Jenkins Adaptive Plugin

Plugin Information
Distribution of this plugin has been suspended due to unresolved security vulnerabilities, see below.

The current version of this plugin may not be safe to use. Please review the following warnings before use:

- Arbitrary code execution vulnerability

This (experimental) plug-in exposes the jenkins build extension points (SCM, Build, Publish) to a groovy scripting environment that has some DSL-style extensions for ease of development.

Description

Sometimes when using Jenkins, one comes across the situation whereby the system doesn't /quite/ do what you want it to, and you need to extend a particular plugin in order to get something done. There are occasions where you want this to be done inside the SCM - either for doing multiple checkouts, or in responding to the 'is this job requiring a new build' trigger, sometimes as a build step, and sometimes as a post-build, publishing action.

The adaptive plugin is similar to the groovy plugin and groovy post build in that it is scripted using groovy. However, it has a number of differences.

Firstly, the script is stored in one place in the project - so if you have elements that are invoked at different phases, they are visible next to each other in the script.

Secondly, the script can bind to the SCM stages of the build

Finally, the script uses a DSL in order to try to simplify access to particular hudson resources. However, this is still experimental!

Example Script

```
jenkins {
  scm {
    poll {    PollingResult.BUILD_NOW  }
    checkout {
        checkout GitSCM("git.repo.url":"git://github.com/magnayn/Ribbons.git",
                        "git.repo.name":"origin",
                        "git.repo.refspec":"",
                        "buildChooser":[ "stapler-class":"hudson.plugins.git.util.DefaultBuildChooser" ]
);
     }
  }


  build {
     println "Trying to build";
     invoke Maven( rootPOM:"pom.xml",
                   targets:"clean install"     );
  }


  publish {
       def ab = new AntBuilder();
       ab.taskdef(name: 'scp', classname: 'org.apache.tools.ant.taskdefs.optional.ssh.Scp');
       ab.scp(file:"/tmp/file.txt", todir:"repo:repo@internalrepo:/home/repo" );
  }
}
```

}

This script tells Jenkins every time it asks that 'yes, I need to build now'. It calls Git to check out the code, Maven to invoke the build, and finally utilises the AntBuilder to SCP it to a remote server.

Calling plugins

The general format is to use

```
invoke <pluginName>( <pluginParameters> )
```

The parameter names are those that would have been returned by the web (stapler) page. However - these are not particularly exposed for end-users, so some sourcecode digging is sometimes needed.

Utility functions

There are several convenience features for common things such as finding a project. E.g, you can do

```
build project("myProject")
```

To trigger a build of myProject

Build Chains

One of the experimental ideas in the plugin is one of build chains - that you can specify an ordered set of builds to be done:

```
def chain = project("test") >> [project("test2"), project("test3")] >> project("test4")
```

You can then build those projects in order (stopping if fails occur)

```
build chain
```

Or even determine if anything within it requires building:

```
jenkins {
  def chain = project("test") >> [project("test2"), project("test3")] >> project("test4")
  scm {
    poll { poll chain; }
  }
}
```

The 'chain' functionality is an (unfinished) idea whereby you may wish that if there is a projectA, build it and all the downstream builds; then IF and ONLY IF they are all successful, then commit that change to the main branch in the SCM. This allows a 'master project' to give the illusion of atomic commits, even if it is comprised of multiple, disparate repositories. So, for example, your developer commit to branch 'dev' may trigger a chained build. If the build of the project is successful (individually), then the commit is promoted to 'master'. If the full chained build (including downstream integration tests), then it is additionally promoted to 'release'.

Caveats:

This is still a work-in-progress;

When it fails (particularly using hpi:run rather than a full jenkins install) the classloader seems to explode in interesting ways...

Version History

0.1 : Initial version

Source Repository

git://github.com/magnayn/Jenkins-AdaptivePlugin.git