

# Tutorial: Simple Clearspace Macro

Macros are a great ways to handle content in wiki documents, blogs, and discussions. With a macro, a user adds the macro tag to their document in plain text, then publishes the document to view the macro's result. An example is using the following to display text in red:

```
{color:red}This is red text.{color}
```

The result would be something like this:

This is red text.

A macro is one or two tags bracketed by curly braces. They look like this:

```
The macro at the end of this sentence could embed a {username}.  
Or you could format text by {blinking_lights}surrounding it with blinking  
lights{blinking_lights}.
```

With just a bit of code, you can add macros to those that Clearspace includes by default. In this tutorial you'll build a simple "Hello World" macro. Along the way you'll see the pieces that make up a basic macro – a plugin.xml file and the Java code behind the macro.

## Macro Basics

For simple macros, you'll really need just two pieces of code:

- A **macro class** that extends `com.jivesoftware.community.renderer.BaseMacro`. This class will implement the methods that Clearspace calls to get the String value for display in the user interface where the user has inserted your macro. If there are attributes (such as the "red" attribute in the macro above or enclosed text (such as "This is red text."), your code will receive those as well for processing. You'll learn more about those methods in the code included in this tutorial.
- Macros are technically plugins, so yours will include a **plugin.xml file** that describes the plugin to Clearspace, including which Java class is your macro class.

You package your macro in a JAR file and deploy it by copying it to the `<jiveHome>/plugins` directory of your Clearspace instance.

## What You'll Need to Get Started

This tutorial is about the basics, so you'll want only a few things to build all the pieces it describes:

- **Java 5.** Doesn't get more basic than that. If you don't have it, you'll find instructions for getting it in the readme of your...
- **Clearspace distribution** for testing; either Clearspace or ClearspaceX will work. Okay, that's pretty obvious, too. If you already have a Clearspace instance you can deploy to, then you're all set. (You'll also need to be able to log in to Clearspace as an admin.) This tutorial assumes you're using the standalone version (see the topic on setting up) because that's the simplest way to get started. You can download the standalone version from [jivesoftware.com](http://jivesoftware.com). (Don't use a production deployment for a first go-round with new code – even simple rock solid code such as you'll write here!)
- **Apache Ant** for building and deploying your code. To make compiling and deploying your plugin easier, the tutorial makes heavy use of an Ant build file you'll find at [Jivespace](#). You can get Ant at the [Apache web site](#).
- **An editor for code** – Java code and XML. IDEs such as Eclipse or IntelliJ IDEA are great, but more lightweight editors will do nicely, too.

That's about it. Check out the next section for a few setup steps, then you can get started writing code.

## Setting Up

These setup steps are likely pretty common to other extension work you'll do – whether with macros or plugins or themes.

- Set up Clearspace. You can skip this if you've already got Clearspace installed and set up.
- Create spaces or communities to test in. You can skip this, too, if you've got spaces or communities to test in.
- Create a place to put your macro project code.

### Set Up Clearspace

If you haven't installed Clearspace and used its setup tool, use the following instructions. If you have, you can skip this part. Here are the steps:

1. Install your Clearspace development instance using the installation instructions.
2. Run Clearspace and use its setup tool to set basic configuration options, such as the location of the `jiveHome` directory. If you're unfamiliar with the setup tool, use the following setup tool settings:
  - For `jiveHome`, use `<distribution_root>/jiveHome`
  - For License, accept "Evaluation".
  - For Database Settings, choose "Evaluation Database".
  - Accept defaults for the rest.

### Set Up Test Spaces

To test your macro you'll create a document and add your macro markup to it. This assumes you've set up spaces or communities in your Clearspace instance. You can test with any spaces or communities you like, but you might find it easier to follow the tutorial if you set up as described here:

1. If you haven't started Clearspace, start it now. With the standalone distribution, open a command prompt and execute the following at the distribution's root:  
**Linux/Unix/Mac**

```
sh start-clearspace.sh
```

### Windows

```
start-clearspace.bat
```

2. Navigate to admin console of your running instance with your browser, then log in. This is probably at a URL such as

```
[http://]&lt;hostname>:&lt;portnumber>/&lt;context>/admin
```

For example, by default you might use <http://localhost:8080/clearspace/admin>

3. Create a space (in Clearspace) or community (in ClearspaceX) for testing.
  - a. Click **Spaces > Create New Sub-Space**
  - b. In **Space Creation - Step 1**, select a parent space and enter the following values:
    - Space Name: My Content
    - Description of Space: <whatever you like>
    - Space display name: mycontent
  - c. In **Step 2**, accept **Open** and click **Next**.
  - d. In **Step 3**, click **Next**.
4. In the **Review** page, click **Finish**.

You don't need to add users if you don't have any. You'll log in as an admin during the tutorial.

**Note:** At this point, if you don't know where your application log files are, you might want to locate them. The logs can be very helpful for debugging. If you're using the standalone distribution, you'll find the log files at <dist\_root>/server/logs.

## Set Up Your Project

It's easiest to develop plugins such as macros if you have your source and compiled artifacts in certain places. In particular, Clearspace expects your compiled Java classes to live in a classes directory under your plugin's root, your plugin.xml file will need to be at the root, and so on.

### If You're Using Another Distribution

This tutorial (and the Ant build file that does with it) assumes you're using the Clearspace standalone distribution. Of course, you can use this tutorial with another Clearspace distribution you've set up for development and testing (such as a Clearspace WAR file deployed to your application server). You can also set a jiveHome directory that's external to the distribution (this is the best practice for production). If you do either of these, you'll want to edit a couple of properties in the included Ant build file (at [Jivespace](#)):

- Edit the server.home.dir property value to point to the root directory of your application server. Note that this is needed to find the servlet-api.jar file, so you might need to edit the <classpath> element, too.
- Edit the clearspace.lib.dir property value to point to the directory that contains the libraries deployed with Clearspace. This might be at a path such as <app\_server\_root>/webapps/clearspace/WEB-INF/lib. The build file will resolve JAR file

dependencies based on this path.

- Edit the `jiveHome` property to point to the `jiveHome` directory you specified when you ran the Clearspace setup tool. The build file will deploy your plugin JAR file to a `plugins` directory inside `jiveHome`.

## Create a Project Hierarchy

1. Create a "SimpleExamples" directory for your plugin code. If you're using the standalone Clearspace distribution with the included application server, set up your project directory at the following path inside the exploded distribution:

```
<distribution_root>/plugins/plugins/SimpleExamples
```

2. Grab the included Ant build file at [Jivespace](#) and copy it into the SimpleExamples directory you created.
3. Open a command prompt and navigate to the SimpleExamples directory.
4. Run the Ant target that creates the project directories:

```
ant create.plugin.dirs
```

Now that you've got a place for your code, you'll actually write some. In the next section you'll start out by adding a simple theme that customizes the Clearspace user interface.

## Create a plugin.xml File

Every plugin has one. This file tells Clearspace what's in the plugin – in short, what Clearspace features you're extending – and where to find code and other supporting files your plugin needs (although not necessarily all of them, as you'll see in the next section).

1. Create a text file called `plugin.xml` in the root directory of your project (where the Ant build file is).
2. Paste the following into the new file:

```
<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.jivesoftware.com/schemas/clearspace/1_1/plugin.xsd">
  <name>SimpleExamples</name>
  <description>Simple macro and plugin examples.</description>
  <author>Me</author>
  <version>1.0.0</version>
  <minServerVersion>1.4.0</minServerVersion>

  <!-- The <macro> element tells Clearspace that a macro is
    included in the plugin. The class attribute points to the
    class that provides logic for the plugin. -->
  <macro name="sayhello" hasBody="false"
class="com.example.clearspace.plugin.macro.SimpleMacro" />
</plugin>
```

The `<plugin>` element's children include information about where the plugin is coming from (you), its version (in case you revise it for upgrade), and so on. Note the `<macro>` element, which points to a Java class you'll make in the next step. The `plugin.xml` can contain information about multiple bits of plugin functionality. For another kind of plugin, see `d\1012`.

## Create a Plugin Class

Your plugin class will provide the logic to tell Clearspace what to do when the user uses your macro.

**Note:** The API for building macros will be greatly simplified in future versions of Clearspace.

1. In the src directory created by the Ant build file, create a Java file called SimpleMacro.java and put it in a package called com.example.clearspace.plugin.macro.
2. Add the following code to the new file. Take a look at the comments before you move on; they describe calls to the Clearspace API you might find yourself using in your own macros.

```
package com.example.clearspace.plugin.macro;

import java.util.Map;

import com.jivesoftware.base.User;
import com.jivesoftware.base.plugin.Macro;
import com.jivesoftware.base.plugin.MacroContext;

/**
 * A "Hello World" macro that merely displays a greeting with the
 * username of the currently logged in user.
 */
public class SimpleMacro implements Macro {

    // For display if the current user name can't be returned.
    private String guestString = "Guest";

    /**
     * Called by Clearspace to retrieve HTML markup for the macro.
     *
     * @body The text between the macro tags, if any.
     * @param parameters Macro parameters set by the user, if any.
     * @param macroContext Information about the macro's context.
     */
    public String render(String body, Map<String, String> parameters, MacroContext
macroContext) {
        // Return the personalized message.
        return "<p>Hello World! This is " + getUsernameValue(macroContext) + "!</p>";
    }

    /**
     * Gets the username of the currently logged in user; returns "Guest" if that name
     * is not available.
     *
     * @param macroContext The context for this macro.
     * @return The logged in user's username or "Guest".
     */
    private String getUsernameValue(MacroContext macroContext) {
        User currentUser = macroContext.getUser();
        if (currentUser != null) {
            return currentUser.getUsername();
        }
        else {
            return guestString;
        }
    }
}
```

That's all the source code you'll need for this macro. Now it's time to...

## Build, Deploy and Test the Macro

Use Ant to compile and package the code into a JAR file, then copy your macro into the <jiveHome>/plugins directory.

1. At your command prompt, run

```
ant build.plugins
```

to build, then

```
ant deploy.plugins
```

to deploy to your server.

There's no need to restart your server, but you'll want to wait five or ten seconds for Clearspace to deploy the plugin for use.

2. Open your browser to Clearspace and click the My Content space you created.
3. In the My Content space, create a new document called "Just saying hello".
4. Add whatever content you like, but be sure to add your macro's tag. So you might have something like the following:

```
Here's a test of a new macro.  
{sayhello}  
And that concludes this test.
```

5. Publish the document.

After you've published the document, you should see something like this:

Cannot resolve external resource into attachment.

That's it for building a simple macro! Be sure to check out [Jivespace](#) for more content and samples.