how to run a complete Jenkins scripted pipeline stage from shared libraries src file

I recently started with Jenkins shared libraries in Jenkins pipeline. I created a "func.groov" class and located it under "src/org/prj/func.groovy" :

```
package org.prj
import jenkins.model.
class func implements Serializable {
 def steps
 func(steps) {
  this.steps = steps
 }
 def sh(args) {
  steps.sh "echo ${args}"
 }
 def setLBL(CurrentNodeName,newLabelName){
   jenkins.model.Jenkins.instance.slaves.each{ slave ->
     if (slave.getNodeName() == CurrentNodeName){
       slave.setLabelString(newLabelName)
     }
  }
}
```

Jenkinsfile (scripted) looks like:

```
@Library('prj') import org.prj.func
def utils = new func(steps)
node(lbl)
{
  stage("A"){
     Build_node_lbl = env.NODE_NAME+System.currentTimeMillis()
     utils.setLBL(env.NODE_NAME,Build_node_lbl)

    }
}
```

so currently it works. my question is how to create a full stage (like "A") as a function in func.groovy shared lib which will include, for example:

1. GIT checkout step
2. sh compilation step
3. Artifactory deploy step

Im actually looking to create a "building blocks" (a "Build" in my example) with Jenkins pipeline and shard libraries.

jenkins groovy jenkins-pipeline jenkins-shared-libraries

1 Answer

1. With Class Instantiation

You can create a class like you would do in Java. Then in your `Jenkinsfile` you instantiate the class and call its function.

src/org/prj/MyPipeline.groovy:

```
package org.prj

class MyPipeline {

  def steps

  MyPipeline(steps) {this.steps = steps}

  public def build() {
    steps.node('lbl') {
      steps.stage('A') {
        // Do build stuff
        // steps.sh(..)
      }
    }
  }
}
```

Jenkinsfile:

```
import org.prj.MyPipeline

def pipeline = new MyPipeline(this)
pipeline.build()
```

2. With Static Functions

You may also work with static contexts, without instantiation. However, this would require to hand over the caller context to the pipeline:

src/org/prj/MyPipeline.groovy:

```
package org.prj

class MyPipeline {

  public static def build(caller) {
    caller.node('lbl') {
      caller.stage('A') {
        // Do build stuff
        caller.sh(..)
      }
    }
  }
}
```

Jenkinsfile:

```
import org.prj.MyPipeline

MyPipeline.build(this)
```

1 Fwiw you won't have access to Jenkins' steps inside your class. In the first example "node" and "stage" would throw exceptions. You have to pass the Jenkinsfile context to the constructor (for instance) and then use this object to call Jenkins' steps. – DevAntoine Aug 14 at 15:56

You are absolutely right (jenkins.io/doc/book/pipeline/shared-libraries/#accessing-steps). I updated the answer. Thank you for pointing it out! – fishi Aug 14 at 16:51