JUnit Web site: http://www.junit.org/


Brian Marick’s Testing Web Site: http://www.exampler.com/testing-com/

Testing for Programmers, Brian Marick, Available at: http://www.exampler.com/testing-com/writings.html

Main Mercurial Website, http://mercurial.selenic.com/

Source Control
Test your Processes
Why Use Source Control?
# Common Free Source Control Systems

<table>
<thead>
<tr>
<th>CVS</th>
<th>Subversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Versions System</td>
<td>Claims to be a better CVS</td>
</tr>
<tr>
<td>Command line interface in Unix</td>
<td>Many commands are same as CVS</td>
</tr>
<tr>
<td>Various interfaces in Window</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Git</th>
<th>Mercurial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by Linus Torvald</td>
<td>Python based</td>
</tr>
<tr>
<td>Distributed Version control</td>
<td>Distributed version control</td>
</tr>
</tbody>
</table>
# Mercurial


<table>
<thead>
<tr>
<th>Runs on:</th>
<th>Command line interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OS X</td>
<td>GUI interface for windows</td>
</tr>
<tr>
<td>Unix</td>
<td>Eclipse plugins</td>
</tr>
<tr>
<td>Linux</td>
<td>Can use locally with no server</td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
</tbody>
</table>
Simple Workflow

Start repository

Commit changes

Commit changes (creates changeset)

Commit changes (creates changeset)

Commit changes

Commit changes

code

code

code

Start repository
Changesets

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Changeset identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeset: 4 : 2278160e78d4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option text</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag:</td>
<td>tip</td>
</tr>
<tr>
<td>user:</td>
<td>Bryan O'Sullivan <a href="mailto:bos@serpentine.com">bos@serpentine.com</a></td>
</tr>
<tr>
<td>date:</td>
<td>Sat Aug 16 22:16:53 2008 +0200</td>
</tr>
<tr>
<td>summary:</td>
<td>Trim comments.</td>
</tr>
</tbody>
</table>
Branch in one Repository

Start repository

Commit changes

Commit changes

Merge

Commit changes

code

code

code

Start repository
Standard Mercurial Workflow

Start repository

Commit changes

code

Start repository
Standard Mercurial Workflow

- Start repository
- Commit changes
- hg clone
- code
- Start repository
- Commit changes
- code
Standard Mercurial Workflow

Commit changes

code

Start repository

Commit changes

code

Start repository
Standard Mercurial Workflow

Commit changes

Start repository

code

Commit changes

code

Commit changes

code

Commit changes

code

Start repository

hg pull

hg update
Basic Source Control Operations

Starting a new project
Adding code to a project
Modifying existing code
Retrieving past versions of code
Handling conflicts in code
Creating code branches
Merging code branches

Cloning repositories
Pulling repositories
Pushing repositories
Mercurial Commands
Creating a Repository

`hg init myproject`

In Eclipse right click on a project and select the "Share Project..." item in the "Team" menu
Repository & Working Directory

Repository
In .hg directory
Contains
  history of changes
  list of files part of project

Working Directory
Contains
  Project files
  Project directories
  .hg
Adding Files to Repository

Must tell Repository which files to maintain

hg add
   Adds all files in the current directory

hg add filename
   Adds the named file

Just adds files to list of files to maintain
Does not add contents of files to repository
Committing files to Repository

hg commit
Adding/Committing in Eclipse

Commit changes to local Mercurial repository
Enter a commit message and select the files to commit.

User to record as committer: whitney

Select Files:
- .classpath
- .project
- settings/org.eclipse.jdt.core.prefs
- src/edu/sdsu/cs/cs580/Example.java

Select/unselect all
Show added/removed files
hg log

/junitexample

<table>
<thead>
<tr>
<th>Graph</th>
<th>Changeset</th>
<th>Tag</th>
<th>Branch</th>
<th>User</th>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>•</td>
<td>5:9a223dee086b</td>
<td>tip</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:47</td>
<td>-08 added a comment</td>
</tr>
<tr>
<td>•</td>
<td>4:67bdbab268f1</td>
<td></td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:45</td>
<td>-08C Just some code for an update</td>
</tr>
<tr>
<td>•</td>
<td>3:5ec9f6625587</td>
<td></td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:42</td>
<td>-08C Something</td>
</tr>
<tr>
<td>•</td>
<td>2:5fff016cfff3</td>
<td></td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:47</td>
<td>-08C Add test class</td>
</tr>
<tr>
<td>•</td>
<td>1:735c23eca0c5</td>
<td></td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:27</td>
<td>-08C Implemented adder</td>
</tr>
<tr>
<td>•</td>
<td>0:13e89dc77a90</td>
<td></td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:25</td>
<td>-08C Initial commit</td>
</tr>
</tbody>
</table>
Reverting to different Version

Please enter a valid revision (local, global, tag or branch):

3:Sec9f6625587833fbf99753f4222c2589c489d69

<table>
<thead>
<tr>
<th>Rev</th>
<th>Global</th>
<th>Date</th>
<th>Author</th>
<th>Branch</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>9a223dee086bf785425f90 2010-01-26 10:47 -0800</td>
<td>whitney</td>
<td></td>
<td>added a comment</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>67dbab26bf18990da3cc3 2010-01-26 10:45 -0800</td>
<td>whitney</td>
<td></td>
<td>just some code for an update</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sec9f6625587833fbf997532010-01-26 10:42 -0800</td>
<td>whitney</td>
<td></td>
<td>Something</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5ff016cffe3935668fc4fc51 2010-01-26 09:47 -0800</td>
<td>whitney</td>
<td></td>
<td>Add test class</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>735c23eca0c50a3fa357be5 2010-01-26 09:27 -0800</td>
<td>whitney</td>
<td></td>
<td>Implemented adder</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>13e89dc77a9077f2d09268 2010-01-26 09:25 -0800</td>
<td>whitney</td>
<td></td>
<td>Initial commit</td>
<td></td>
</tr>
</tbody>
</table>
Creating Branches

<table>
<thead>
<tr>
<th>Graph</th>
<th>Changeset</th>
<th>Tag</th>
<th>Branch</th>
<th>User</th>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>/junitexample</td>
<td>6771a04</td>
<td>tip</td>
<td>whitney</td>
<td>2010-01-26 11:38 -08</td>
<td>added a comment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:9a223dee086b</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:47 -08</td>
<td>added a comment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4:67bdab268f1</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:45 -08</td>
<td>Just some code for an update</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3:5ec9f6625587</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 10:42 -08</td>
<td>08C Something</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2:5fff016cfff3</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:47 -08</td>
<td>Add test class</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1:735c23eca0c5</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:27 -08</td>
<td>Implemented adder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:13e89dc77a90</td>
<td></td>
<td>whitney</td>
<td>2010-01-26 09:25 -08</td>
<td>Initial commit</td>
<td></td>
</tr>
</tbody>
</table>
Merging - without conflicts

use the merge item in the team menu
and commit
Uploading to BitBucket

After creating a project in BitBucket
Use the Push item in the team menu
Downloading From Bitbucket

Clone repository
Create a clone from another repository.

Repository location
URL: https://rogerwhitney@bitbucket.org/rogerwhitney/junitexample

Authentication
Username: rogerwhitney
Password: **************

Clone destination
Parent directory of clone (default: workspace)
Clone directory name (default: basename) junitExample

Options
- Abort clone when a timeout occurs
- Do not update the new working directory
- Use pull protocol to copy metadata
- Use uncompressed transfer (fast over LAN)
Changeset you would like to have after cloning

Finish
Unit Testing
Testing

Johnson's Law

If it is not tested it does not work

The more time between coding and testing

More effort is needed to write tests
More effort is needed to find bugs
Fewer bugs are found
Time is wasted working with buggy code
Development time increases
Quality decreases
Unit Testing

Tests individual code segments

Automated tests
What wrong with:

Using print statements

Writing driver program in main

Writing small sample programs to run code

Running program and testing it be using it
We have a QA Team, so why should I write tests?
When to Write Tests

First write the tests

Then write the code to be tested

Writing tests first saves time

Makes you clear of the interface & functionality of the code

Removes temptation to skip tests
What to Test

Everything that could possibly break

Test values
  Inside valid range
  Outside valid range
  On the boundary between valid/invalid

GUIs are very hard to test
  Keep GUI layer very thin
  Unit test program behind the GUI, not the GUI
Common Things Programs Handle Incorrectly

Adapted with permission from “A Short Catalog of Test Ideas” by Brian Marick,
http://www.testing.com/writings.html

Strings
Empty String

Collections
Empty Collection
Collection with one element
Collection with duplicate elements
Collections with maximum possible size

Numbers
Zero
The smallest number
Just below the smallest number
The largest number
Just above the largest number
XUnit

Free frameworks for Unit testing

SUnit originally written by Kent Beck 1994

JUnit written by Kent Beck & Erich Gamma

Available at: http://www.junit.org/

Ports to many languages at:
   http://www.xprogramming.com/software.htm
# XUnit Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.x</td>
<td>Old version&lt;br&gt;Works with versions of Java</td>
</tr>
<tr>
<td>4.x</td>
<td>Current version 4.8.1&lt;br&gt;Uses Annotations&lt;br&gt;Requires Java 5 or later</td>
</tr>
</tbody>
</table>
public class Adder {
    private int base;
    public Adder(int value) {
        base = value;
    }
    public int add(int amount) {
        return base + amount;
    }
}
Creating Test Case in Eclipse
Creating Test Case in Eclipse

Fill in dialog window & create the test cases
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
import org.junit.Test;

public class TestAdder {

    @Test
    public void testAdd() {
        Adder example = new Adder(3);
        assertEquals(4, example.add(1));
    }

    @Test
    public void testAddFail() {
        Adder example = new Adder(3);
        assertTrue(3 == example.add(1));
    }
}

Test Class
Running the Tests
The result

![JUnit Test Result](image)
Assert Methods

assertArrayEquals()
assertTrue()
assertFalse()
assertEquals()
assertNotEquals()
assertSame()
assertNotSame()
assertNull()
assertNotNull()
assertNotisNull()
fail()
Annotations

After
AfterClass
Before
BeforeClass
Ignore
Rule
Test
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

public class TestAdder {
    Adder example;
    @Before
    public void setupExample() {
        example = new Adder(3);
    }

    @Test
    public void testAdd() {
        assertEquals(4, example.add(1));
    }
}

Using Before