

## Developing Filters

You use filters as to modify content globally across your Clearspace instance. A filter will modify all content within a body, subject, an so on. Users don't interact with filters installed on the system — filters do their work behind the scenes. For example, without any special markup or user action, a filter could do something such as replace one word with another — say, replace "cloudy" with "sunny":

It is a cloudy day.

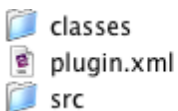
would become

It is a sunny day.

**Note:** Macros are related to filters in that they're intended to modify content. For more on building macros, see [Macro Development Guide](#).

## Directory Structure

When you create a filter, you create a plugin with the same directory structure and layout described in the plugin development guide. The following illustration shows the directory hierarchy that's best for developing your filter plugin.



Java source files will be stored in the src directory, with their compiled results in the classes directory. The plugin.xml file tells Clearspace what's included in the plugin, where to find related files, and so on. You can also include a lib directory for third-party JAR files your code needs.

## Implementing a Filter Class

To implement a filter extend [com.jivesoftware.community.renderer.BaseFilter](http://com.jivesoftware.community.renderer.BaseFilter)

## Annotations

You can use Java annotations to configure many aspects of a filter. You'll find more information about these in the [Clearspace Javadoc](#) .

**In [com.jivesoftware.community.annotations](http://com.jivesoftware.community.annotations)**

Annotation	Description	Default Value
Name	The name of the filter.	The short name of the class.
Description	The description of the filter.	No description.
PropertyNames	Parameters exposed (via getters and setters) for configuration. These parameters are set through the admin console and persisted by the filter framework.	Filters always automatically have an "enabled" property.

#### In `com.jivesoftware.community.render.annotations`

Annotation	Description	Default Value
RenderTypes	Used to specify which renderers the filter supports.	The filter will be applied to the body of the various content types.
EnableByDefault	Whether or not the filter is enabled by default.	True.

#### Annotation Example

```
@Name("base64")
@Description("Converts text to Base64.")
@RenderTypes({RenderType.BODY, RenderType.SUBJECT})
public class Base64Filter extends BaseFilter {
```

## Configuration With a plugin.xml File

Every plugin has a plugin.xml file you use to tell Clearspace what the plugin includes, what version it is, and so on. Here's a simple example:

```
<plugin>
  <name>base64</name>
  <description>Converts text to Base64.</description>
  <author>Gladys Kravitz</author>
  <!-- This plugin's version. -->
  <version>1.0.0</version>
  <!-- The earliest version of Clearspace on which this plugin is supported. -->
  <minServerVersion>1.1.0</minServerVersion>

  <!-- The fully-qualified name of the class that contains the filter's logic. -->
  <filter class="com.jivesoftware.clearspace.plugin.filterexample.Base64Filter" />
</plugin>
```