CS 580 Client-Server Programming

Spring Semester, 2005

Doc 3 CVS & Store

Contents

CVS 3
TortoiseCVS 5
CVS & Eclipse 6
Example of Using CVS 7
One Time Only 7
Starting a New Project 9
Creating a shortcut or Module 10
Making Changes 15
Deleting the local Copy 18
Accessing CVS Files on Rohan from Off-Campus 21
Using Store 24
Loading Store into an Image 24
Packages, Parcels & Categories 25
Using Packages 29
Browser Level Source Control 33
Connecting to the CS580 Store Repository 35

Copyright ©, All rights reserved. 2005 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

References

http://www.cvshome.org/

http://www.tortoisecvs.org/

http://www-rohan.sdsu.edu/~stremler/CS530/AS1/remote_cvs.html
Stremler's remote CVS on Rohan page

UCB InfoSys 255 course information on CVS & Eclipse
http://www.sims.berkeley.edu/academics/courses/is255/f04/labs/lab090904/CVSHowTo/eclipseCvsHowTo.html

CVS Eclipse Plug-in FAQ
http://dev.eclipse.org/viewcvs/index.cgi/%7Echeckout%7E/platform-vcm-home/docs/online/cvs_features2.0/cvs-faq.html

Source Code Management Guide, doc/SourceCodeMgmtGuide.pdf in VW 7.x installation

CVS

Concurrent Versions System

Allows multiple users to work on code

Allows access from multiple machines

Manual http://www.cvshome.org/docs/manual/

Windows Version

http://www.wincvs.org/ client

http://www.cvsnt.org/ server

http://www.tortoisecvs.org/ GUI client and server

Rohan

On rohan need /usr/local/bin in your path

This seems to be in the default path

CVS Books

Manual http://www.cvshome.org/docs/manual/

There are a number of books on CVS

Pragmatic Version Control by Thomas & Hunt has been endorsed by at least one student

http://www.pragmaticprogrammer.com/starter_kit/vc/

TortoiseCVS

Nice CVS client and Server for Windows

Free

Adds CVS menu options in Windows Explorer

Can use Rohan as remote server

http://www.tortoisecvs.org/

CVS & Eclipse

Eclipse has a GUI interface for accessing CVS repositories

Can access local & remote CVS repositories

See
- CVS Eclipse Plugin- FAQ
- UCB InfoSys 255 course information on CVS & Eclipse

Must create the CSV repository root first

Example of Using CVS

All examples assume you are using Unix

One Time Only

To store your own files you need a cvs root (repository)

This is created once

Command

```
cvs -d cvsLocation init
```

Example

```
cvs -d /home/ma/whitney/cvsRoot init
```

Where is the Repository?

cvs command need to know the location of the repository

You can
- Add this path to each command or
- Set the CVSROOT environment variable

Example of setting Environment variable

For tcsh or csh in .cshrc or .login file add a line

```
setenv CVSROOT  /home/ma/whitney/cvsRoot
```

For other shells you may need to use:

```
set CVSROOT='/home/ma/whitney/cvsRoot'
```

Starting a New Project
Create the directory structure for the project

Example

```
rohan-> mkdir xmlrpcClient
rohan-> cd xmlrpcClient/
rohan-> cvs import -m "start assn1" cs580/xmlrpcClient yoyo
start

No conflicts created by this import
```

Import Command

- -m
  Initial message for the project

- cs580/xmlrpcClient
  where the files are stored in the repository

- yoyo
  means nothing, can be anything but must be here

- start
  this is the start of the project

After this command the directory is stored in the repository

Creating a shortcut or Module

Remembering cs580/xmlrpcClient is going to be hard so create a short cut or module

Example

```
rohan-> cd ..

rohan-> cvs checkout CVSROOT/modules
rohan-> cd CVSROOT/
rohan-> ls
~/CVSROOT
CVS/      modules
rohan 40-> vi modules
```

Add the followng line at the end the file

```
assn1 cs580/xmlrpcClient
```

Now to commit the changes

```
rohan-> cvs commit -m "Added assn1 module" modules
Checking in modules;
/home/ma/whitney/cvsRoot/CVSROOT/modules,v  <--
modules
new revision: 1.6; previous revision: 1.5
done
cvs commit: Rebuilding administrative file database
```

cvs commit updates the repository with all changes in the current project

Now to remove the module files

```
rohan-> cd ..
rohan-> cvs release -d CVSROOT/
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory
`CVSROOT/': y
```

cvs release –d directoryName

- Tells cvs we are done with the local copy of this project
- Deletes the local files
- Directory and files are still in cvs main repository
- You can check them out again

Using our assn1 Project
Getting the Files

```
rohan-> cvs checkout assn1
rohan-> cd assn1
rohan-> ls
CVS/
```

cvs checkout moduleName

Makes a local copy of the files in the give module

Adding a new File

Create a file called client.java using an editor

```
rohan-> cvs add -m "main client file" client.java

rohan-> cvs commit -m "added main"
cvs commit: Examining .
RCS file:
/home/ma/whitney/cvsRoot/cs580/xmlrpcClient/client.java,v
done
Checking in client.java;
/home/ma/whitney/cvsRoot/cs580/xmlrpcClient/client.java,v  <-
-  client.java
initial revision: 1.1
done
```

cvs add
- Tells cvs about the file

cvs commit
- Commits all changes in the current project to repository
- Can add file name to commit only a single file

Making Changes

We can now make more changes to the file

At anytime we can commit the files

```
rohan-> cvs commit -m "added foobar method"
cvs commit: Examining .
Checking in client.java;
/home/ma/whitney/cvsRoot/cs580/xmlrpcClient/client.java,v  <-
-  client.java
new revision: 1.2; previous revision: 1.1
done
```

Getting the log of Changes

We can view the log of changes made to the project

```
rohan-> cvs log
cvs log: Logging .

RCS file:
/home/ma/whitney/cvsRoot/cs580/xmlrpcClient/client.java,v
Working file: client.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2;     selected revisions: 2
description:
main client file
----------------------------
revision 1.2
date: 2002/09/12 22:45:17;  author: whitney;  state: Exp;  lines:
+2 -0
added foobar method
----------------------------
revision 1.1
date: 2002/09/12 22:44:16;  author: whitney;  state: Exp;
added main
```

Comparing Files

```
rohan-> cvs diff -r 1.1 -r 1.2 client.java
Index: client.java
===================================================================
RCS file:
/home/ma/whitney/cvsRoot/cs580/xmlrpcClient/client.java,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
1a2,3
> // main
> // more code
```

The diff command shows the difference between versions a file

Deleting the local Copy

We can even delete the local copy of the files

This tends to make novices nervous

```
rohan-> cd ..
rohan-> cvs release -d assn1/
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory `assn1/':
y
```

What is the Point?

So far we have done a lot of work without any benefit

We can get back previous versions!

The -D flag for checkout

We can tell cvs to give us the lasted version of a project before a given time

```
rohan-> cvs checkout -D yesterday assn1
rohan-> cvs checkout -D "2002-9-11 20:00" assn1
rohan-> cvs checkout -D "2002-9-11" assn1
rohan-> cvs checkout -d "1 hour ago" assn1
```

The –r flag

We can checkout by version number

```
rohan-> cvs checkout -r 1.1 assn1
```

More Features

cvs has more features

Read the manual to find out about

- Abbreviations for commands
- Branching
- Merging

Accessing CVS Files on Rohan from Off-Campus

Command line Access – CVS over SSH

For complete instructions see the page by Stewart Stremler:
http://www-rohan.sdsu.edu/~stremler/CS530/AS1/remote_cvs.html

First set CVS_RSH environment variable

It points to the location of cvs

On my machine I do this with:

```
setenv CVS_RSH /usr/bin/ssh
```

Note the path
- Must end in the ssh executable
- May be different on your machine

Second set the CVSROOT environment variable

The general format is:

```
:ext:username@servermachine:/absolutePathToCVSRoot
```

For example I use:

```
setenv CVSROOT
:ext:whitney@rohan.sdsu.edu:/home/ma/whitney/cs580
```

Remote Access to Rohan CVS using TortoiseCVS

After you have installed TortoiseCVS on a windows machine, when you right click on an Windows Explorer window the popup menu will contain a CVS items. If you select the Checkout item you will get the TortoiseCVS Checkout Module window. Enter the following:

For the Protocol select “Secure shell(:ext)
For the server enter the name of the machine with the CVS root. For rohan enter rohan.sdsu.edu

For the Repository folder enter the absolute path to the directory containing your CVS root on the server machine

For the User name enter your user name on the server machine.

If your CVS root is set up on the server machine this will allow you to check out files from the server machine. Adding and updating files is also easy to do. One problem is that you will be asked your password each time you make a request of the server. To avoid this read the TortoiseCVS FAQ on how to avoid entering your password every request with SSH at:

http://www.tortoisecvs.org/faq.html#sshkeys

For more information on using TortoiseCVS read the tutorial that comes with TortoiseCVS download.

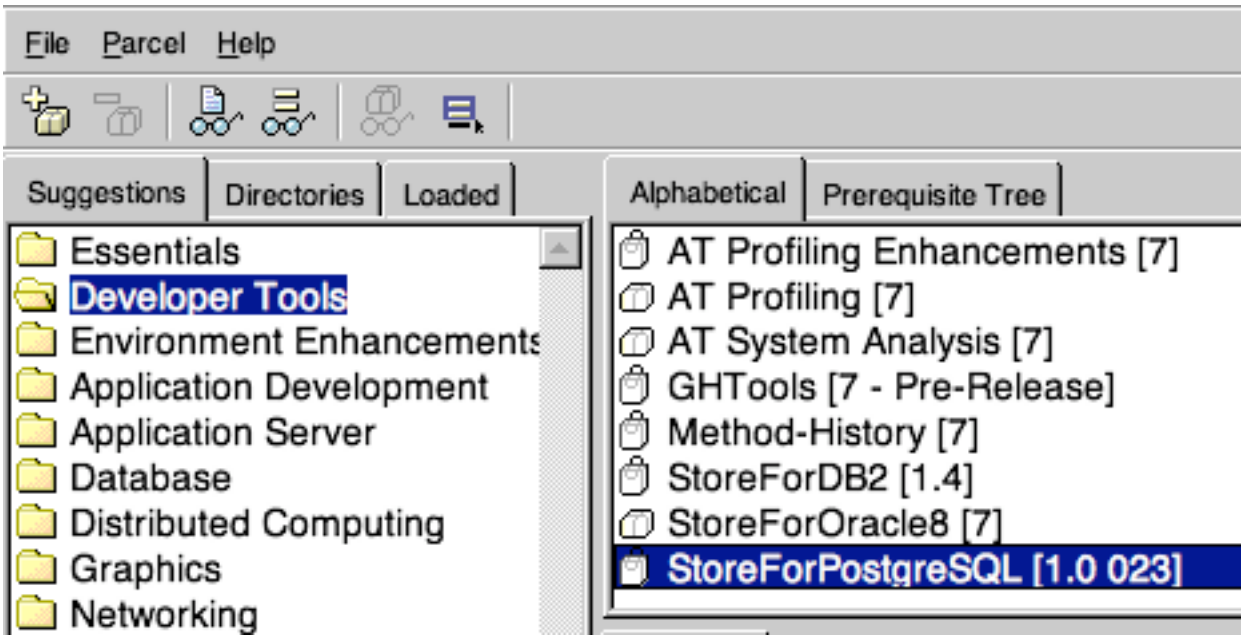
Using Store
Loading Store into an Image

In VW 7.3 Store is already loaded so this step is not needed.

Load the StoreForPostgreSQL parcel

Use the Parcel manager

- In the launcher in the System menu select “Parcel Manager”
- In the Parcel Manager Window select Developer Tools
- Scroll the upper right window to find StoreForPostgreSQL
- Select StoreForPostgreSQL



- Click on the load Parcel icon.

The “Load Parcel” icon is the left more icon on the icon bar.

Packages, Parcels & Categories

Open a System browser

In Launcher in Browse menu select “System Browser”

Or select the “System Browsers” icon – the third icon from the left in the Launcher

System browser now supports three ways to organized code:

- Categories
- Package
- Parcels

In the System browser use the “Browser” menu to switch between viewing the code via these methods

Category

A collection of classes

Use to view any class in the system

Package

A package can contain:
- A collection of classes
- Methods added to classes elsewhere in system
- Prerequisite requirements
- Actions to perform when loaded

Development time grouping of related code

Packages can be stored in a repository

Can be grouped into bundles for large projects

Parcel

File based version of a Package

Runtime grouping of related code

Contains binary for fast loading

Can be created without Smalltalk source if you wish to hide source from users

Using Packages
Creating a Package

Open a System browser and set it to view packages

In the System Browser “Browser” menu select “Package” item

In the System Browser “Package” menu select “New” and the submenu item “Package”

You will be prompted for a package name

For this document I will use the name “StoreLectureExamples”

With your new package selected in the System Browser “Package” menu select “Make Current”

This will cause any change you make anywhere in the system to be part of your package when you are dealing with Categories

Now you can set the browser to show categories until you are ready to save your code to the repository

Adding Code to a Package

Assumptions
- You can created a package
- You have made the package the current package
- You are viewing code in Category mode
- You have a System browser open

We will
- Make a change to an existing class
- Add our own class

Changing an Existing Class

Go to the Integer class

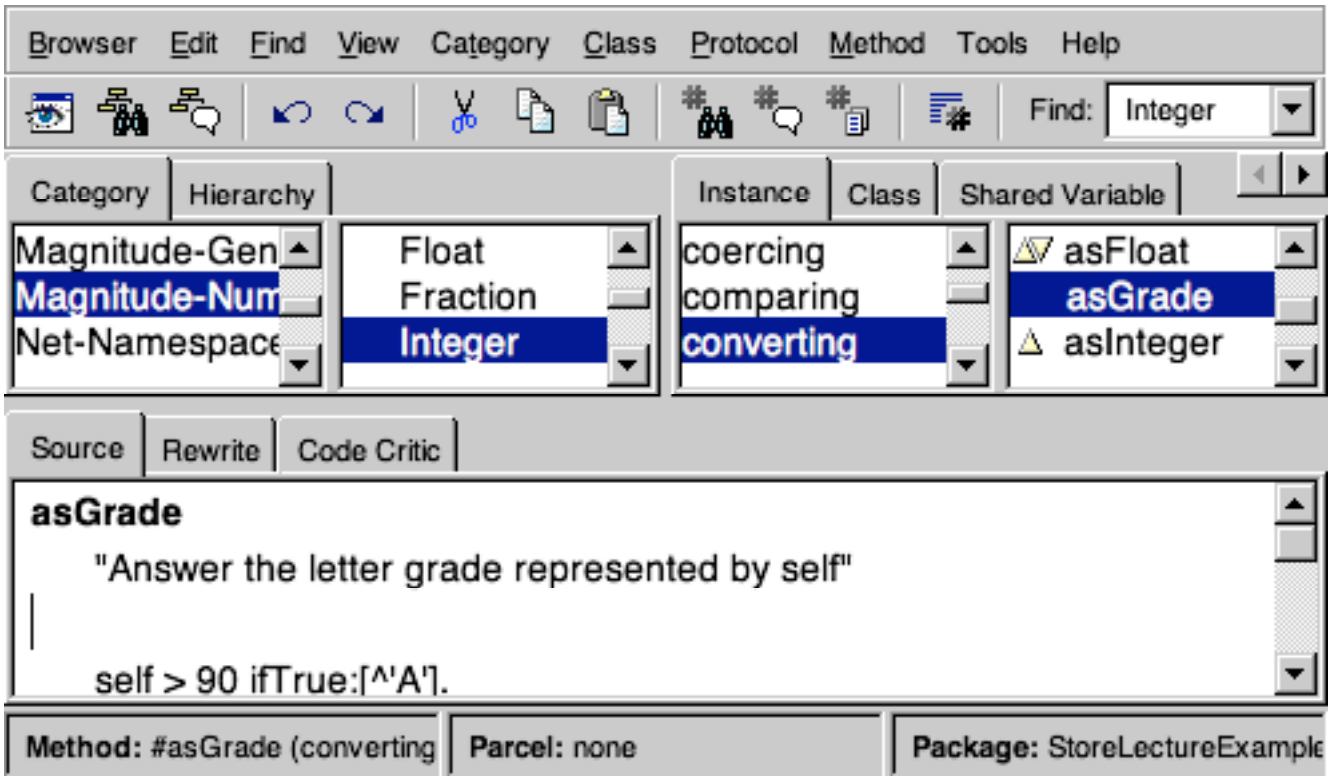
Type “Integer” in the find box and hit return

Add the asGrade method to the converting category

```
asGrade
    "Answer the letter grade represented by self"

    self > 90 ifTrue:[^'A'].
    self > 80 ifTrue:[^'B'].
    self > 70 ifTrue:[^'C'].
    self > 60 ifTrue:[^'D'].
    ^'F'
```

In the bottom of the browser you should see that method is in your package. See below.


Browser Edit Find View Category Class Protocol Method Tools Help
Find: Integer
Category Hierarchy
Instance Class Shared Variable
Magnitude-Gen
Magnitude-Num
Net-Namespace
Float
Fraction
Integer
coercing
comparing
converting
asFloat
asGrade
asInteger
Source Rewrite Code Critic
asGrade
"Answer the letter grade represented by self"
self > 90 ifTrue:[^'A'].
Method: #asGrade (converting
Parcel: none
Package: StoreLectureExample

Browser Level Source Control

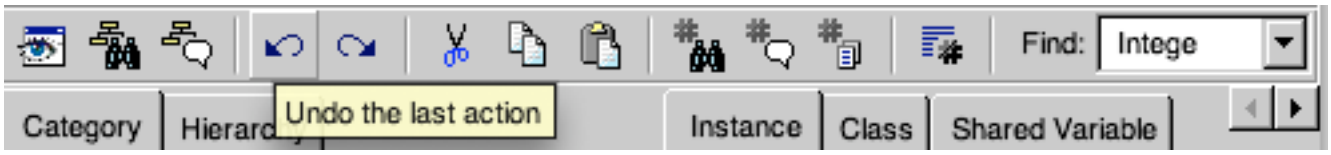
The System browser provides some source control without dealing with Store

To see this functionality
- Modify the asGrade method changing 90 to 95
- Save the method
- Change 80 to 85 and save the method
- Change 70 to 75 and save the method

Now click on the “Undo” icon to undo the last change

You can rollback all the changes to asGrade with the undo icon

Use the redo icon to redo each change



To see the history of all changes to asGrade

While asGrade is selected, in the browser “Method” menu select “Browse Versions”

Creating a New Class

Create a new category for your new class

In the Browser “Category” menu select the “New” menu

Then create a new class

For the notes & class demo I will use the following class.

```
Smalltalk defineClass: #BankAccount
    superclass: #{Core.Object}
    indexedType: #none
    private: false
    instanceVariableNames: 'balance '
    classInstanceVariableNames: ''
    imports: ''
    category: 'Store-LectureNotes'

BankAccount class methodsFor: 'instance creation'

new
    ^super new initialize

BankAccount methodsFor: 'accessing'

balance
    ^balance

deposit: aNumber
    balance := balance + aNumber

withdrawl: aNumber
    balance := balance - aNumber

initialize
    balance :=0
```

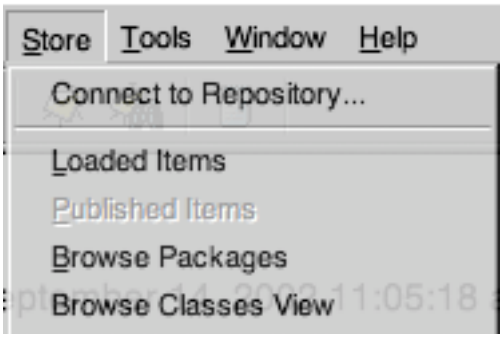
Now to save our package in the repository

Connecting to the CS580 Store Repository

Before saving a package we must connect to the repository

To connect to the CS580 repository on rugby.sdsu.edu your machine needs a connection to the Internet

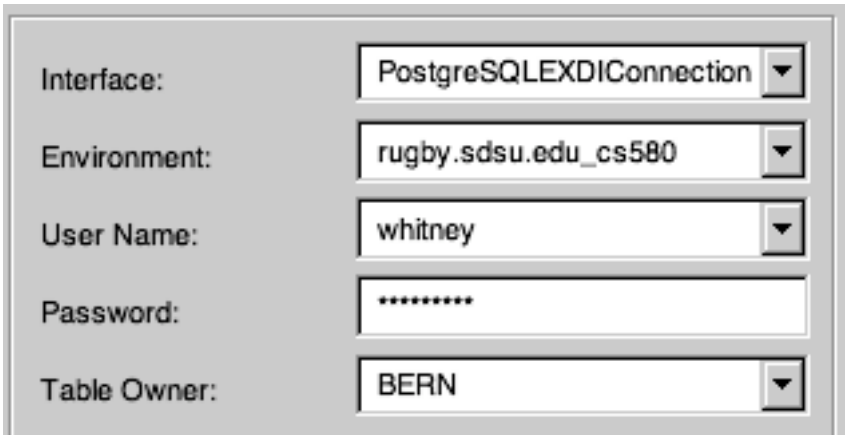
In the Launcher Store menu select the “Connect to Repository” item



First Time Connection

In the “Connect to Database” window:

For Interface select PostgreSQLEXDIConnection
For Environment type rugby.sdsu.edu_cs580
Contact me for a username and password
Keep BERN as owner of the table.



If you click on the “Save...” button you can give the settings a name to make it easier to connect later.

Now click on the “OK” button to connect to the repository

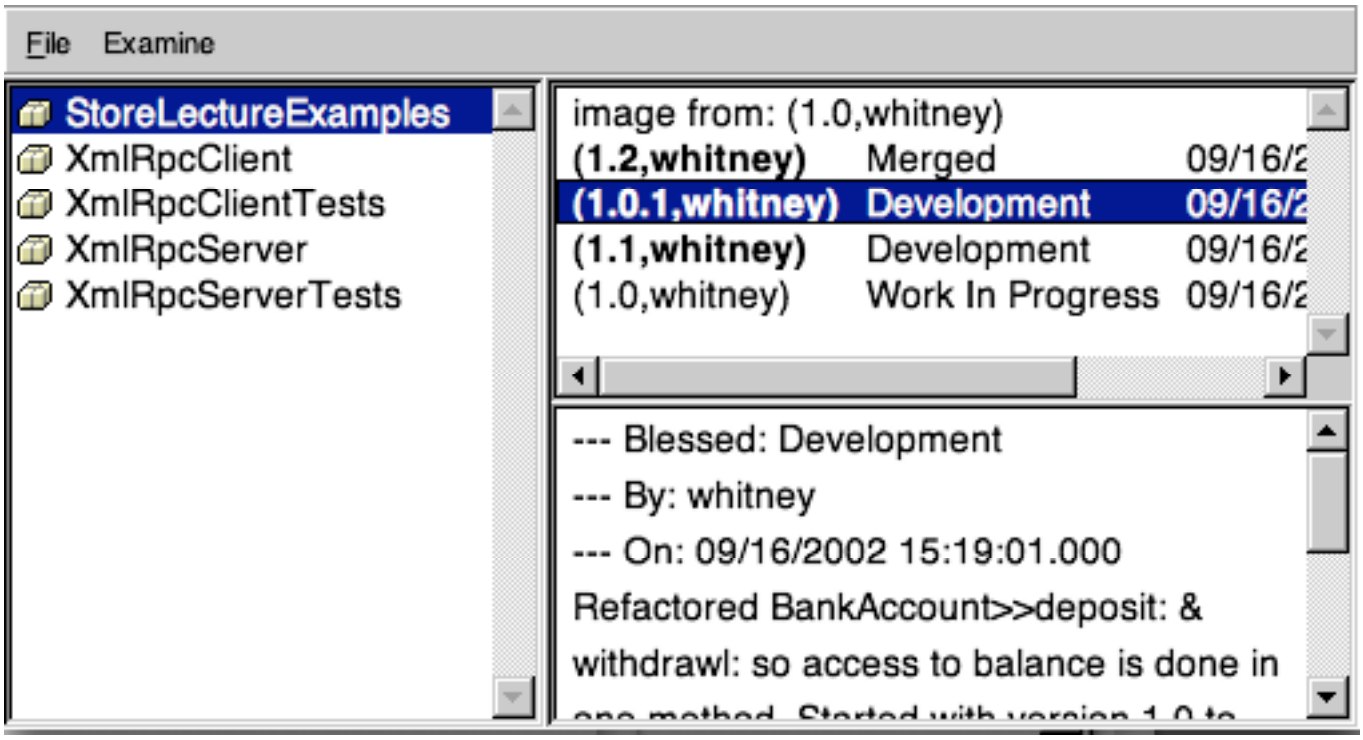
If all went well you are now connected to the repository. If you do not get any errors you will be connected.

Viewing Existing Items in the Repository

Once you are connected to a repository you can view its contents

In the Launcher's "Store" menu select "Published Items"

The resulting window will show you the contents of the repository. When you select a package you will see the different versions of the package. When you select version you see the comment for that version. When a version is selected you can either right click in the upper right window or use the File menu to load that version into your image.



Putting a Package into the Repository

To put a package into a repository

- Open a System browser if one is not currently open
- Use the Package view (Browser menu item “Package”)
- Select the package you wish to publish
- In the browser “Package” menu select the item “Publish...”
- Enter your commit comment in the “Publishing package” window
- Click on the publish button

There are a number of other useful things one can do like merge different versions of a package. However, you should know enough to read the manual to find out how to do that.