CSE 331 Software Design and Implementation

Handout T6: *Assignment Submission*

Contents:

Introduction

Students in CSE 331 are required to submit the implementation parts of their assignments electronically. Many assignments have two submission phases: the first is the original assignment submission (due by the date listed at the top of the assignment), and the second is a "returnin" submission, which gives you a second chance to improve your graded original submission.

We collect your assignments from your SVN repository. Thus, you must commit your files to SVN. For the original submission, this is sufficient. However, the validate step is *highly recommended*.

For assignments 1-4, you are required to resbumit your implementation after getting feedback from your TA. To resubmit your implementation for returnin, run the `returnin331` script *after* committing your files to SVN. The `returnin331` script tells the CSE 331 autograder to check out your assignment, run the test suite, and email the results to you.

validate

Validating your code checks it for common errors, such as your code not compiling correctly on CSE 331. Such errors could prevent you from receiving credit for your code, so you should always validate your assignment before you complete it (and before you run returnin331). However, validation is not guaranteed to catch all errors in your code.

You should validate your assignment by running `ant validate` on attu (and making sure that it completes successfully!), as follows:

```
cd ~/workspace331/cse331/src/hwN/
ant validate
```

or

```
cd ~/cse331/src/hwN/
ant validate
```

You need to select the correct version depending on where you have checked out your copy of your Subversion repository. If you have not yet checked out your repository on a

CSE Linux machine (say, because you work on Windows or at home), then you must first check out your repository using the command line. Note that if you check out your working copy in the location suggested there, the path to your project (as listed in the directions above and in the ouput below) will not include the `workspace331/` directory.

What does this do?

This checks out a fresh copy of your code (to a temporary directory) and ensures that your implementation:

- contains the required files
- compiles without errors. This is important because if you are not working in the Allen Center software labs, you may be using a different compiler. You may work wherever you like, but your code must work on `attu` (and must compile without errors or warnings).
- passes **the student's** `hwN.test.ImplementationTests` and `hwN.test.SpecificationTests`. (You can optionally read more about the CSE 331 JUnit Framework.)

What is the result?

If validation was successful, you should see output that looks something like:

```
Buildfile: /homes/iws/username/workspace331/cse331/src/hw1/build.xml

validate:
     [echo] Validate checks out a fresh copy of the hw, checks for the
     [echo]        presence of required files, and runs all your tests to make sure
     [echo]        they pass.  This target can only run on the attu IWS machine.
     [echo]
     [echo]        Note: the test reports will be generated under the scratch
     [echo]        directory the validate target creates.
     [echo]
   [delete] Deleting directory /homes/iws/username/workspace331/cse331/scratch
    [mkdir] Created dir: /homes/iws/username/workspace331/cse331/scratch
     [echo] /projects/instr/13sp/cse331/username/workspace331/REPOS
     [exec] A    cse331
     [exec] A    cse331/.classpath

     ...

     [exec] BUILD SUCCESSFUL
     [exec] Total time: 2 seconds
   [delete] Deleting directory /homes/iws/username/workspace331/cse331/scratch
```

If there is an error, the validate script should provide some information about what is wrong:

```
Buildfile: /homes/iws/username/workspace331/cse331/src/hw1/build.xml

validate:
     [echo] Validate checks out a fresh copy of the hw, checks for the
     [echo]        presence of required files, and runs all your tests to make sure
     [echo]        they pass.  This target can only run on the attu IWS machine.
     [echo]
     [echo]        Note: the test reports will be generated under the scratch
     [echo]        directory the validate target creates.
     [echo]

     ...

     [exec] cleancopy:
     [exec]      [echo] Hw directory: /homes/iws/username/workspace331/cse331/scratch/cse331/src/hw1
     [exec]
     [exec] BUILD FAILED
     [exec] /homes/iws/username/workspace331/cse331/scratch/cse331/src/common.xml:106: The following error occurred while
executing this line:
     [exec] /homes/iws/username/workspace331/cse331/scratch/cse331/src/common.xml:121: Could not find required file:
answers/hw1_answers.pdf
     [exec]
     [exec] Total time: 1 second
     [exec]
     [exec] cleancopy.check:

BUILD FAILED
/homes/iws/username/workspace331/cse331/src/common.xml:160: exec returned: 1
```

This error would indicate that a required file, `answers/hw1_answers.pdf` is missing. Make sure you've committed this file to SVN.

If the validate output indicates errors, you should fix them before the deadline, or you will lose points on your assignment. If validate failed because the public test suite failed, you can view a summary of the JUnit failures in your `YourWorkspaceDirectory/scratch/cse331/src/hwN/test/reports` directory.

Important: be aware that the validation script tests your code against **your own test suite**. Although by default we populate `hwN.test.SpecificationTests` with the public test suite, it is your responsibility to retain those tests in `hwN.test.SpecificationTests` if you want the validatation script to check your code against the public tests.

Why does this have to run on attu from the command line?

Most ant targets that the staff supplies should work both in the Allen Center software labs and on your home computer, but `ant validate` only works on `attu`. This is because we grade your solutions on attu, so it is important to verify that your code compiles and runs correctly in exactly that environment.

Eclipse's integrated Ant support does not handle the ant `validate` target well. Even if you use Eclipse as your development environment, you should validate *on the command line* as shown above.

You can log in to attu via SSH from any machine. (Even if you are working on an Allen Center Linux machine you still need to SSH into attu.)

returnin331

The `returnin331` program allows students to resubmit code for an assignment; it automatically grades your resubmission and gives you feedback. The returnin command tests that your implementation is correct. It does not check your specifications or documentation, though getting those right is likely to help you produce error-free code.

To use returnin, first correct and test your code on your own. Then, commit your code to your SVN repository and run both `ant clean total.build` and `ant validate` to ensure that your code can be compiled by our system and meets some minimal requirements (you don't want to use up one of your returnin tries due to a silly error!). Note: in order to pass returnin, *all of your hws must build*, even the ones that are newer than the one you are returning in! While we do only run tests on the given hw, all of your hws must compile; the `total.build` command above will ensure that. Finally, from your workspace directory, run

```
/cse/courses/cse331/13sp/bin/returnin331 hwN
```

to submit your work to our autograder. You will receive an email message indicating success or failure, usually within half an hour. If you do not receive a success/failure report within an hour, please send mail to cs331-staff@cs.washington.edu describing the problem.

If a student's implementation fails the staff test suite, the student can fix the code and resubmit a certain number of times without penalty. The student receives an email with the result of the resubmission. See the General Information handout for details about the requirements for, and rewards from, using returnin.

Returnin is worth 25% of the assignments to which it applies. You have three free tries to

run returnin and seven tries total. If any of your first three tries succeeds, you get full credit. Otherwise, you lose 5% for each returnin run beyond the third. (Your fourth attempt, if successful, earns you 20 points out of 25, your fifth earns you 15, and so on.) (Not all assignment grades add up to 100; for example, if returnin is worth 45 points, then you lose 9 each time.) If you do not pass on your seventh try, you will get no credit.

Limited feedback from returnin331

If you fail any test, you will receive only very limited feedback from returnin331 - typically the name of the staff test that failed, and one line indicating an expected and actual result, or the message from an exception. You will not receive a full stack trace, nor the actual test case.

Oftentimes, the information you receive is plenty to help you reproduce, understand, and correct the defect in your program. If the information you receive is not sufficient, then you should use reasoning, defensive programming, and testing to prevent errors or make them more debuggable. The purpose of returnin is to reward students who are doing their work systematically and carefully, and to help the other students to learn to work in that way.

Back to tools.
Back to the CSE 331 home page.
For problems or questions regarding this page, contact: cse331-staff@cs.washington.edu.
