
'06
JAVAPOLIS
11 - 15 DECEMBER ■ ANTWERP ■ BELGIUM
Adobe
JBoss
a division of Red Hat
ORACLE
Sun
microsystems
The Network is the Computer™

Unitils

Make unit testing easy and maintainable

Filip Neven
Sr. Software Engineer
Ordina Belgium

Goal of this Quicky

Unitils can help your team to implement maintainable unit tests.

Origin of the project

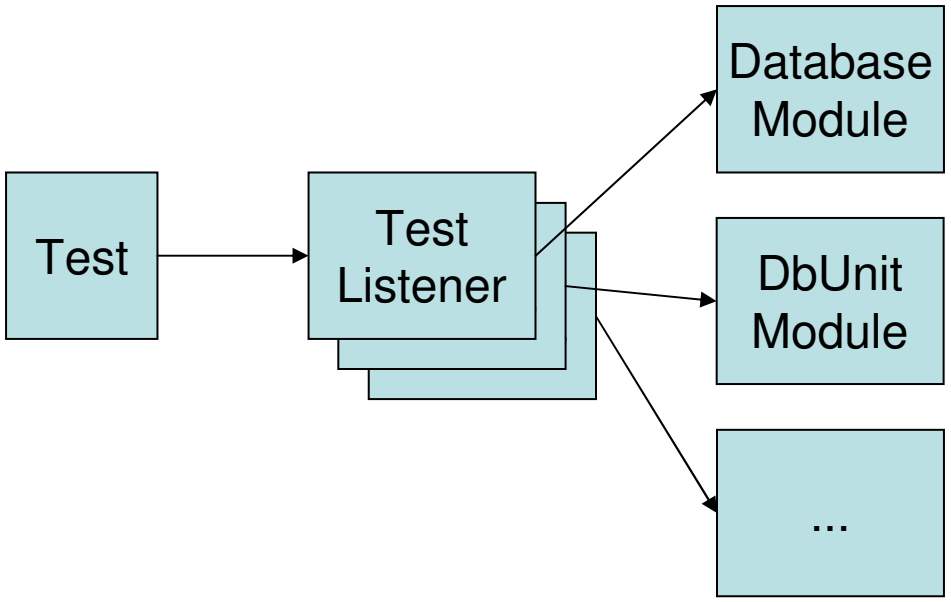
- Ordina unit testing workgroup
  - Guidelines
  - Supporting code
    - → Unitils

Unitils Features

- Database testing
  - Test data management
    - DBUnit
  - Test database maintenance
  - Testing with Hibernate
- Assertion utils
- Mock objects
  - Mock creation and setting expectations
    - EasyMock
  - Mock injection

Open & extensible architecture

- Module system
  - Listener hooks up modules into tests



  - Support for JUnit 3 / 4 + TestNG

Database testing

- Control over test data
  - Unit test database without data
  - Small test data sets
- Limited sharing of data sets
  - Test class or method level

Class-level data file

UserDAOTest.java

```
@DatabaseTest
public void UserDAOTest extends UnitilsJUnit4 {

    @Test public void testFindByName {
        User johnDoe = userDao.findByName("doe",
                    "john");
        assertEquals("jdoe", johnDoe.getUserName());
    }

}
```

UserDAOTest.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<dataset>
  <user userName="jdoe" name="doe" firstName="john" />
</dataset>
```

Method-level data file

UserDAOTest.java

```
@DatabaseTest
public void UserDAOTest extends UnitilsJUnit4 {

    @Test public void testFindByName {
        User johnDoe = userDao.findByName("doe",
                    "john");
        assertEquals("jdoe", johnDoe.getUserName());
    }

}
```

UserDAOTest.testFindByName.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<dataset>
  <user userName="jdoe" name="doe" firstName="john" />
</dataset>
```

Specify file name explicitly

UserDAOTest.java

```
@DatabaseTest
@DataSet(fileName = "userdata.xml")
public void UserDAOTest extends UnitilsJUnit4 {

    @Test
    @DataSet(fileName = "userdata-specific.xml")
    public void testFindByName {
        User johnDoe = userDao.findByName("doe",
                    "john");
        assertEquals("jdoe", johnDoe.getUserName());
    }

}
```

Result data file

UserDAOTest.java

```
@DatabaseTest
public void UserDAOTest extends UnitilsJUnit4 {

    @ExpectedDataSet
    @Test public void testDisableInactiveAccounts {
        userDao.disableInactiveAccounts();
    }

}
```

UserDAOTest.testDisableInactiveAccounts-result.xml

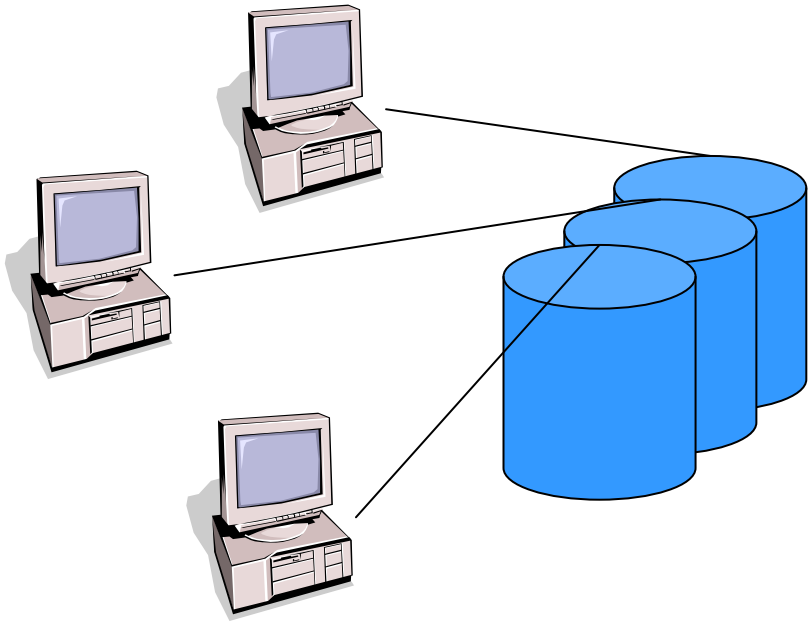
```
<?xml version='1.0' encoding='UTF-8'?>
<dataset>
    <user userName="jack" enabled="true" />
    <user userName="jim" enabled="false" />
</dataset>
```

Keeping tests maintainable

- Data files must be small
  - Few records
  - Only join & where clause columns
    - →Disable FK & NOT NULL constraints

Execute unit tests anytime

- Database schema per developer

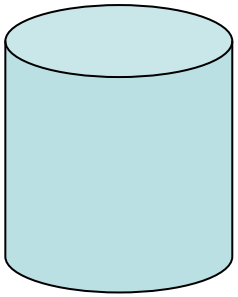


Database schema maintenance

- Update unit test schema
  - Incrementally
  - From scratch
- Disable constraints
  - Foreign key
  - NOT NULL
- Set sequences to value 1000
  - Don’t interfere with test data

Database schema maintenance

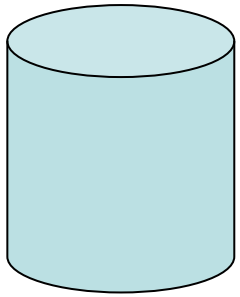
```
DbScripts/
        001_initial_structure.sql
        002_add_user_table.sql
        003_add_version_column.sql
```



DB_VERSION	INDEX	TS
	3	510894628

Incremental updates

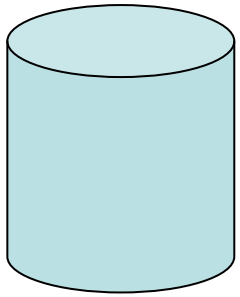
```
DbScripts/
        001_initial_structure.sql
        002_add_user_table.sql
        003_add_version_column.sql
        004_drop_age_column.sql
```



DB_VERSION	INDEX	TS
	3	510894628

Incremental updates

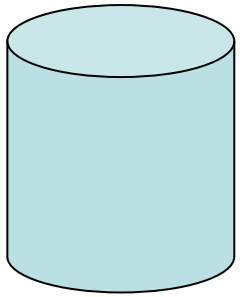
```
DbScripts/
        001_initial_structure.sql
        002_add_user_table.sql
        003_add_version_column.sql
        004_drop_age_column.sql
```



DB_VERSION	INDEX	TS
	4	613874226

Recreation from scratch

```
DbScripts/
        001_initial_structure.sql
        002_add_user_table.sql
        003_add_version_column.sql
        004_drop_age_column.sql
```



DB_VERSION	INDEX	TS
	4	510894628

Hibernate support

- Simplify configuration
- Automatically check consistency of mapped classes with database

```
junit.framework.AssertionFailedError:
Found mismatches between Java classes and database tables.
Applying following DDL statements to the database should resolve
the problem:

alter table PRODUCT add column barCode varchar(255);
```

Assertion utils

```
Expected

jdoeExp
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
Actual

jdoeAct
        userName=jdoe
        first=john
        last=doe
        age=20
```

Assertion utils

```
Expected

jdoeExp
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
Actual

jdoeAct
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
assertEquals(jdoeExp, jdoeAct);
```

Assertion utils

Expected	*Actual*
jdoeExp userName=jdoe first=john last=doe age=20	jdoeAct userName=jdoe first=john last=doe age=20

```
assertEquals(jdoeExp, jdoeAct);
```

```
assertEquals(jdoeExp.getUsrName(), jdoeAct.getUsrName());
assertEquals(jdoeExp.getFirst(), jdoeAct.getFirst());
assertEquals(jdoeExp.getLast(), jdoeAct.getLast());
assertEquals(jdoeExp.getAge(), jdoeAct.getAge());
```

Assertion utils

```
Expected

jdoeExp
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
Actual

jdoeAct
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
assertEquals(jdoeExp, jdoeAct);
```

```
assertEquals(jdoeExp.getUsrName(), jdoeAct.getUsrName());
assertEquals(jdoeExp.getFirst(), jdoeAct.getFirst());
assertEquals(jdoeExp.getLast(), jdoeAct.getLast());
assertEquals(jdoeExp.getAge(), jdoeAct.getAge());
```

```
ReflectionAssert.assertRefEquals(jdoeExp, jdoeAct);
```

Maintainability

- Only check what's essential
  - Ignore type of numbers
  - Ignore certain properties
  - Ignore order and type of collections

Ignore certain properties

- Leniency option: Ignore defaults

```
Expected

jdoeExp
        userName=jdoe
        first=null
        last=null
        age=0
```

```
Actual

jdoeAct
        userName=jdoe
        first=john
        last=doe
        age=20
```

```
ReflectionAssert.assertLenEquals(jdoeExp, jdoeAct);
```

Ignore order and type of collections

- Leniency option: collections

Expected	*Actual*
`expPersons: Person[]`	`actPersons: HashSet<Person>`
`[john, jane]`	`[jane, john]`

```
ReflectionAssert.assertLenEquals(expPersons, actPersons);
```

Ignore order and type of collections

- Leniency option: collections

Expected	*Actual*
`expPersons: Person[]`	`actPersons: HashSet<Person>`
`[john, jane]`	`[jane, john]`

```
ReflectionAssert.assertLenEquals(expPersons, actPersons);
```

```
ReflectionAssert.assertPropertyLenEquals("firstName",
      Arrays.asList("jane", "john"), actPersons);
```

Mock object support

```
public void UserServiceTest extends UnitilsJUnit4 {

    @Mock
    private UserDao mockUserDao;

    @Mock
    private AccountService mockAccountService;

    private UserService userService;

    @Test testDisableInActiveAccounts() {
        expect(mockUserDao.getAccountsNotAccessedAfter(null))
            .andReturn(accounts);
        mockAccountService.disableAccount(accounts.get(0));
        mockAccountService.disableAccount(accounts.get(1));
        replay();

        userService.disableInActiveAccounts();
    }
}
```

Mock injection

```
public void UserServiceTest extends UnitilsJUnit4 {

    @Mock @Inject(property = "userDao")
    private UserDao mockUserDao;

    @Mock @AutoInject
    private AccountService mockAccountService;

    @TestedObject
    private UserService userService;

    @Test testDisableInActiveAccounts() {
        expect(mockUserDao.getAccountsNotAccessedAfter(null))
            .andReturn(accounts);
        mockAccountService.disableAccount(accounts.get(0));
        mockAccountService.disableAccount(accounts.get(1));
        replay();

        userService.disableInActiveAccounts();
    }
}
```

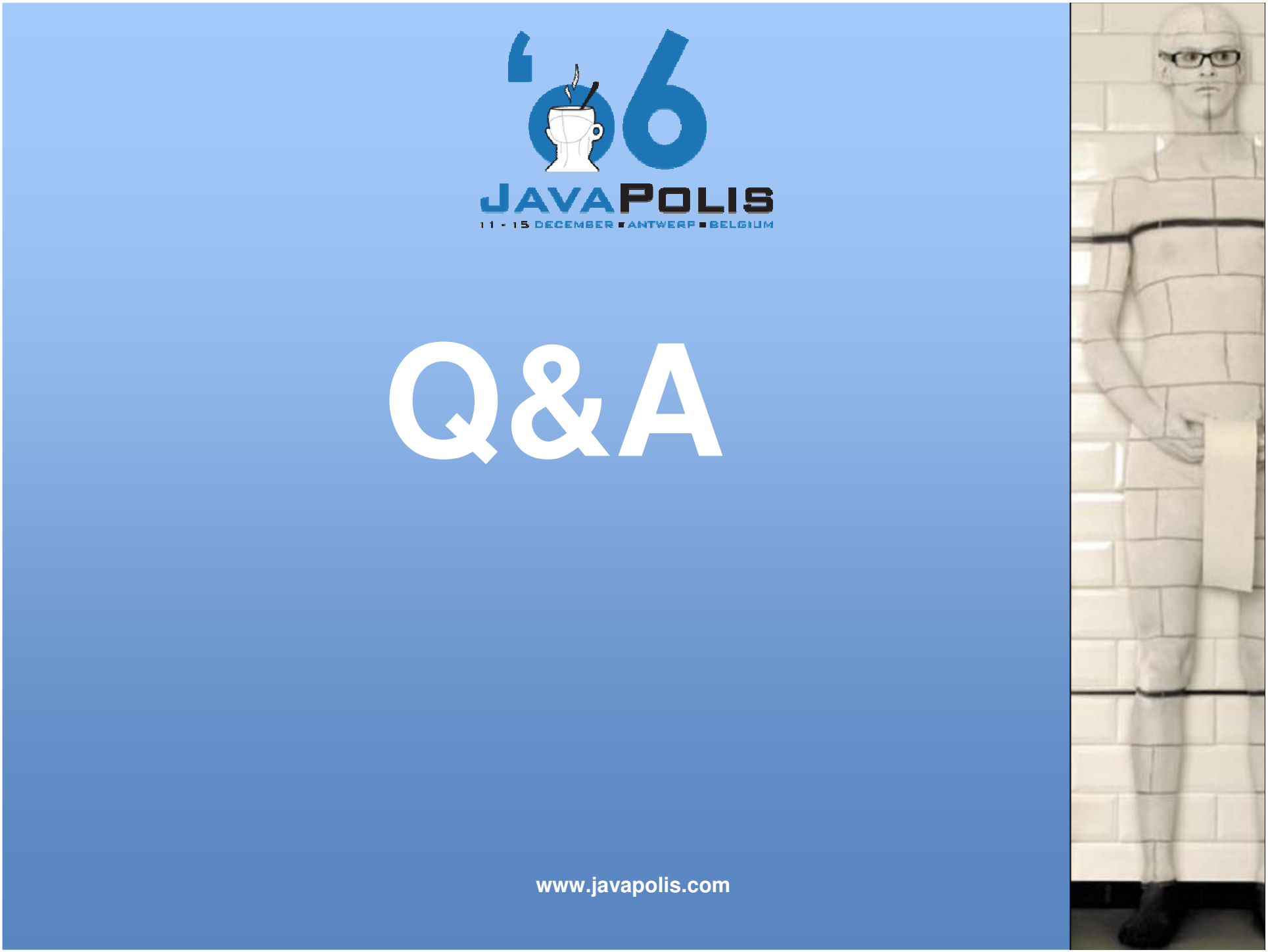
Links

Unitils website:

http://unitils.sourceforge.net

Unit testing guidelines:

http://unitils.sourceforge.net/guidelines


'06
JAVAPOLIS
11 - 15 DECEMBER ■ ANTWERP ■ BELGIUM
Q&A
www.javapolis.com