Remote deployment with Ant and ssh

You can do many things with Ant, one of them is the ability to use tools like *tar*, *gzip*, *FTP* and *scp* for remote deployment. At work, we use a shell script on our servers to deploy our apps, and we are using *rsync* to send the applications to the server before calling the shell script. Would be great to actually do this in one step ? It's quite easy!.

First, you need to install JSch. We use the *jsch-0.1.29* release, we didn't try a later release. Copy the JAR into */Developer/Java/Ant/lib/* (Mac OS X 10.4) or */usr/share/ant/lib/* (Mac OS X 10.5).

Next, you need to add this JAR to the Ant lib list inside Eclipse. Open Eclipse's preferences (Eclipse->Preferences), open the *Tab* triangle and click on *Runtime*. Select *Ant Home Entries* and click on *Add External JARs...* Browse up to */Developer/Java/Ant/lib/* and select *jsch-0.X.XX.jar*. The JAR should now be part of the list of JARs available for Ant.

> If you use multiple workspaces, you have to add the JAR to each of your workspaces to the *Ant Home Entries* list.

Ok, now it's time to actually create a Ant task for deployment. You can add a property like this to *build.properties*:

```
servers.production=my.server.address
```

And you add this to your *build.xml* file.

```
<target depends="setProps,init.install,build.woapp"
name="deployProduction">
  <echo message="Starting file transfer to
${user.name}@${servers.production}" />
  <exec dir="." executable="rsync" os="Mac OS X" failonerror="true">
    <arg value="-aog" />
    <arg value="-e ssh" />
    <arg value="${dest.dir}/${project.name}.woa" />
    <arg value="${user.name}@${servers.production}:~" />
   </exec>
   <sshexec command="myshellscriptfordeployment -d ${project.name}.woa"
host="${servers.production}" keyfile="${user.home}/.ssh/id_rsa"
passphrase="" username="${user.name}"/>
</target>
```

Wait a minute... Maybe *Ant* will complain because it can't send the password to the remote server. How to fix this ? By creating a SSH public key if you don't already have one.

First, check if you already have a public key on your computer:

```
$ ls -al ~/.ssh/id_rsa.pub
-rw-r--r--   1 monuser  monuser  230 Dec  5  2006 .ssh/id_rsa.pub
```

No *id_rsa.pub* file ? Create one:

```
$ ssh-keygen -t rsa
```

Now copy your SSH public key to your remote server:

```
$ scp ~/.ssh/id_rsa.pub mynuser@remoteserver:myuser.pub
```

and put your public key in the *authorized_keys* file:

```
remoteserver$ cat ~/monuser.pub >> ~/.ssh/authorized_keys
```

Now, next time that you login by SSH from your computer to the server, it will stop asking for a password and the Ant task will stop complaining. For added security, when you create your key, use a passphrase. Don't forget to put the passphrase into the *sshexec* task. Don't use the passphrase if you put the *build.xml* file in CVS or SVN, because your co-workers will see your passphrase and also they won't be able to use the sshexec task unless they use the same passphrase as you.