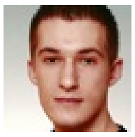
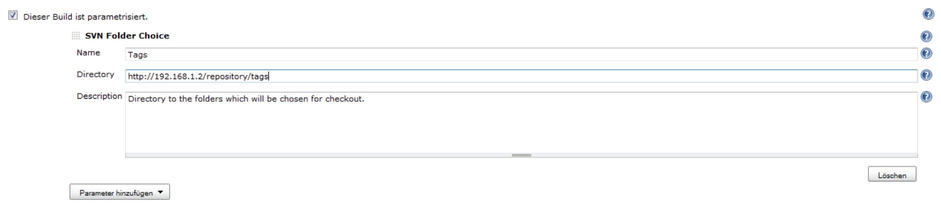
Parameterized Builds in Jenkins – choosing subversion folders

 Dieser Artikel wurde von Michael Döcke geschrieben.

25 Mai, 2011 Java, Programmierung

For software projects with a large amount of code it is essential to emphasize maintainability. Continuous Integration Systems, such as Jenkins CI, are good instruments to simplify and monitor your builds in an automatic way. An advantage of Jenkins CI – the possibility to integrate some, possibly self-written plug-ins – is the topic of a small project that could be helpful for some Jenkins/Hudson-using developers.

SVNFolderParameterPlugin is a plug-in that provides forms for parameterized builds where you can browse in a Subversion repository for a subdirectory in a specified directory. Together with a custom check-out script, this allows building Jenkins jobs which run against arbitrary source code versions in your Subversion (e.g., in order to rebuild a tagged version). When starting the build, all subfolders will be shown in a dropdown list and the URL of the selected folder will be forwarded as a property to the build script.

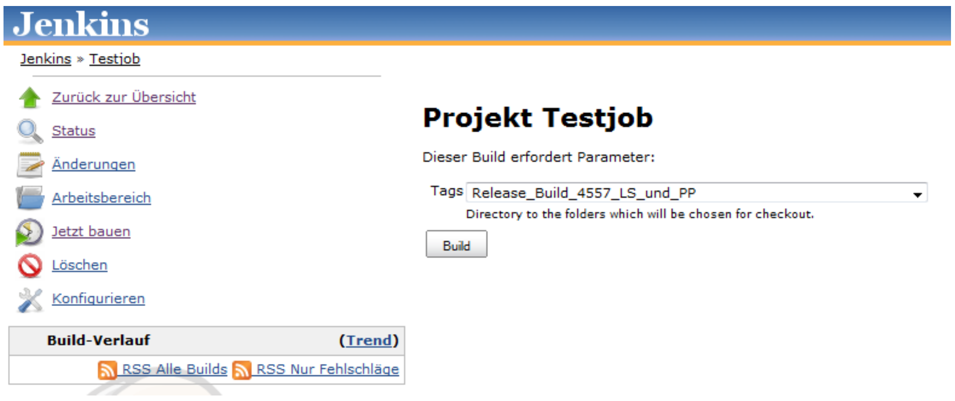


☑ Dieser Build ist parametrisiert.

SVN Folder Choice

Name	Tags
Directory	http://192.168.1.2/repository/tags
Description	Directory to the folders which will be chosen for checkout.

Fig. 1: Added parameter in job configuration.



Projekt Testjob

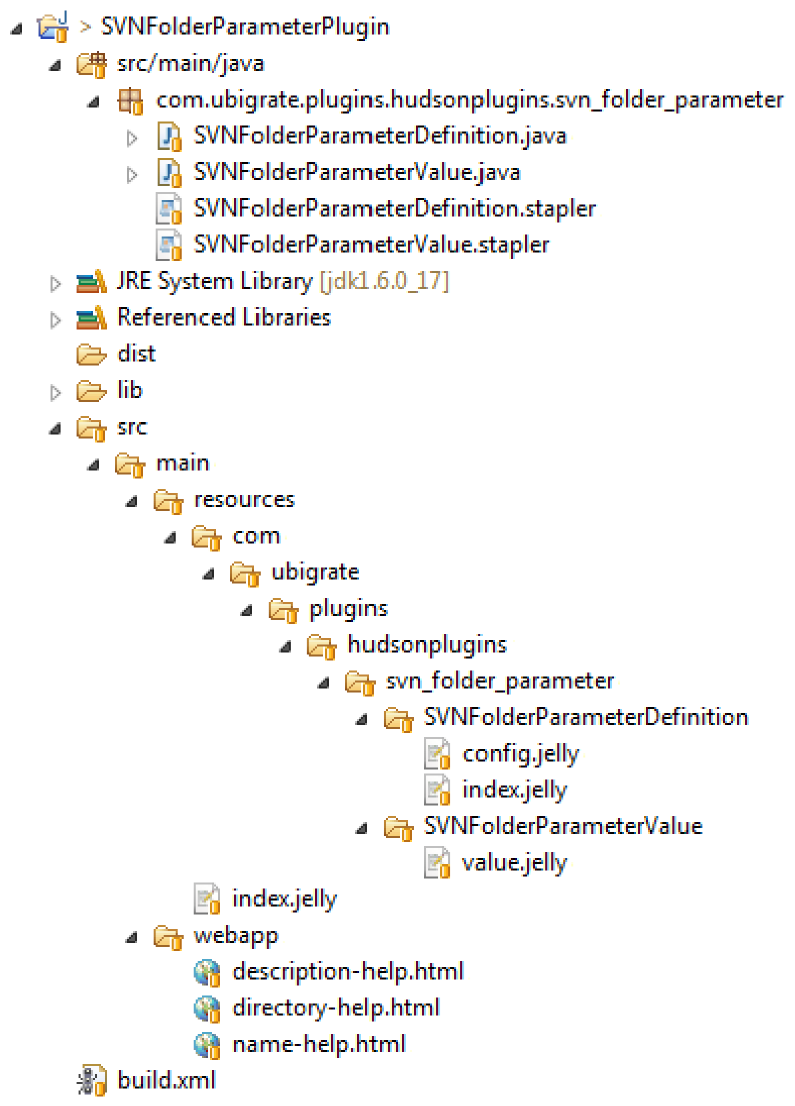
Dieser Build erfordert Parameter:

Tags Release_Build_4557_LS_und_PP

Directory to the folders which will be chosen for checkout.

Fig. 2: Combobox filled with folders of the configured Subversion path while starting the build.

The project structure consists of 3 parts which can be found in the ‚src/main‘ folder. While ‚java‘ is clearly the folder for java source files, ‚resources‘ (with jelly files) and ‚webapp‘ (with html files) will be used for displaying web forms by filling their controls with the member-values of the corresponding java objects. The assignment between the jelly and the java files will be achieved by putting the *.jelly in ‚resources‘, in the same subfolder as the java source code and a further folder which is named exactly like the class you refer to. Since Hudson/Jenkins uses Stapler as its web framework, *.stapler files with a constructor definition have to be created next to the java files (again, with the same name), to be used by Stapler.



```
◢ > SVNFolderParameterPlugin
    ◢ src/main/java
        ◢ com.ubigrate.plugins.hudsonplugins.svn_folder_parameter
            ▷ SVNFolderParameterDefinition.java
            ▷ SVNFolderParameterValue.java
              SVNFolderParameterDefinition.stapler
              SVNFolderParameterValue.stapler
    ▷ JRE System Library [jdk1.6.0_17]
    ▷ Referenced Libraries
      dist
    ▷ lib
    ◢ src
        ◢ main
            ◢ resources
                ◢ com
                    ◢ ubigrate
                        ◢ plugins
                            ◢ hudsonplugins
                                ◢ svn_folder_parameter
                                    ◢ SVNFolderParameterDefinition
                                          config.jelly
                                          index.jelly
                                    ◢ SVNFolderParameterValue
                                          value.jelly
                  index.jelly
            ◢ webapp
                  description-help.html
                  directory-help.html
                  name-help.html
      build.xml
```

Fig. 3: Project structure.

As seen in the figure above, only two java source files were necessary in this plug-in. *SVNFolderParameterDefinition.java* provides functions for resolving the folders for a given Subversion path, for validating user input for the corresponding form and accessing member variables by Stapler. *SVNFolderParameterValue.java* is a simple Bean-like class which holds the value of the user selection during the build step after submitting the parameters, which is essential for passing the value to the build script.

The form controls shown in Figure 1 will be created by the *config.jelly* and in Figure 2 by the *index.jelly* in ‚src/main/resources/com/ubigrate/plugins/hudsonplugins/svn_folder_parameter/SVNFolderParameterDefinition‘. Adding the files in this directory defines which object should be used while filling the form elements by accessing the it- or instance-object fields and getting their values. Accessing these values is done via the “getter”-method of the denoted field.

With this you will be able to access the value parameter from *SVNFolderParameterValue.java* in *value.jelly* in the following way (considering a correct folder structure):

```
<j:jelly xmlns:j="jelly:core" xmlns:f="/lib/form">
  <f:entry title="${it.name}" description="${it.description}">
    <f:textbox name="value" value="${it.value}" readonly="true"/>
  </f:entry>
</j:jelly>
```

The *index.jelly* from SVNFolderParameterValue is only kept for checking the parameter value after the build.

You can download the project including the sources at:

http://svn.ubigrate.com/tools/trunk/SVNFolderParameterPlugin

Use anonymous:anonymous for login.

The source can be extended easily with additional parameters such as a login field if you don't want to provide an anonymous read-only subversion-account or some other important arguments which characterize or constrain the access to the repository. You just need to edit the *config.jelly* and *index.jelly* for the forms and *SVNFolderParameterDefinition.java* to handle the input.

Since Jenkins/Hudson allows only *.hpi files as plug-ins, an appropriate build script is needed. The *build.xml* in the root folder uses the ant war task to create the necessary *.hpi file. You still have to copy the libraries from $HUDSON_HOME/war/WEB-INF/lib into the projects lib folder. Additionally two more libraries are required:

- svnkit.jar (from org.tmatesoft.svn_1.3.5.standalone)
- servlet-api-2.5-20081211.jar

For building this plug-in it's sufficient to use the default target. You will have noticed that while most of the other Jenkins plug-ins were built with Maven, this one uses ANT. The simple reason for this is to comply with our other builds.

The entire plug-in (SVNFolderParameterPlugin.hpi) is available here.

Just use the advanced options tab in the plug-in manager in your Jenkins/Hudson environment and upload the plug-in. Another possibility is to copy the hpi file into the plugin folder of your Jenkins/Hudson installation. Consider that a restart of Jenkins/Hudson is necessary in both cases.

For developing your own plug-in take a look at the Jenkins plugin tutorial.

Schlagworte: Java • jenkins • plugin • subversion

1 Kommentar bisher »

1. **Matt Doar sagt**

   am 4. August 2011 @ 19:52

   Nice plugin, but the parameter does not appear to be exported to the REST API using @Exported.

~Matt

Komentar RSS · TrackBack URI

Hinterlasse einen Kommentar

Name:

eMail:

Website:

Kommentar:

Sag es

Speichern bei

- ## Letzte Artikel
  - Was ist eigentlich Business Activity Monitoring?
  - Parameterized Builds in Jenkins – choosing subversion folders
  - Building a custom UI Object with GT Designer 2 for Mitsubishi GOT 1000
  - Capture-Replay Regression Testing in Eclipse RCP
  - ServiceBrowser: browsing your rails controllers
- ## Schlagworte

  AOP ASM AT Automatisierung BCEL Bluetooth Bytecode Bytecode-Engineering C-MUX CEP Demo DPWS Eclipse Energy Monitoring Escape Flex Gastbeitrag GPS HMI Instrumentierung Java Javassist JAXB JUG Saxony Logistik Mindstorms Performance Preis Produktion RCP RFID RS232 RS485 SAP SOA Sun SPOT Telnet ubigrate usability USB Web Service wireless XJC XML Schema Zukunftsforum
- ## Kategorien

- Allgemein (21)
  - Anwendungen (10)
  - Forschung (1)
  - GUI (1)
  - JUG Saxony (5)
  - Markt (1)
  - Programmierung (15)
    - C (1)
    - Flex (4)
    - Java (10)
  - Technik (11)
    - Automatisierung (1)
    - Protokolle (4)
  - ubigrate (8)

- ## Archiv

  - Januar 2012 (1)
  - Mai 2011 (1)
  - Mai 2010 (1)
  - Oktober 2009 (1)
  - Juni 2009 (1)
  - Mai 2009 (2)
  - April 2009 (2)
  - März 2009 (2)
  - Februar 2009 (1)
  - Januar 2009 (2)
  - Dezember 2008 (1)
  - November 2008 (3)
  - Oktober 2008 (5)
  - September 2008 (2)
  - August 2008 (2)
  - Juli 2008 (2)
  - Juni 2008 (4)
  - Mai 2008 (7)
  - April 2008 (5)

- ## Letzte Kommentare

  - Marcus bei Was ist eigentlich Business Activity Monitoring?
  - RalfLippold bei Was ist eigentlich Business Activity Monitoring?
  - Md Sirajuddin bei Creating Flex UIs in Java – A Short Tutorial
  - Matt Doar bei Parameterized Builds in Jenkins – choosing subversion folders
  - RalfLippold bei Neues vom RFID-Markt: Welche Krise?

- ## Sonstiges

  - Anmelden
  - Artikel-Feed (RSS)

    - Kommentare als RSS
    - WordPress.org

LetzteBeiträge

- Was ist eigentlich Business Activity Monitoring?
- Parameterized Builds in Jenkins – choosing subversion folders
- Building a custom UI Object with GT Designer 2 for Mitsubishi GOT 1000
- Capture-Replay Regression Testing in Eclipse RCP
- ServiceBrowser: browsing your rails controllers

MeineLinks

- ubigrate Website
- Java Usergroup Saxony

AndereBlogs

- M2M Blog
- The RTLS Blog

ubigrate smart device integration