**Example: Creating and using a Jenkins shared library**

**So now the only thing left to do in this guide is show you how to do it, for real.**

In this section, we’ll see how to set up a shared library in Jenkins, with a quick example.

1. **Create the shared library**

First you need to create a Git repository which will contain your library of functions (steps). (You can also use Subversion.)

In your repository, create a directory called vars. This will hold your custom steps. Each of them will be a different .groovy file underneath your vars directory, e.g.:

vars/

sayHello.groovy

**Add your custom steps**

Each of your custom steps is a different .groovy file inside your vars/ directory. In Jenkins terminology, these are called *Global Variables*, which is why they are located inside vars/.

Create a file for your custom step, and fill in the code. For example, a simple greeting function would look like this:

#**!**/usr/bin**/**env groovy

**def** **call(**String name **=** 'human'**)** **{**

echo "Hello, ${name}."

**}**

Notice how the Groovy script must **implement the call method**.

After writing that, you should write your custom code within the braces { }. You can also add parameters to your method - the example above has one parameter called name, which has a default value of human (cos we’re being really personal here.)

1. **Set up the library in Jenkins**

Now you’ve created your library with custom steps, you need to tell Jenkins about it.

You can define a shared library within a *Jenkinsfile*, or you can configure the library using the Jenkins web console. Personally, I think it’s better to add from the web console, because you then you can share the library across all of your build jobs.

To add your shared library (I’m using my demo repository on GitHub as an example):

**In Jenkins**, go to Manage Jenkins → Configure System. Under *Global Pipeline Libraries*, add a library with the following settings:

* + Name: pipeline-library-demo
  + Default version: Specify a Git reference (branch or commit SHA), e.g. master
  + Retrieval method: **Modern SCM**
  + Select the **Git** type

1. Project repository: <https://github.com/chandanchanchal/pipeline-library-demo.git>
2. **Use the library in a pipeline**

To use the shared library in a pipeline, you add @Library('your-library-name') to the top of your pipeline definition, or *Jenkinsfile*. Then call your step by name, e.g. sayHello:

**@Library(**'pipeline-library-demo'**)**\_

stage**(**'Demo'**)** **{**

echo 'Hello world'

sayHello 'Dave'

**}**

**NOTE:** The underscore (\_) is **not a typo!** You need this underscore if the line immediately after the @Library annotation is not an import statement.