Atlassian Bamboo, multiple projects, one repository and shared libraries

by Benny Bottema - Friday, April 01, 2011

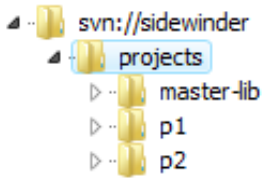
http://www.bennybottema.com/2011/04/01/atlassian-bamboo-multiple-projects-one-repository-and-shared-libraries/

Having several projects in Bamboo that share a repository, build script and libraries, it can be tricky to configure Bamboo build plans correctly; but it isn't hard at all. I'm going to quickly show you how I configured Bamboo Plans for two projects that are in the same repository, but share a master library. This is a very common setup and the point is to make Bamboo build only for the plans if their associated project has been updated.

In addition to the shared master library folder, I'm going to include a master *build.xml* file as well and see how that works with the nested projects.

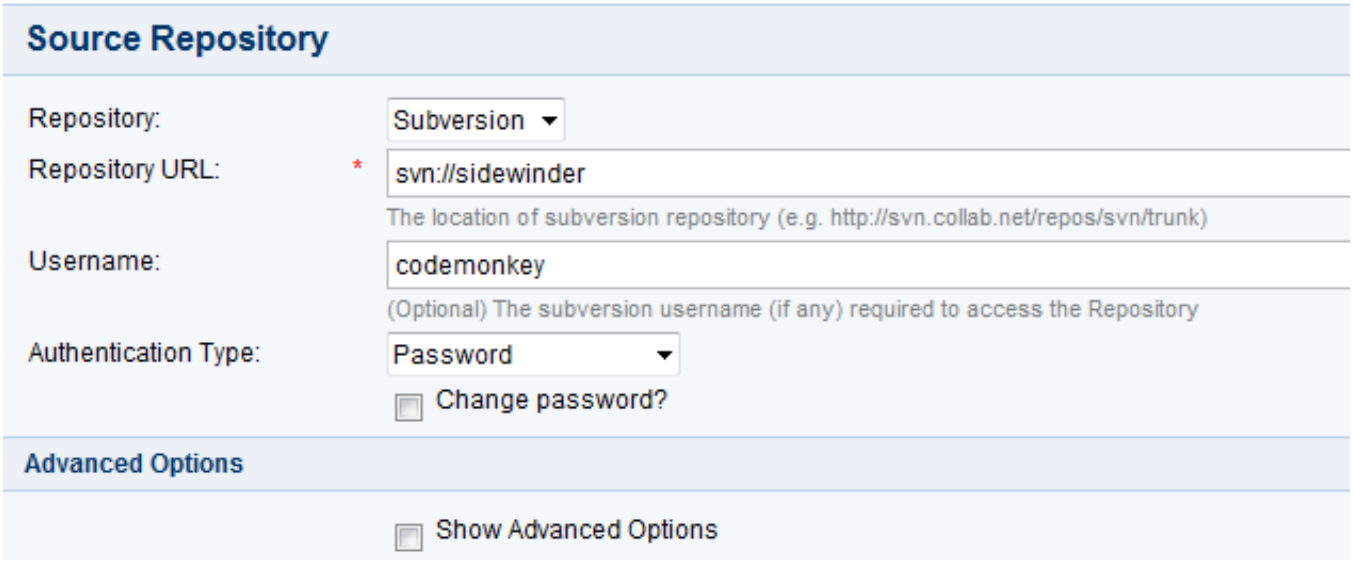
Basic setup

Here's my project setup in subversion (note that these names are of course simplified):



Very simple: one root folder with **Project 1**, **Project 2** and one **master library and build script**. The project structure is according to the Maven standard (*src/main/java* etc), but this is not really relevant right now.

Bamboo is configured to look at the root of this repo. This is nescesary if we want Bamboo to see (and update) the master folder as well. Here's the basic configuration for both Bamboo Plans (look under tab *Configuration -> Source Repository*):



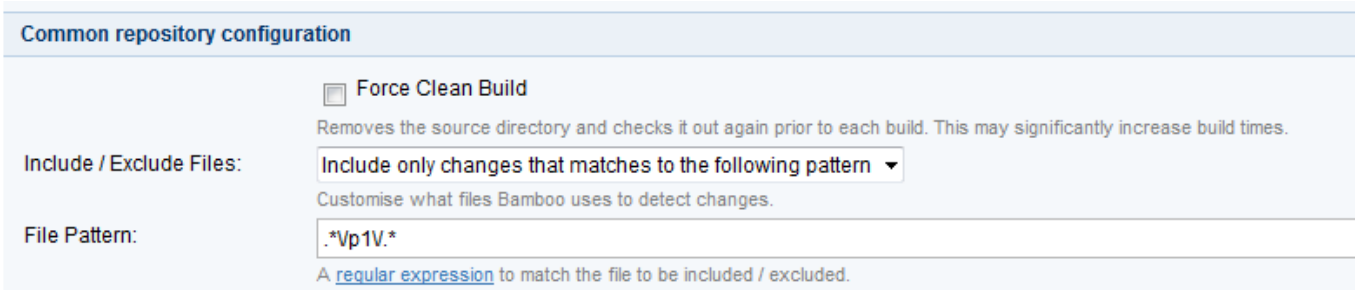
Basic plan configured with my repo, the same for all build plans

Next up is getting bamboo to trigger a plan only for commits for that plan.

Triggering the right plan

As you can see above, you can only indicate in one place where Bamboo looks for your code. Now we need to further configure how Bamboo builds the projects within this code. Furthermore if we commit changes to *project 1*, we don't want *project 2* to be built as well.

We'll need to tell Bamboo to only trigger when changes happen within the repository for the correct project. Under **Common repository configuration**, select **Include only (..)** and then fill out the File Pattern to look for files within the projects folder in svn:



This filter includes only changes to p1

This example is for the Project 1 build plan and it tells Bamboo that when changes are committed to the indicated repository, check if the changes comply with the given regular expression. Nifty eh. Now I'm not sure you need to escape those forward slashes, but I had some trouble testing my expressions and I know at least this works correctly.

If you're having trouble here, you may want to keep an eye on the Bamboo logfile: it will tell you based on the include/exclude filter which commits are included or excluded. You can use the following command under Ubuntu: **tail -f -n100 atlassian-bamboo.log**. The log file should be under */usr/bamboo/*.

You should look for INFO lines from the *DefaultChangeDetectionManager* class.

Now we've got several projects, built apart from each other based on commits for these respective projects. Awesome! Next up: sharing a master build and lib between projects.

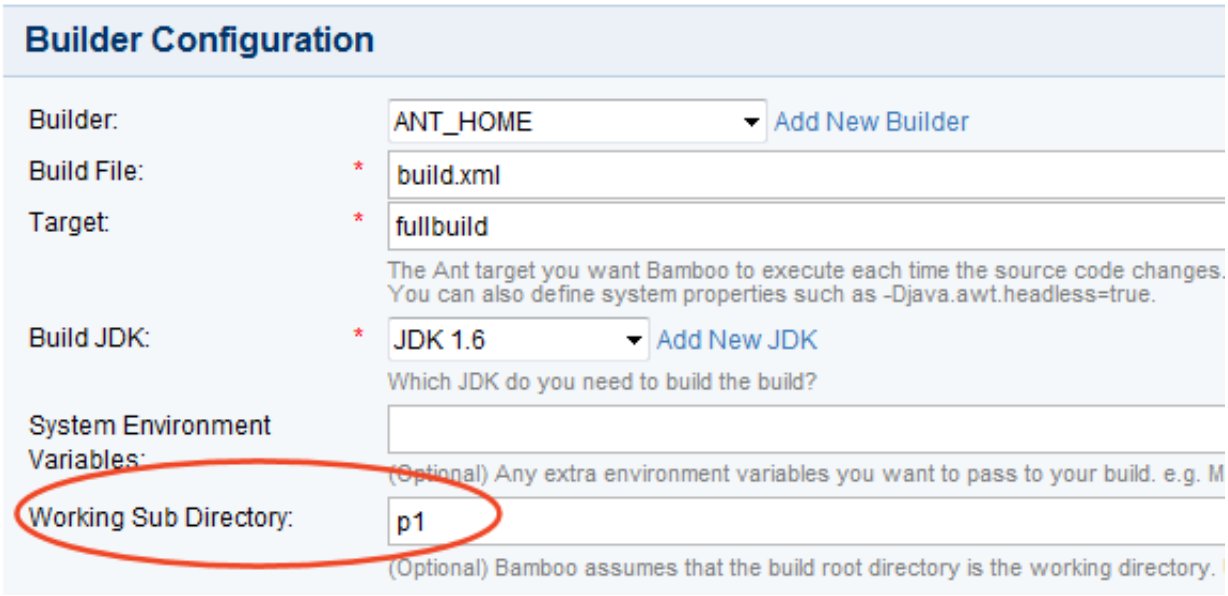
Sharing a master build script and libraries

Project 1 and Project 2 use a small template build.xml that imports the master build script and adds some details specific for that project. For example, here's the build.xml file for Project 1:

```
<?xml version="1.0"?>
<project name="Project1" basedir="." default="fullbuild">
 <property name="application.name" value="project-1" />
   <import file="../master-lib/master-build.xml"/>
</project>
```

The master-build.xml will assume a default project structure and work with that. The trouble is, the Project 1 build script needs to be run from the root of Project 1, which Bamboo won't do by default considering we indicated the very root of the svn repository. We could make the master-build.xml path agnostic by using an extra variable but this makes things unnecessarily complex.

We can simply tell Bamboo to use a working directory to run the ant script from, as seen under the tab *Configuration -> Builder*:



Tell Bamboo to call the build script from p1

And that's it, now Bamboo will execute the ant targets under the correct project root. Configure the plan for Project 2 the same way and they will be built seperate from each other triggered by the right commits

and using a parent master-lib.

One thing you may wish to add to all build plans is that they are also triggered by changes to the master-lib folder. Simply adapt the *File Pattern* again to include that folder.
