Eclipse CVS Plugin Basic Usages

Table of Contents

1. Revision History

Author	Date	Version	Description
Jeff Yu	12/02/2005	1.0	Initial Version

2. Terminology

Term	Definition

3. Content

In this document, I just try to write some basic usages about eclipse CVS plugin, because for a developer, what we often use are some basic usages, if you want to learn cvs usage more, there are a lot of books on the Internet, and I also recommend a book called <Pragmatic.Version.Control.Using.CVS>, and this document is according to my previous experience using CVS, so feel free to correct it if you find something wrong here. ;-)

3.1. CONNECT TO CVS SERVER

1. Start up Eclipse, and then select Window -> Open Perspective -> Other -> CVS Repository Exploring.
2. In the Perspective, click the right mouse, Select New -> Repository Location, then you will see the screen as following. (Figure 3.1)

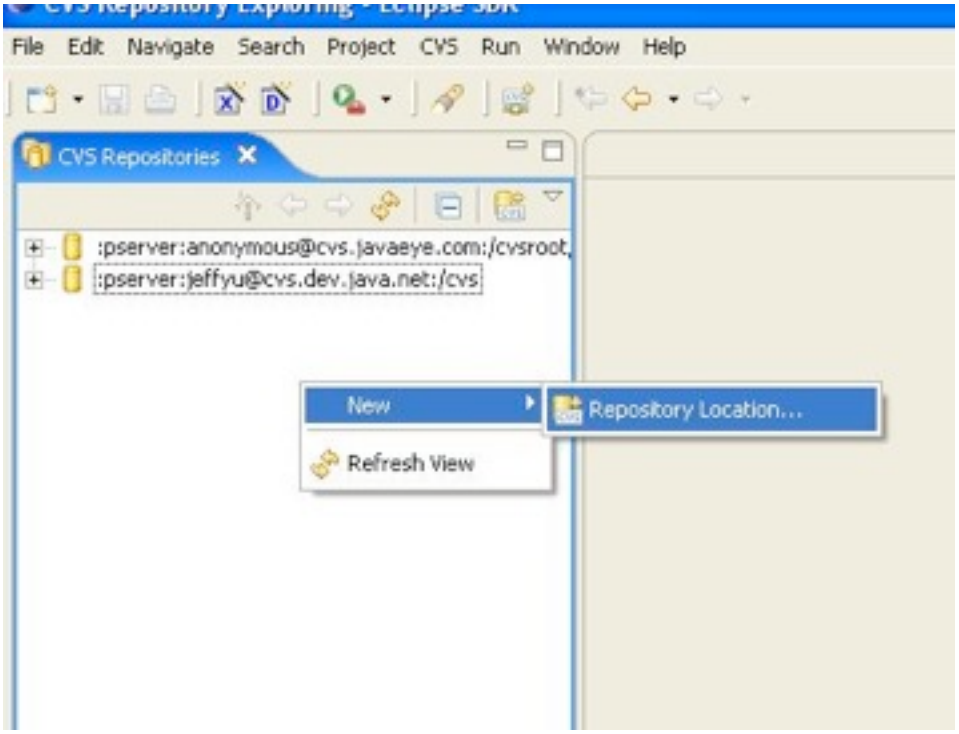

Figure3.1

3.Click it, it will forward to dialog box as following (Figure3.2), fill out the form, and click the finish button.

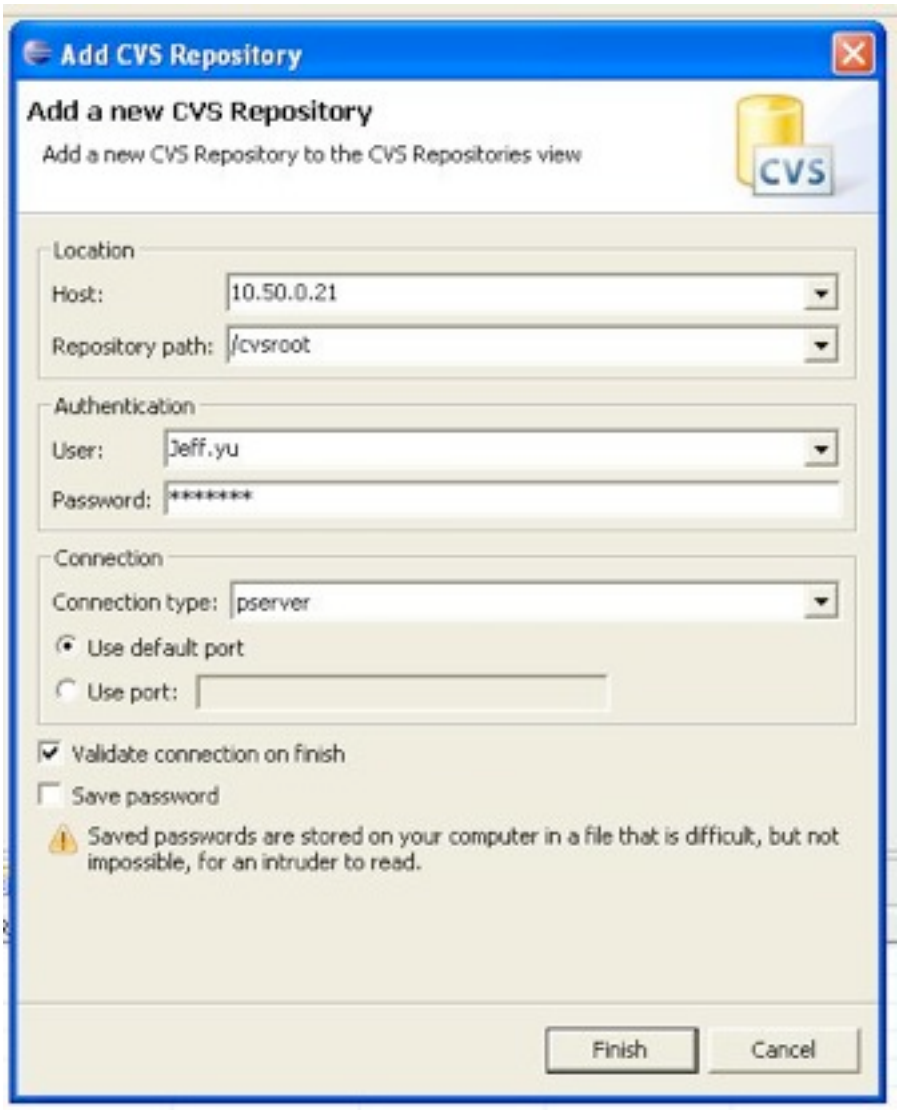

Figure 3.2

4. Once you establish a successful connection, you can see below screen (Figure 3.3). Click the plus (+) icon in front of HEAD to expand it.

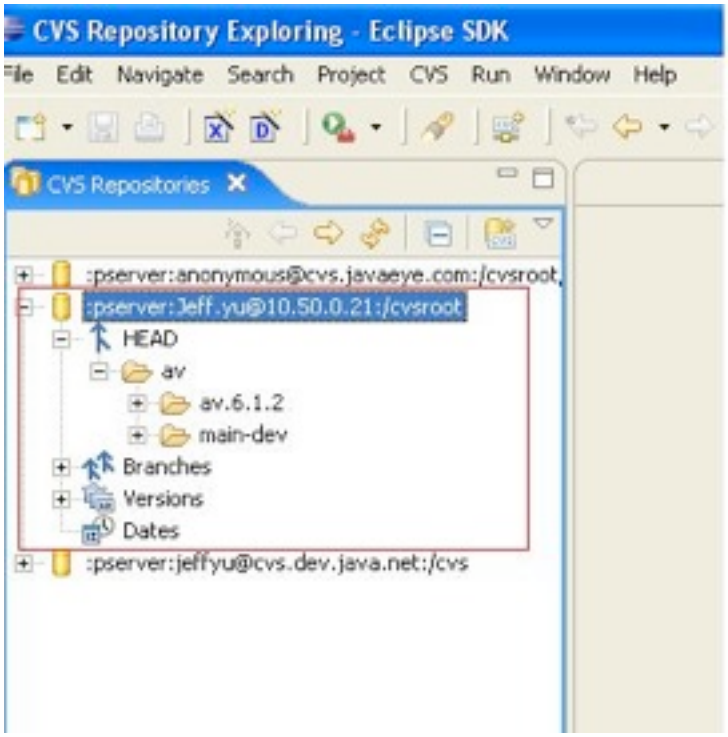

Figure 3.3

3.2. CHECK OUT A PROJECT

Ok, Once you have established the connection, what you need to do next is Check out the project you want.

1. Recall that we have established the connection, and what we see that is figure 3.3, we can see our project such as main-dev in the head section.
2. Right click the main-dev folder, it will pop up a menu as following (Figure3.4).
3. You can select checkout directly if you don't want to change your local workspace, but if you want to specify a folder to your this project, you need to choose the Check out As…

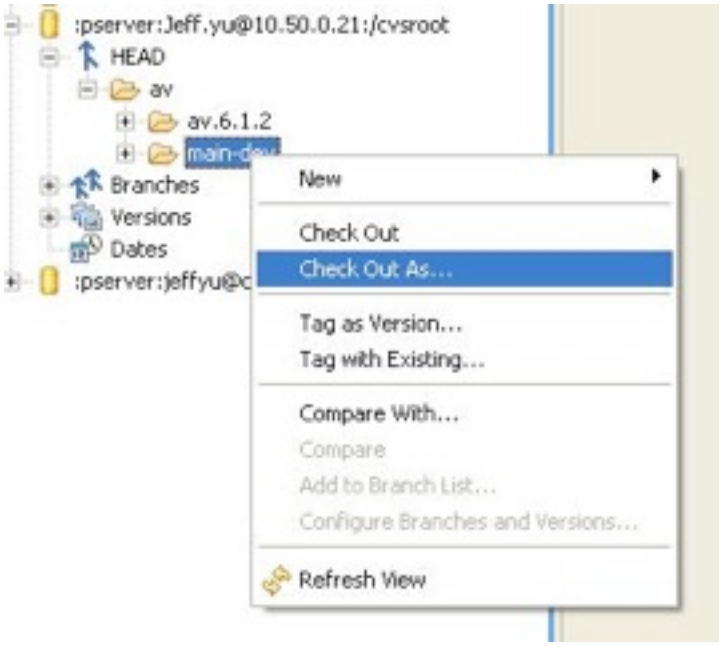

Figure 3.4

Following are some steps if you choose the Check Out as…. Choice.

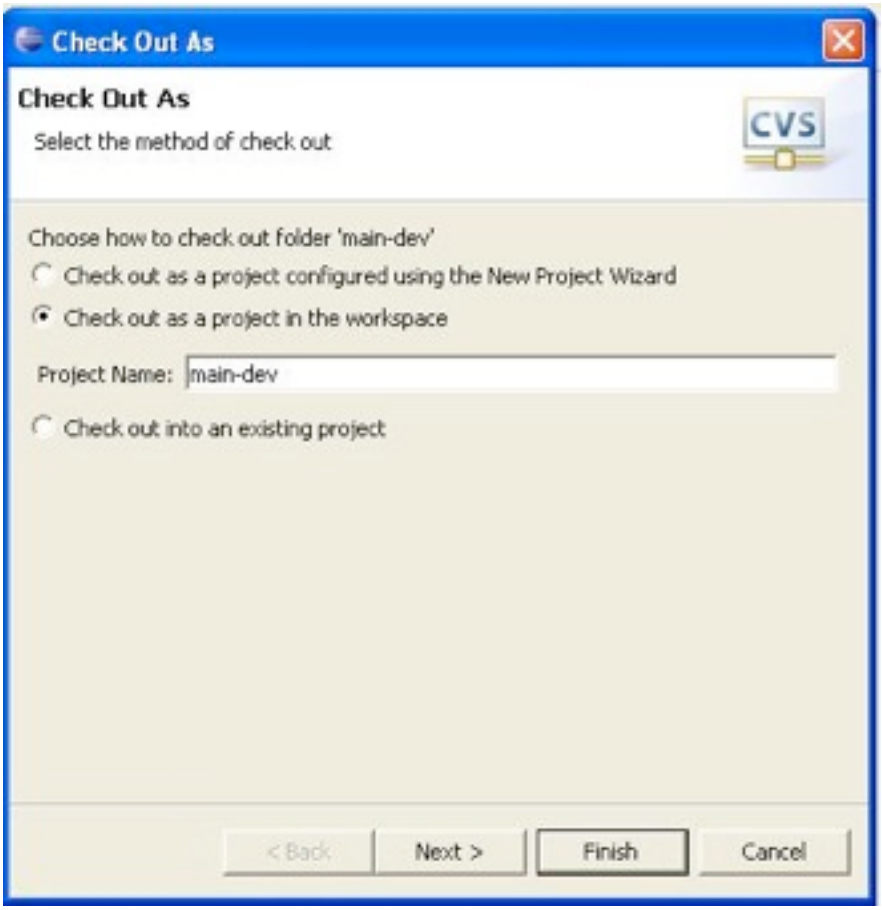

Figure 3.5

4. Click Next button, then it will forward to Figure 3.6 screen.

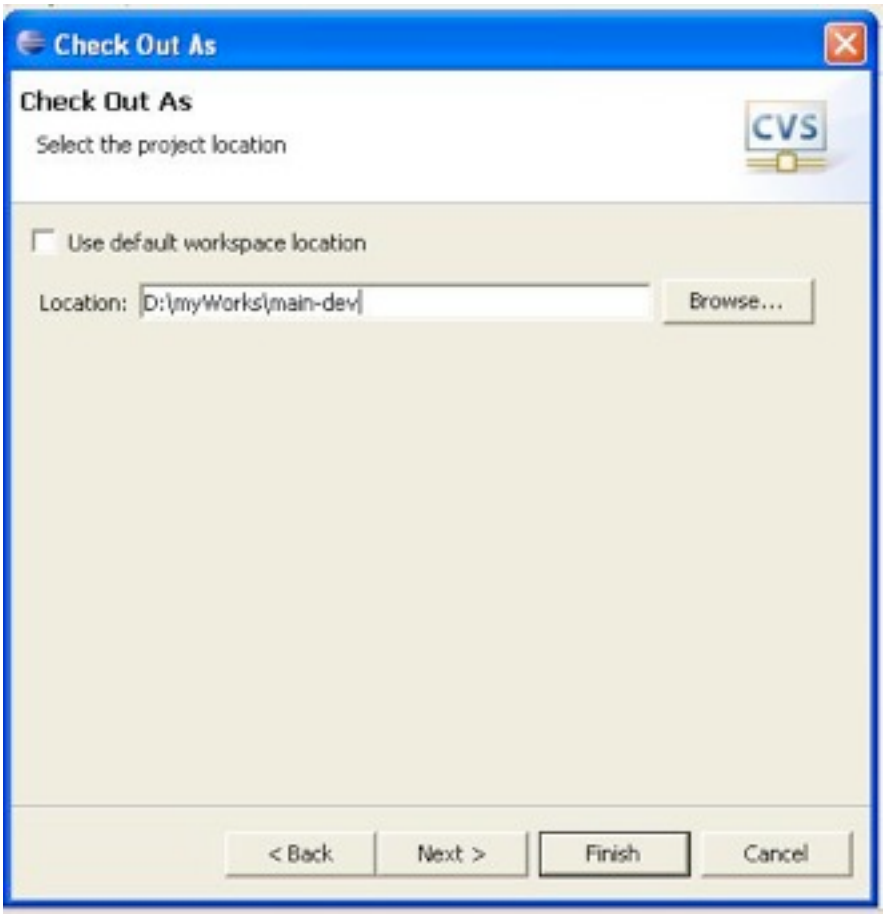

Figure 3.6

5. Click Finish button.
6. Once you have checked out the project successfully, switch to the Java Perspective in eclipse, you can see the project you have checked out as bellowing Figure 3.7.

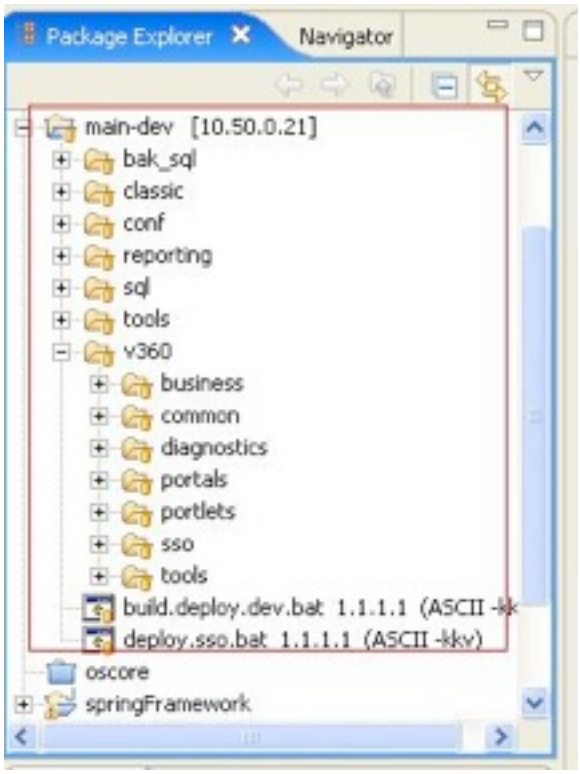

Figure 3.7

3.3. COMMON ACTIONS

Now, what we have done before is called preparatory actions prior to our topic. Since we just change from VSS to CVS, some guys may be not familiar to the CVS's pattern; here are some my points I want to say.

CVS and VSS are totally different in purpose and philosophy, while VSS provide an exclusive mode if you want to update a file, but CVS doesn't do it. CVS purpose is to provide a collaboration mode, the file cannot be check out exclusive, so there are a new concept called CONFLICT which doesn't exist in the VSS, we will discuss it in-depth in following section.

Firstly, we are going to see the first action, Synchronize Action, is sort of Files Compare action in VSS.

3.3.1. SYNCHRONIZE ACTION

Since we are all programmers, I want to explain it from examples, maybe it is a CVS HelloWorld.;-)

Say, now I want to change a java file called InspectionUtil.java, I open it and add the code what I want, then save it, See the following screen (Figure 3.8).

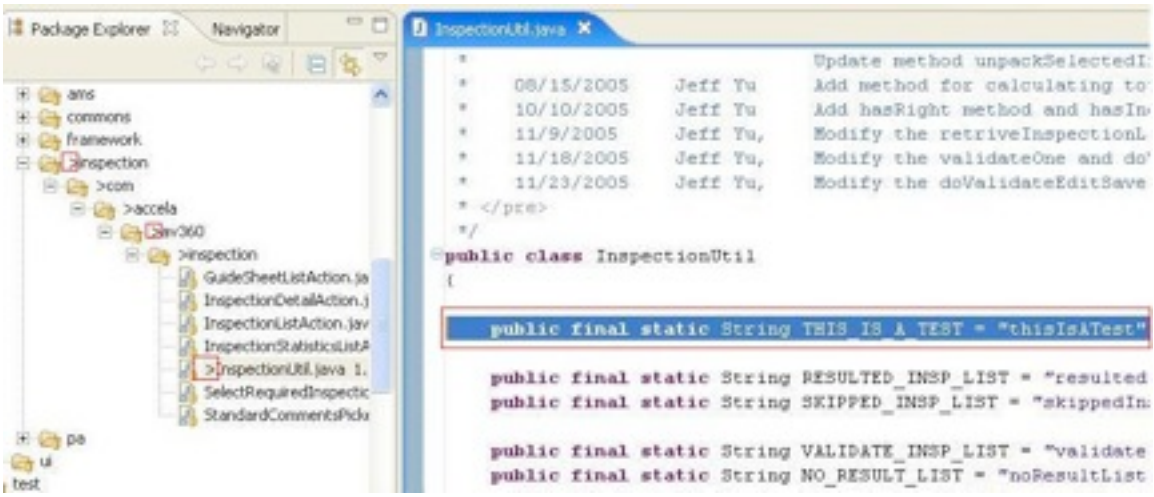

Figure 3.8

Tips: Do you notice that the Greater sign (>) in the left side, which highlight in red rectangle. What does it mean? It means this file is newer than the server.

So now I want to save my changes to the CVS server, but before our save, we do the Synchronize action to see what I change from the latest version.

Click the Inspection folder, right click the mouse. (We often synchronize the whole folder or whole project in case we forget to commit something we have changed.) See the figure 3.9.

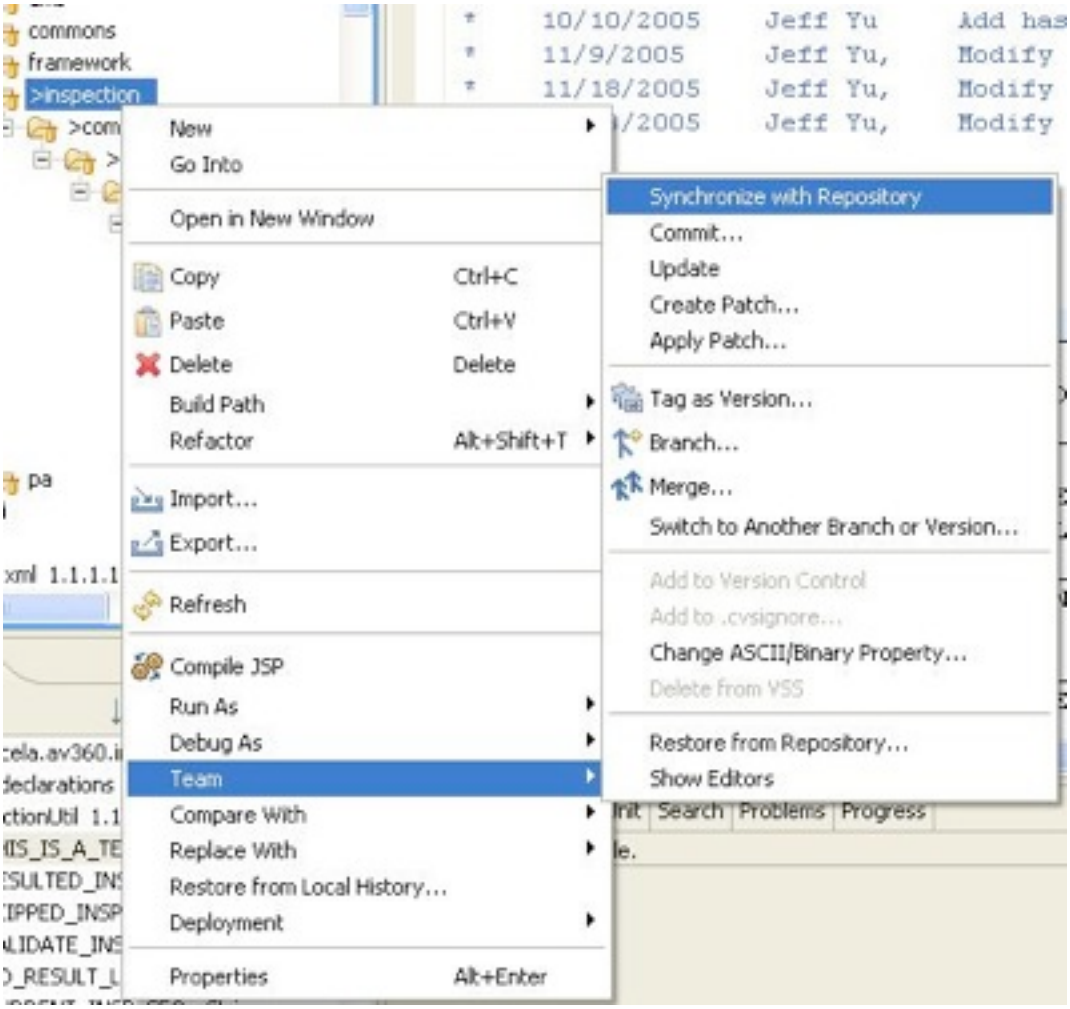

Figure 3.9

See the below Screen (Figure 3.10), there are two files we need to deal with. By the way, we also see some icons on the top of window.

Figure 3.10

Tips: Please notice the four different color rectangle in the figure 3.10, the red one rectangle's icon is called Incoming Mode, it indicates server's code is newer than your local's code; the second one is called Outgoing Mode, it indicates your local's code is newer than the server's; the third one is called Incoming/Outgoing Mode, it is the combination of the former two mode; the last one is called Conflicts Mode, it display the conflict files.

Tips: we also can see the red ellipse and blue ellipse; the forward (=>) sign is as same as outgoing meaning, while the backward (<=) sign is as same as incoming meaning.

As we see from the Figure 3.10, there are no conflicts in the InspectionUtil.java file, so what we need to do next is to COMMIT to server in order to save our modification.

3.3.2. *COMMIT ACTION*

Commit action is sort of check in action in the VSS, see the Figure 3.10, now we want to do the commit action, so it is safe for us to switch to the Outgoing Mode. See the Figure3.11

Figure 3.11

Now, you can just see the one file in this Mode, for the InspectionList.java is belonging to the Incoming Mode.

Click the main-dev, right click the mouse, see the Figure 3.12.

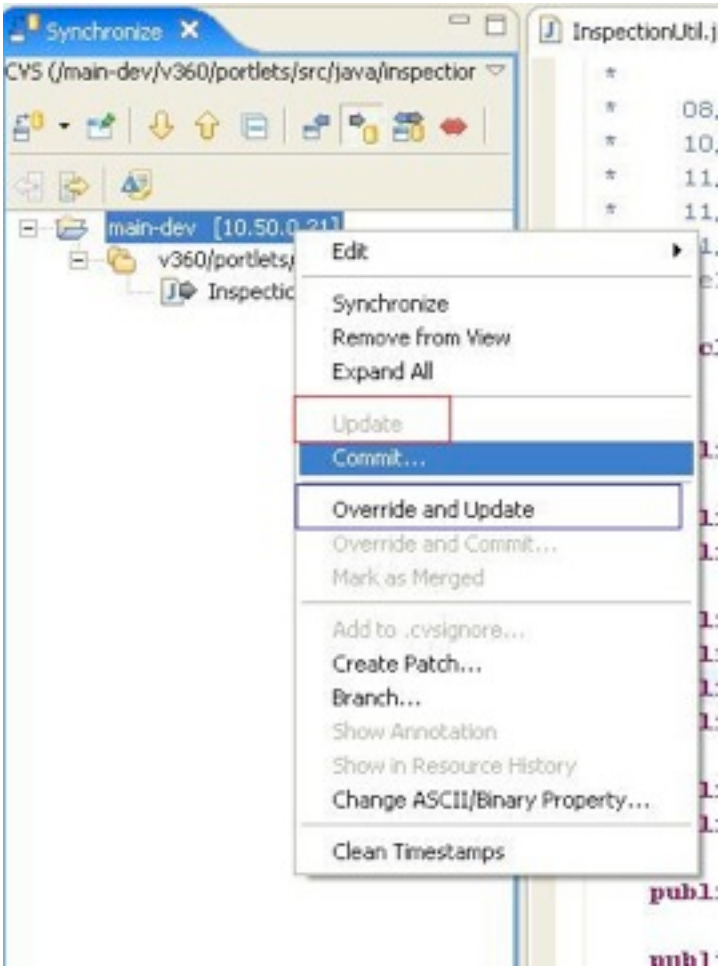

Figure3.12

Tips: Do you see the Update word in the red rectangle, why is gray? Because this file is belong to the outgoing mode. How clever this plugin is. :-)

Tips: Please notice the Override and Update words in the blue rectangle, what does this action do? It will override your local file, which is newer than server with the server's old file. So you need to do this action carefully, otherwise you will lose what you have modified. :), PS, any actions contain Override we should also do it carefully.

Ok, Click the Commit Action, it will forward to another screen as following.

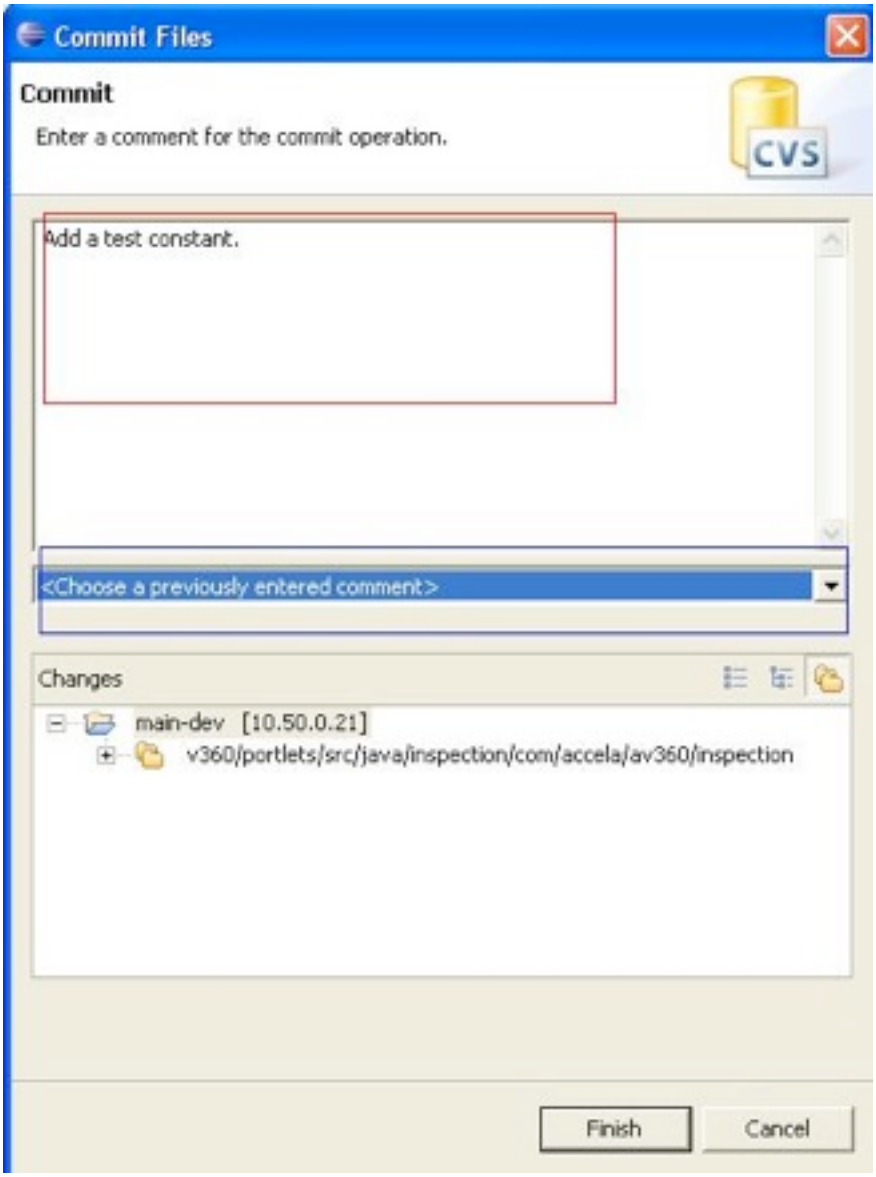

Figure 3.13

As a good habit, we need to do fill out the comment if we made some commit action, you can fill it in the red rectangle or choose previous entered comment from the dropdown which is in blue rectangle.

Click the Finish button.

And now the commit action is finished. :-)

3.3.3. UPDATE ACTION

Now, as we finish the Commit action, as recall from the Figure 3.10, we have 2 files to deal with, and now we just have committed one file, how about the other one. As we mentioned before, the InspectionListAction.file is belong to Incoming Mode, so we need to do is using Update Action to get it from the Server. This Update Action is sort of Get latest version in VSS.

Firstly, we change our mode to the Incoming Mode, we can see the following screen as Figure 3.14.

Figure 3.14

As same as Commit Action, we click the main-dev, and right click the mouse; it will pop up a menu as Figure 3.15.

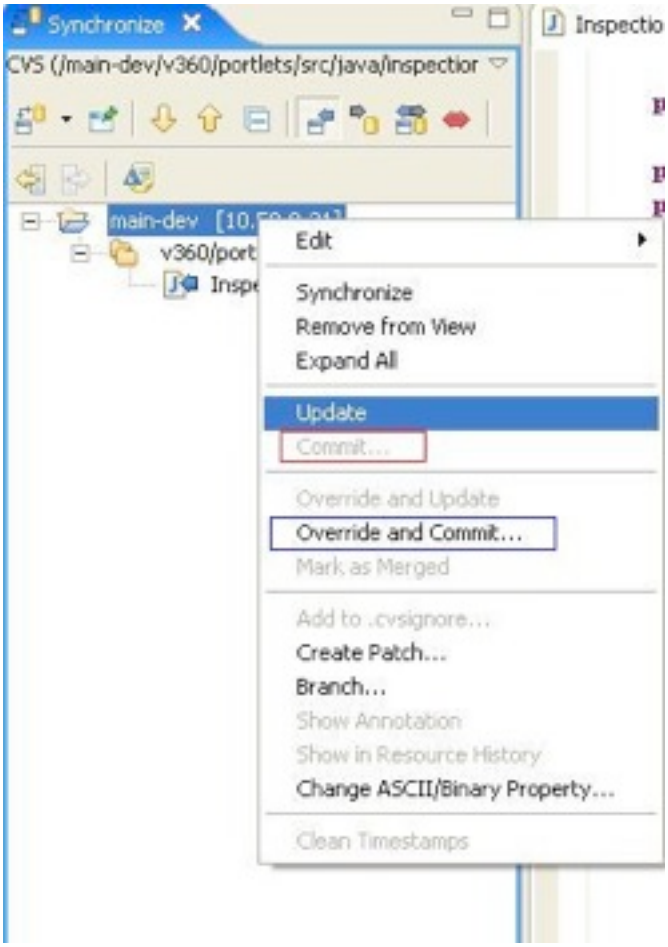

Figure 3.15

Tips: Because this file is belong to incoming mode, the commit action is gray, here I need to underscore the Override and Commit Action, as I said before, we need to do actions which contain Override very carefully, and what will this action do? It will override the server's latest code using your old local's file, so it means you will override other's code.

Click the Update; you will get the latest code from the Action.

At this moment, we have covered synchronize, update, commit actions, are they simple? Yup, is it any annoying thing in CVS usage for beginner? Yeah, it is conflict. And now we are going to see its concept and how to deal with it when we fact the conflict files.

3.4. CONFLICT

At first, we will try to figure out what the conflict is. Say, I and Anna both Update the InspectionUtil.java file, and we both modify this code at the same time, then Anna commit this file firstly, as then I have modified the codes and do the Synchronize Action to want to save the changes to the server. This time it will cause the problem, because I didn't get anna's modified file when I modified this file. And we called this problem as

Conflict.

We can see the conflict from the below screen (Figure 3.16)

Figure 3.16

Tips: This sign (⇔) represents conflict.

Hold on a second, at this moment, I want to tell how the CVS work from my view before we explore some conflict solutions.

3.4.1. EXPLORE HOW CVS WORK

We still use the InspectionUtil.java as our example, when anna and I update this file; its version is 1.0 for instance.

SN	Version Number	Action	Result
1	1.0	Anna Update the file	Successful
2	1.0	I update the file	Successful
3	1.1	Anna finish the modification and commit	Successful
4		I finished my modification and synchronize	Conflict

God, how does CVS know it? It is due to the version number, when I want to commit my modified file, my local file version is 1.0, but in the CVS server, this file's version number is 1.1 now, So the server know that someone else have made the modification before me, so the conflict occurs…

Now that we know what is the Conflict and how does it happen, we need to know how to deal with it next, right? ;-).

As general, we can categorize to 2 scenarios.

3.4.2. *CONFLICT SCENARIO 1*

Recall the Figure 3.16, we can see there is a conflict file called InspectionUtil.java, now double click this conflict file, you can see the following screen as Figure 3.17

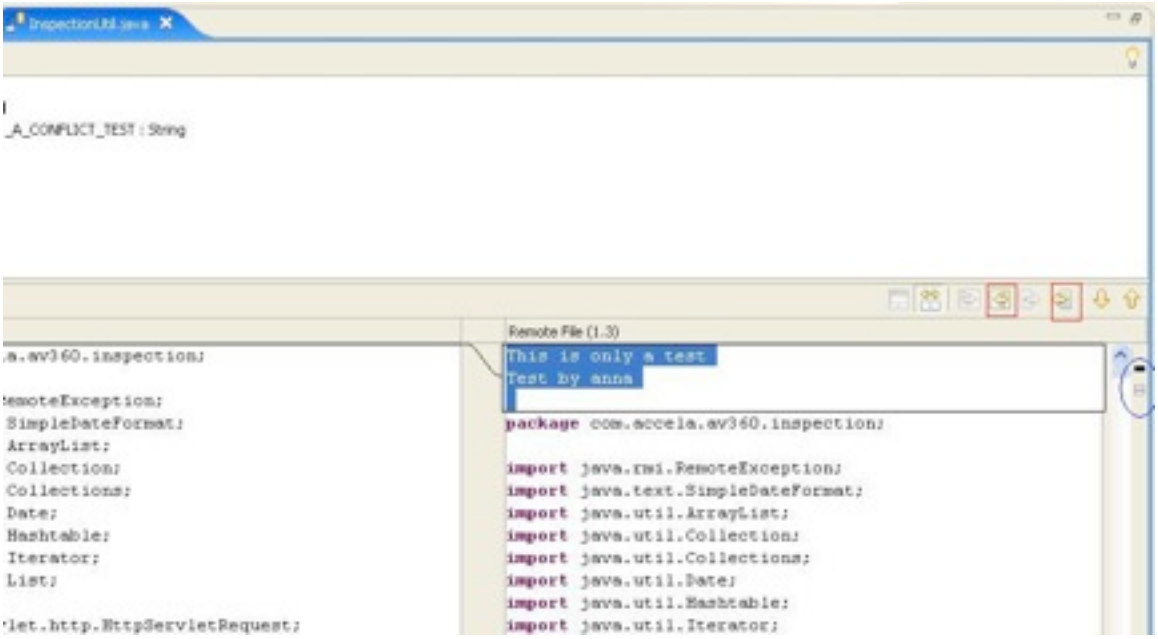

Figure 3.17

From this screen, you can see there are some buttons on the right side, which I highlight with red rectangle. If you have used the BeyondCompare software, I think you will know it easily. The first highlight one is called "Copy all non-conflicting changes from right to left", the second is called "Copy current Change from right to left". I think you guys all know theirs function from their name.

From above screen, we can see the difference between two files, and you also can know how many difference counts from the right side which I highlight it with blue ellipse, in this example, we can see that there are 3 differences between them and you also can see the file modification which was modified by Anna.

Now, what we are going to do is merge the difference.

As we have move the Anna's modification to my file, then my local file will include all the modifications which was modified by Anna. PS: merge is a very careful work, anyway.

Then, we close this comparing file, back to the Figure 3.16, Select the InspectionUtil.java and right click the mouse.

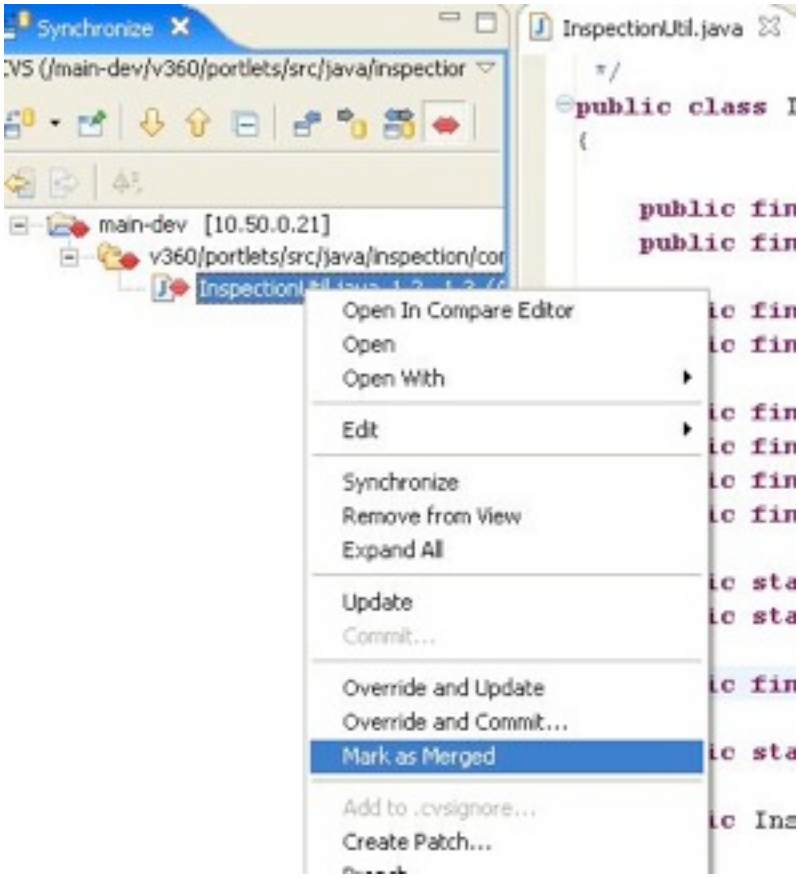

Figure 3.18

Click the Mark as Merged, then this file will not be conflicted any more, what will it be??? It will be belong to the Outgoing Mode, because CVS Server thinks you guy have done the merge, so your file now is the latest version.

Then do the commit action as we described it before.

Then work done.

3.4.3. *CONFLICT SCENARIO 2*

Same file – InsepctionUtil.java , if anna and I both Change the same code, say we both change the

```
public final static String THIS_IS_A_TEST =
"thisIsATest90000000";
```

code, now when I synchronized the file, what will happened. See the Figure 3.19 as following.

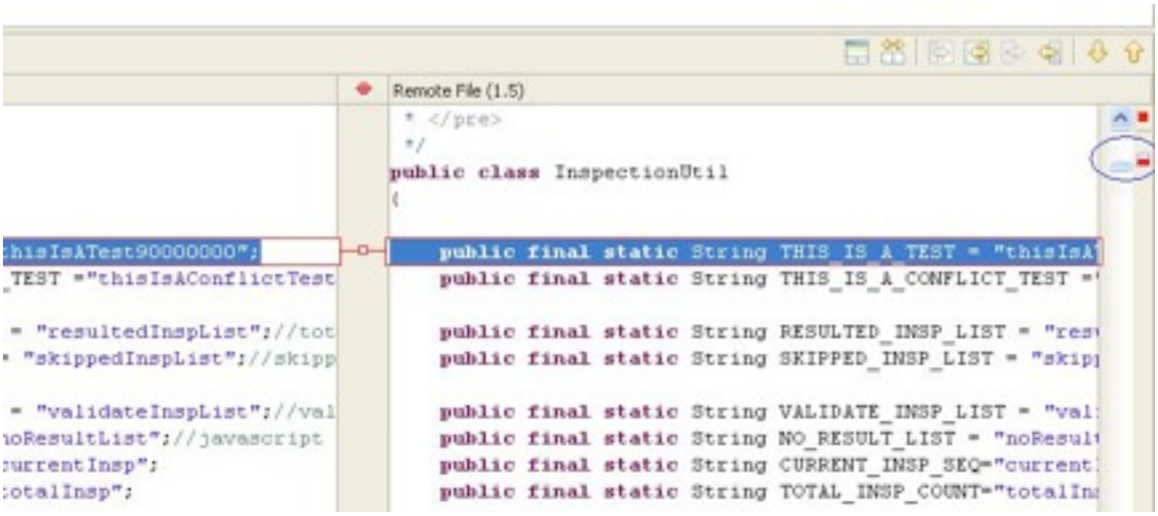

Figure 3.19

The red rectangle represents this line has conflicts, this situation is belonging to the worst situation, at this moment, you had better to negotiate with people who made conflict with you, and then if you are right, you can use the Override and Commit Action to do commit forcedly.;)

4. Best Practice

- There is a sentence called. "Update frequently, Commit carefully."
- When you want to do Commit action, please remember do the synchronized action firstly.

5. FAQ (Frequently Asked Question)

Q: If I made modification in my local code, and then I command the Update Action, will it override my local code?

A: Nope, it will not override the local code.

6. Reference