Subversion Service

February 2011

Contents

1. Introduction

The following manual shows how to access the RDlab Subversion repository service, and how to perform key tasks in text mode (command line).

There are other Subversion clients with graphic interface that are beyond the scope of this guide. Some of the most popular are Subclipse [1] and Tortoise [2].

Were you looking for information on the SVNFrontend application, please refer to the manual at:

http://rdlab.lsi.upc.edu/index.php/en/services/documentation.html

2. Accessing the service

In order to get access to the service, a username and password is needed. These can be requested by the form located at the page http://rdlab.lsi.upc.edu/index.php/es/servicios/peticionacceso.html or by e- mail at rdlab@lsi.upc.edu.

Once received, the user will have a repository for his projects. It can be checked at http://svn-rdlab.lsi.upc.edu/subversion/<*username*>.

3. Putting initial data into the repository

If the project does not yet exist in the repository, the first step is to create and upload the contents of the project into the repository. To this a temporary folder has to be created and the contents of the project must be copied into the folder. Let's suppose there is a folder named "myProject-temp" and that contains a file named "HelloWorld.java". From this folder the command "svn import" must be run:

```
$ svn import . http://svn-rdlab.lsi.upc.edu/subversion/
<username>/myProject -m "first import" --username
<username>
```

The server will ask for password:

```
'<username>' password:
```

Once introduced, the system will show the following output:

```
Adding myProject/HelloWorld.java Committed revision 1.
```

This process has created a folder named "myProject" in the repository with the contents of the project and with the comment "first import".

Now the temporary folder previously created for the first import can be deleted:

```
$ rm -rf myProject-temp
```

4. Creating a working copy

If project already exists in the repository a working copy can be created. The working copy is the folder on your computer that contains the copy of the project to be synchronized with the copy at the repository.

Note that the command "svn import" above just uploaded the content of the project to subversion server, but has not created a working copy. To do so a new folder must be created and the command "svn checkout" must be run:

```
$cd ..
$mkdir myProject
$svn checkout http://svn-rdlab.lsi.upc.edu/subversion/
<username>/myProject/    myProject --username
<username>
```

The system will show the following output:

A myProject/HelloWorld.java Checked out revision 1

At this point a copy associated with the repository exists. The local copy may be edited and changes can be uploaded to the repository (as explained in the following section) or the local copy can be updated with changes made by other users in the repository (section 7).

5. Editing the working copy

If changes to the "working copy" are made and have to be incorporated into the repository, the command "svn commit" has to be run from the folder of the "working copy":

```
$ cd myProject
$ svn commit . -m “HelloWorld modified" --username
<username>
```

If, for instance, we have modified the HelloWorld.java, the system will show the output:

```
Sending        HelloWorld.java
Transmitting file data .
Committed revision 2.
```

6. Adding files to the project

If new files or folders in the working copy are created and have to be incorporated into the repository, the command "svn add" must be run to add the files to the project. Afterwards the "svn commit" command must be run in order to load them into the repository as explained in section 5. If, for instance, a new file "newFile.java" has been created:

```
$ svn add newFile.java --username <username>
```

The system will show the following output

```
A        newFile.java
```

7. Updating the working copy

If another user has made changes to the project files (and incorporated these changes into the repository), the working copy can be updated with these changes using the “svn update” command:

```
$ svn update . --username <username>
```

The system will show the following output:

```
U        myProject/HelloWorld.java Updated to revision 3.
```

The "svn update" command allows also to retrieve a previous revision of a file or an entire folder using the "-r" option followed by the revision number. For instance, to retrieve the revision 2 of "HelloWorld.java" file:

```
$ svn update ./myProject/HelloWorld.java –r2 –username
<username>
```

To retrieve the whole “myProject” folder at revision 2:

```
$ svn update ./myProject –r2 --username <username>
```

8. External links

1. Subclipse (subversion for Eclipse):
   http://subclipse.tigris.org/

2. Tortoise (subversion for Windows):
   http://tortoisesvn.net/

3. Subversion, oficial page
   http://subversion.apache.org/

4. Book “Version control with Subversion” (O'Reilly Media)
   http://svnbook.red-bean.com/