preferences mode, they are presented with three available activities.

1. Their list of selected feeds is displayed (populated by the default feeds if they haven't yet set preferences), and they can remove feeds from this list
2. A tree of all available feeds is displayed, and users can add feeds to their selected feed list in (1) by clicking on one of the feed names
3. Text fields are present for entry of the title and URL of an arbitrary RSS feed. They may enter any valid RSS URL and title in these boxes, and they will be added to the list of selected feeds in (1)

Once the user is done, they can return to the reader either by clicking the "Return to Reader" option or using the portal's mechanism to return to 'view' mode

1.2. Prerequisites

Before you begin, you should know how to deploy portlets into your portal environment. We developed and tested this portlet with uPortal v2.5.3, but it may work with other version of uPortal. If you discover a shortcoming in our code that we can correct to make the portlet more JSR-168 compliant, please let us know.

You will need to have ant and a Java 1.5+ compiler installed.

1.3. Dependencies

Where permissible, all dependencies are included in the /dep directory. These are separated into those dependencies required for runtime, and those required only for building (such as unit testing). Dependant libraries are listed below.

Compile/Test Time

- JUnit 3.x
- Servlet API
- Portlet API
- Spring Mock Objects

RunTime

- Duke Core Services (included)
- Spring Framework 2.x
- Spring Portlet Support
- Spring AOP Support
- AspectJ (comes with Spring)
- Jdom

- Rome RSS Parser
- Commons Logging
- Log4j

JavaScript

This portlet uses portions of the Yahoo User Interface JavaScript library, and it is **not** included in this distribution, or included in the JSPs delivered by default with this package. We include common javascript libraries in the header of our portal itself, rather than each individual portlet, to avoid loading redundant copies of large JavaScript libraries.

- YUI Core
- YUI Tree
- YUI Event

2. Configuration, Compilation, and Deployment

2.1. Configuration

Unzip the distribution .zip file to a working directory that will be used to build and configure the portlet. Change to the working directory. All paths referenced below are relative to the working directory.

This portlet has a Spring-based configuration, so you should be able to change all relevant parameters in text files rather than changing the source code. However, you are welcome to change the source code to suit your needs if you are comfortable doing so.

It is important that you edit build.properties. This file contains properties that will be used to name the generated war file, and create the uPortal deployment descriptor for you.

The name of the generated war file is important -- the property "portlet-name" allows you to uniquely identify multiple instances of the portlet, if you so desire, with their own individual configurations.

You must also edit tabbedrssportlet.properties. Information in this properties file tells the portlet how to assemble a URL that will be used to retrieve and XML list of available RSS feeds.

URLs are assembled with the following pattern: feedList.url + feedPortlet.prefix + / + options + feedList.extension

```
# Example

feedList.url=http://example.duke.edu/portlet/rss/
feedList.extension=.html
feedPortlet.prefix=GeneralNews

# Will tell the portlet to look for a list of feeds at
# http://example.duke.edu/portlet/rss/GeneralNews/options.html
```

Again, the feedPortlet.prefix is there so that you can manage multiple instances of the tabbed RSS portlet, with potentially different lists of available feeds.

An example of the expected XML format for the list of RSS feeds appears in the "Feed Lists" section that follows.

2.2. Compilation

Once the configuration is complete, you're ready to compile and create a deployable version of the portlet. By default, the portlet will be compiled in a format readily deployable for uPortal, a pluto-based portlet container. Alternate deployments will require modification of the portlet.xml file, the web.xml file, and possibly the build.xml file.

```
ant war
```

This will compile the application, run the unit tests, and generate a .war file that is ready to go into your portal environment.

If the build succeeds, you will find the .war file in `/dist/**filename**.war` where `**filename**` is the portlet-name you specified in `build.properties`

2.3. Deploying the portlet to Tomcat

Deploying the portlet only requires copying the portlet .war file to your servlet's webapp directory.

```
cp dist/**filename**.war $TOMCAT_HOME/webapps
```

Installing the Portlet (uPortal only)

Every portlet container has a different method for installing a new portlet once it is deployed as a web application. Instructions are included here for installing the portlet for use with uPortal.

3. Data Formats

3.1. Feed Lists

Here is an example of the format used to tell the portlet what RSS feeds to make available to the user. Its name should be "options" plus the extension you specified in `tabbedrssportlet.properties`.

```
<feed-list name="">
  <feed-category description="Duke">
    <feed id="1178799878762">
      <title>Library Hacks Blog</title>
      <description>Library Hacks: Tips and tools to save you time</description>
      <url>http://library.duke.edu/blogs/libraryhacks/feed/</url>
    </feed>
  </feed-category>
  <feed-category description="Entertainment">
    <feed id="1175104948052">
      <title>People Magazine</title>
      <description>People Magazine</description>
      <url>http://rss.people.com/web/people/rss/topheadlines/index.xml</url>
    </feed>
    <feed id="1175104918095">
      <title>Entertainment Weekly</title>
      <description>Entertainment Weekly</description>
      <url>http://rss.ew.com/web/ew/rss/todayslatest/index.xml</url>
    </feed>
    <feed id="1175104886346">
      <title>Babbledocket - UNC Events</title>
      <description>Babbledocket - UNC Events</description>
      <url>http://babbledocket.com/rss/uncrss.xml</url>
    </feed>
    <feed id="1175104855783">
      <title>Babbledocket - Duke Events</title>
      <description>Babbledocket - Duke Events</description>
      <url>http://babbledocket.com/rss/dukerss.xml</url>
    </feed>
    <feed id="1175104818981">
      <title>Quotes of the Day</title>
      <description>Quotes of the Day</description>
      <url>http://feeds.feedburner.com/quotationpage/qotd</url>
    </feed>
    <feed id="1175104737854">
      <title>Wikipedia</title>
      <description>Wikipedia</description>
      <url>http://feeds.feedburner.com/WikinewsLatestNews</url>
    </feed>
    <feed id="1175104750721">
      <title>Urban Legends</title>
```

```
        <description>Urban Legends</description>
        <url>http://www.snopes.com/info/whatsnew.xml</url>
    </feed>
</feed-category>
<feed-category description="National News">
    <feed id="1178888615478">
        <title>The Onion Daily News</title>
        <description>Needs description</description>
        <url>http://feeds.theonion.com/theonion/daily</url>
    </feed>
    <feed id="1175607832017">
        <title>MSNBC</title>
        <description>Latest news from MSNBC</description>
        <url>http://rss.msnbc.msn.com/id/3032091/device/rss/rss.xml</url>
    </feed>
    <feed id="1175105317012">
        <title>Salon.com</title>
        <description>Salon.com</description>
        <url>http://feeds.salon.com/salon/index</url>
    </feed>
    <feed id="1175105228311">
        <title>Washington Post - National</title>
        <description>Washington Post - National</description>
        <url>http://www.washingtonpost.com/wp-dyn/rss/nation/index.xml</url>
    </feed>
    <feed id="1175105193092">
        <title>New York Times</title>
        <description>New York Times - National</description>
        <url>http://www.nytimes.com/services/xml/rss/nyt/National.xml</url>
    </feed>
    <feed id="1175105131650">
        <title>Herald-Sun Durham</title>
        <description>The Herald-Sun (Durham News)</description>
        <url>http://www.heraldsun.com/rss/durham/th.s.xml</url>
    </feed>
    <feed id="1174498111178">
        <title>CNN Top Stories</title>
        <description>Top Stories from CNN.com</description>
        <url>http://rss.cnn.com/rss/cnn_topstories.rss</url>
    </feed>
</feed-category>
<feed-category description="Sports">
    <feed id="1175105539942">
        <title>WRAL Sports</title>
        <description>WRAL Sports</description>
        <url>http://www.wral.com/rs/rss/29</url>
    </feed>
    <feed id="1174498307352">
        <title>ESPN Top Stories</title>
        <description>Top Stories from ESPN.COM</description>
        <url>http://sports.espn.go.com/espn/rss/news</url>
    </feed>
    <feed id="1174498205194">
```

```
        <title>CNN-SI</title>
        <description>Sports Illustrated News</description>
        <url>http://rss.cnn.com/rss/si_topstories.rss</url>
    </feed>
</feed-category>
<feed-category description="Technology">
    <feed id="1175613377060">
        <title>Gizmodo</title>
        <description>The latest technical news from Gizmodo</description>
        <url>http://gizmodo.com/index.xml</url>
    </feed>
    <feed id="1175613331351">
        <title>Engadget</title>
        <description>The latest tech news from engadget</description>
        <url>http://www.engadget.com/rss.xml</url>
    </feed>
    <feed id="1175613274322">
        <title>SlashDot</title>
        <description>Latest entries from SlashDot</description>
        <url>http://rss.slashdot.org/Slashdot/slashdot</url>
    </feed>
    <feed id="1175105824581">
        <title>Washington Post - Technology</title>
        <description>Washington Post - Technology</description>
        <url>http://www.washingtonpost.com/wp-dyn/rss/technology/index.xml</url>
    </feed>
    <feed id="1175105782048">
        <title>Wired</title>
        <description>Wired</description>
        <url>http://feeds.wired.com/wired/index</url>
    </feed>
</feed-category>
<feed-category description="World News">
    <feed id="1175105482602">
        <title>Time Magazine</title>
        <description>Time Magazine</description>
        <url>http://feeds.feedburner.com/time/topstories</url>
    </feed>
    <feed id="1175105447472">
        <title>Washington Post - World</title>
        <description>Washington Post - World</description>
        <url>http://www.washingtonpost.com/wp-dyn/rss/world/index.xml</url>
    </feed>
    <feed id="1175105410903">
        <title>New York Times - World</title>
        <description>New York Times - World</description>
        <url>http://www.nytimes.com/services/xml/rss/nyt/International.xml</url>
    </feed>
</feed-category>
</feed-list>
```

In the above, categories can have a depth of 1 only, but there can also be feeds at the root level, not a part of any category.

There is another XML file required called "defaults." Defaults has the same format of the above XML file, and will tell the portlet which feeds to show by default before a user has made any configuration changes. This will be refactored shortly so that "options" is the only required file, and 'default' will be an attribute available on each feed.

4. Customizing the look of the Portlet

4.1. CSS

This section details the CSS styles used to lay out the portlet. You'll need to change these to match the look and feel of your portal's themes and skins. As the standards evolve in CSS styling for portlets, we'll be converting these specific names to more general CSS classes as defined by the specs.

Here is the section of CSS we use to style the portlet at Duke. We chose to include all of our CSS in one .css file that's downloaded as a part of the portal page's template rather than distribute a .css file with each portlet.

```
/** Portlet Tabs **/  
div.portletTabsWrap {background:#305291 url(../slices/drk-blue-short.png) repeat-x left top; width:100%; height:100%;}  
ul.portletTabs {clear:both; margin:0; padding:0 0 0 8px;}  
ul.portletTabs li {float:left;}  
ul.portletTabs li a {float:left; display:block; padding:0 1em; margin:0 .1em 0 0; font-size:1em;}  
ul.portletTabs li a.current, ul.portletTabs li a:hover {background-color:#FFF; color:#000; text-decoration:none;}
```

Example 4.1 TabbedRSS CSS

4.2. JSP

We've used CSS strategies that will flow wherever possible, however, our JSP assumes a fairly wide layout. Therefore you may want to re-tool how the data are laid out on your page. We've used Spring WebMVC with standard JSTL tags. There are, however, text strings embedded in the JSP. We'll be working to extract these strings to make internationalization a more straightforward procedure.