

Tutorial IV: Unit Test

- What is Unit Test
 - Three Principles
 - Testing frameworks: JUnit for Java
CppUnit for C++
Unit Test for Web Service
- <http://www.cs.toronto.edu/~yijun/csc408h/handouts/unittest-HOWTO.html>

What is Unit Test?

- Unit Test:
A unit can be an operation, a class, a software package, or a subsystem
- Integration Test:
Interactions between units
- System Test:
System verification and validation as a whole
- Acceptance Test:
Testing as a end user; Expected results from system

Three Principles

- Testing as you go: the earlier a bug is found, the better!
- Test can be done once a unit is ready:
Bottom-up testing: with Drivers
Top-down testing: with Stubs
- Design test cases systematically:
Include boundary values for each feature
Make sure every line of code is executed

What can be tested in units?

- A functional requirement
- Given input that satisfies the precondition, whether the output satisfies the post-condition
- A unit can be a member function, a class, a package or component or a subsystem ...
- Automation is the key! Replace user interaction with the scripts, if possible; replace some units with stubs
- A unit tested can still have bugs, but most trivial bugs should have been found

What can not?

- Generally, test can not replace the verification or code review
- Specifically for unit test, interactions between this unit and other units after integration, system and user acceptance are not possible when the system is not ready yet

JUnit and Example

- Refer to: <http://www.junit.org>
- Some concepts or classes:
 - Fixture:** a set of objects against which tests are run
 - Test Case:**
 - a class which defines the fixture to run multiple tests
 - create a subclass of TestCase
 - add an instance variable for each part of the fixture
 - override [setUp\(\)](#) to initialize the variables
 - override [tearDown\(\)](#) to release any permanent resource allocated in setUp
 - setup:** a method which sets up the fixture, called before a test is executed.
 - tearDown:** a method to tear down the fixture, called after a test is executed.
 - Test Suite:** a collection of test cases.

JUnit and Example (cont'd)

TestRunner: a tool to define the test suite to be run and to display its results

- A JUnit example (in Eclipse):
 - source code: [junit\samples\money \(simplified\)](#)
 - functionality: single currency arithmetic

CppUnit and Example

- Refer to: <http://cppunit.sourceforge.net/cgi-bin/moin.cgi>
- A compiled CppUnit module in CDF
 - [/u/yijun/software/cppunit-1.10.2](#)
- An example of CppUnit
 - [/cppunit-1.10.2/examples/money](#)

Develop Web service in AXIS

See [/u/yijun/software/axis-1_1/addr.sh](#)

`deploy.wsdd`, `undeploy.wsdd` can be generated from WSDL:

- `java -cp $AXISCLASSPATH org.apache.axis.wsdl.WSDL2Java -s -d Session -Nurn:AddressFetcher2=samples.addr samples/addr/AddressBook.wsdl`

Start a simple Axis server

- `java -cp .:$AXISCLASSPATH org.apache.axis.transport.http.SimpleAxisServer -p 9012 &`

Deploy the web service

- `java -cp $AXISCLASSPATH org.apache.axis.client.AdminClient -p 9012 samples/addr/deploy.wsdd`

Call the web service from the client program

- `java -cp .:$AXISCLASSPATH samples.addr.Main -p 9012 $*`

Feedback from the client

```
Using proxy without session maintenance.
(queries without session should say: "ADDRESS NOT FOUND!")
>> Storing address for 'Purdue Boilermaker'
>> Querying address for 'Purdue Boilermaker'
>> Response is:
    [ADDRESS NOT FOUND!]
>> Querying address for 'Purdue Boilermaker' again
>> Response is:
    [ADDRESS NOT FOUND!]
Using proxy with session maintenance.
>> Storing address for 'Purdue Boilermaker'
>> Querying address for 'Purdue Boilermaker'
>> Response is:
    1 University Drive
    West Lafayette, IN 47907
    Phone: (765) 494-4900
>> Querying address for 'Purdue Boilermaker' again
>> Response is:
    1 University Drive
    West Lafayette, IN 47907
    Phone: (765) 494-4900
```

Test Web Service using JUnit

Test Cases (e.g. `AddressBookTestCase.java`) can be generated by:

- `java -cp $AXISCLASSPATH org.apache.axis.wsdl.WSDL2Java -s -d Session -Nurn:AddressFetcher2=samples.addr --testCase samples/addr/AddressBook.wsdl`

Modify the generated `AddressBookTestCase.java` :

```
public void doTest () throws Exception {
    String[] args = {"-p", "9012"};
    Main.main(args);
}
```

Run the following command:

- `java -cp .:$AXISCLASSPATH junit.textui.TestRunner -noloding samples.addr.AddressBookTestCase`

Feedback from the Unit Test

```
• - Testing address book sample.
• Using proxy without session maintenance.
• (queries without session should say: "ADDRESS NOT FOUND!")
• >> Storing address for 'Purdue Boilermaker'
• >> Querying address for 'Purdue Boilermaker'
• >> Response is:
•     [ADDRESS NOT FOUND!]
• >> Querying address for 'Purdue Boilermaker' again
• >> Response is:
•     [ADDRESS NOT FOUND!]
• Using proxy with session maintenance.
• >> Storing address for 'Purdue Boilermaker'
• >> Querying address for 'Purdue Boilermaker'
• >> Response is:
•     1 University Drive
•     West Lafayette, IN 47907
•     Phone: (765) 494-4900
• >> Querying address for 'Purdue Boilermaker' again
• >> Response is:
•     1 University Drive
•     West Lafayette, IN 47907
•     Phone: (765) 494-4900
• - Test complete.

• Time: 1.51

• OK (1 test)
```