

Mobile Application Development

Higher Diploma in Science in Computer Science

Produced
by

John Fitzgerald (jfitzgerald@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>

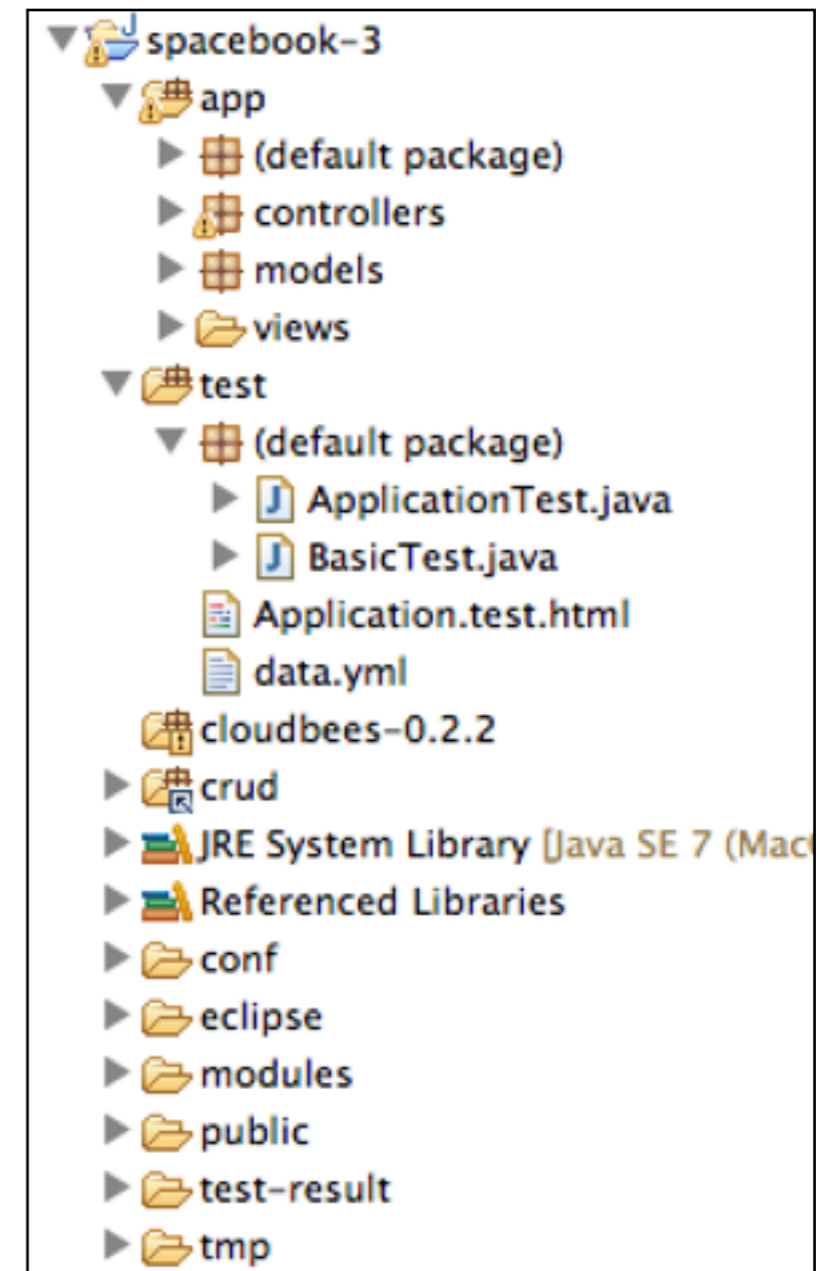


Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE



Test Driven Development & Play

- Play comes with TDD support built in
- 'test' folder contains placeholders for simple tests
- Write standard 'JUnit' tests in here



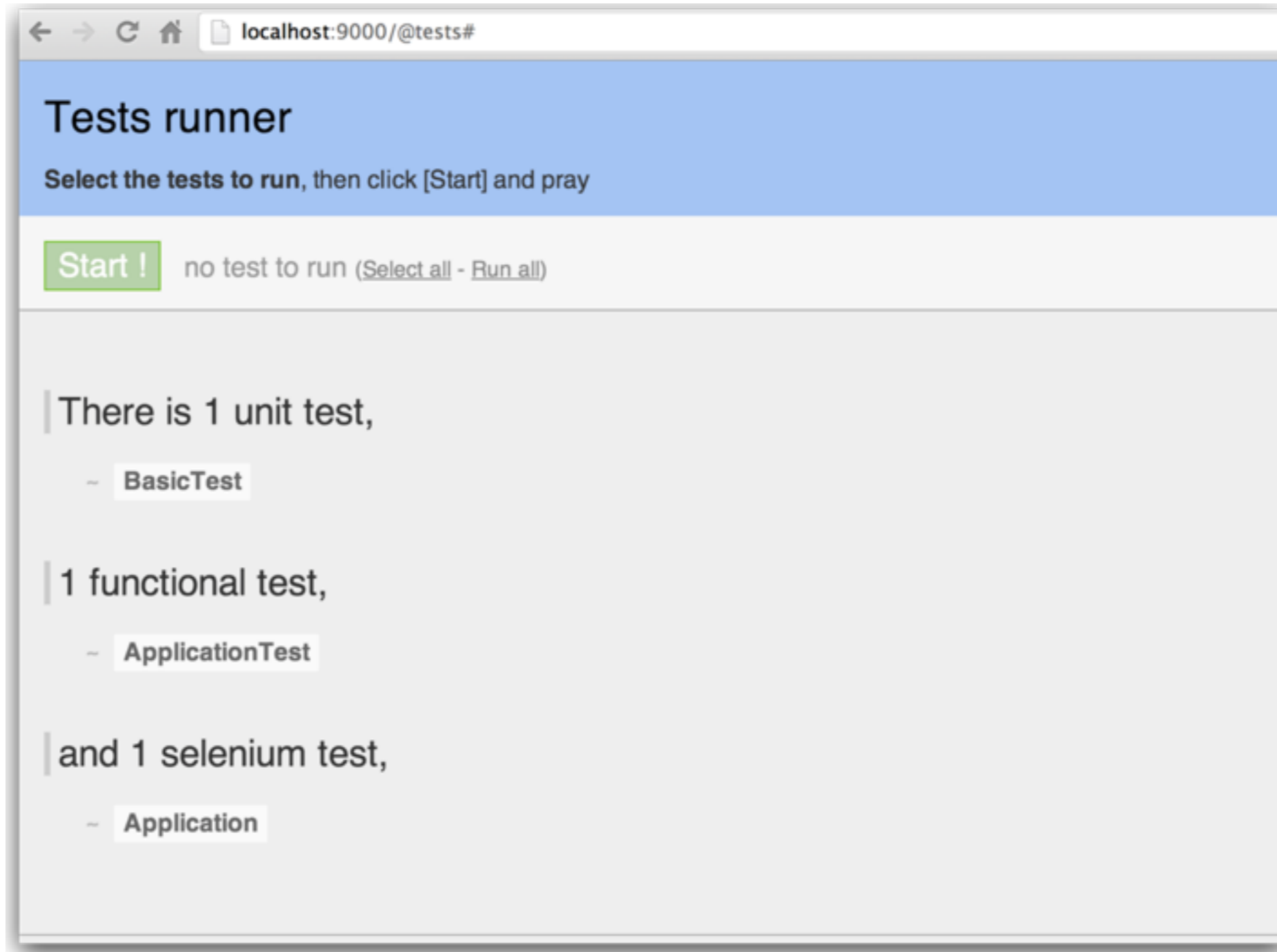
Run Play in Test Mode

- Type 'play test' instead of 'play run'

```
$ play test
~
~  _ _ _ | | _ _ _ _ _ | |
~ | ' _ \ | / _ ' | | | | |
~ | _ / | \ _ _ | \ _ ( )
~ | |      | | /
~
~
~ play! 1.2.5, http://www.playframework.org
~ framework ID is test
~
~ Running in test mode
~ Ctrl+C to stop
~
CompilerOracle: exclude jregex/Pretokenizer.next
Listening for transport dt_socket at address: 8000
12:56:48,840 INFO ~ Starting /Users/edelestar/repos/modules/app-dev-modelling-prj/spacebook-3
12:56:48,843 INFO ~ Module cloudbees is available (/Users/edelestar/repos/modules/app-dev-modelling-prj/spacebook-3/
modules/cloudbees-0.2.2)
12:56:48,844 INFO ~ Module crud is available (/Users/edelestar/dev/play-1.2.5/modules/crud)
12:56:49,353 WARN ~ You're running Play! in DEV mode
~
~ Go to http://localhost:9000/@tests to run the tests
~
12:56:49,420 INFO ~ Listening for HTTP on port 9000 (Waiting a first request to start) ...
```

Play JUnit Test Runner

- Browse to '<http://localhost/@tests>'



Selecting Tests...

localhost:9000/@tests#

Tests runner

Select the tests to run, then click [Start] and pray

Start ! 1 test to run ([Bookmark this link to save this configuration](#)) - [Unselect all](#)

| There is 1 unit test,

- ~ **BasicTest**

| 1 functional test,

- ~ **ApplicationTest**

| and 1 selenium test,

- ~ **Application**

Running Tests... Success

localhost:9000/@tests#

Tests runner

Select the tests to run, then click [Start] and pray

Start ! 1 test to run ([Bookmark this link to save this configuration](#)) - [Unselect all](#)

| There is 1 unit test,

- + **BasicTest**

| 1 functional test,

- ApplicationTest

| and 1 selenium test,

- Application

Test Source

- Simple test to see if test infrastructure is operating successfully

```
public class BasicTest extends UnitTest
{
    @Test
    public void aVeryImportantThingToTest()
    {
        assertEquals(2, 1 + 1);
    }
}
```

Test Source - Deliberate Error

- Introduce change we know will definitely fail...

```
public class BasicTest extends UnitTest
{
    @Test
    public void aVeryImportantThingToTest()
    {
        assertEquals(3, 1 + 1);
    }
}
```


Tests runner

Select the tests to run, then click [Start] and pray

Start !

1 test to run ([Bookmark this link to save this configuration](#)) - [Unselect all](#)

| There is 1 unit test,

- **BasicTest**

aVeryImportantThingToTest

Failure, expected:<3> but was:<2>

3 ms

In /test/BasicTest.java, line 11 :

```
assertEquals(3, 1 + 1);
```

Show trace

| 1 functional test,

- **ApplicationTest**

| and 1 selenium test,

- **Application**

Test for the User Model class

```
@Test
public void testCreateUser()
{
    User bob = new User("bob", "jones", 20, "irish", "bob@jones.com", "secret");
    bob.save();

    User testUser = User.findByEmail("bob@jones.com");
    assertNotNull (testUser);
}
```

- Simple test:
 - Create a user object 'bob'
 - Store it in the database
 - Try to find it in the database
- This test confirms that we can write/read simple object from the database

Another User Test

```
@Test
public void testFindUser()
{
    User jim = new User("jim", "smith", 20, "irish", "jim@smith.com", "secret");
    jim.save();

    User test = User.findByEmail("jim@smith.com");
    assertNotNull (test);

    User alice = User.findByEmail("alice@jones.com");
    assertNull (alice);
}
```

- Similar to last test, except:
 - Confirm that if we try to find a non-existent user, we get 'null'
- This test confirms that a null will be returned when querying got a non-existent user

Message Test (1)

```
@Test
public void testCreateMessage()
{
    User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
    mary.save();

    User joan = new User("joan", "collins", 20, "irish", "joan@collins.com", "secret");
    joan.save();

    Message msg = new Message (mary, joan, "Hi there - how are you");
    msg.save();

    List<Message> joansMessages = Message.find("byTo", joan).fetch();
    assertEquals (joansMessages.size(), 1);
}
```

Message Test (2)

```
@Test
public void testNoMessagese()
{
    User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
    mary.save();

    List<Message> joansMessages = Message.find("byTo", mary).fetch();
    assertEquals (joansMessages.size(), 0);
}
```

Message Test (3)

```
@Test
public void testMultipleMessages()
{
    User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
    mary.save();

    User joan = new User("joan", "collins", 20, "irish", "joan@collins.com", "secret");
    joan.save();

    Message msg1 = new Message (mary, joan, "Hi there - how are you");
    msg1.save();
    Message msg2 = new Message (mary, joan, "Where are you now?");
    msg2.save();

    List<Message> joansMessages = Message.find("byTo", joan).fetch();
    assertEquals (joansMessages.size(), 2);
    Message message1 = joansMessages.get(0);
    assertEquals(message1.messageText, "Hi there - how are you");
    Message message2 = joansMessages.get(1);
    assertEquals(message2.messageText, "Where are you now?");
}
```

Create Tests for Each Model Object

UserTest

```
public class UserTest extends UnitTest
{
    @BeforeClass
    public static void loadDBC()
    {
        Fixtures.deleteAllModels();
    }

    @Test
    public void testCreateUser()
    {
        User bob = new User("bob", "jones", 20, "irish", "bob@jones.com", "secret");
        bob.save();

        User testUser = User.findByEmail("bob@jones.com");
        assertNotNull (testUser);
    }

    @Test
    public void testFindUser()
    {
        User jim = new User("jim", "smith", 20, "irish", "jim@smith.com", "secret");
        jim.save();

        User test = User.findByEmail("jim@smith.com");
        assertNotNull (test);

        User alice = User.findByEmail("alice@jones.com");
        assertNull (alice);
    }
}
```

MessageTest

```
public class MessageTest extends UnitTest
{
    @BeforeClass
    public static void loadDBC()
    {
        Fixtures.deleteAllModels();
    }

    @Test
    public void testCreateMessage()
    {
        User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
        mary.save();

        User joan = new User("joan", "collins", 20, "irish", "joan@collins.com", "secret");
        joan.save();

        Message msg = new Message (mary, joan, "Hi there - how are you");
        msg.save();

        List<Message> joansMessages = Message.find("byTo", joan).fetch();
        assertEquals (joansMessages.size(), 1);
    }

    @Test
    public void testNoMessages()
    {
        User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
        mary.save();

        List<Message> joansMessages = Message.find("byTo", mary).fetch();
        assertEquals (joansMessages.size(), 0);
    }

    @Test
    public void testMultipleMessages()
    {
        User mary = new User("mary", "collins", 20, "irish", "mary@collins.com", "secret");
        mary.save();

        User joan = new User("joan", "collins", 20, "irish", "joan@collins.com", "secret");
        joan.save();

        Message msg1 = new Message (mary, joan, "Hi there - how are you");
        msg1.save();
        Message msg2 = new Message (mary, joan, "Where are you now?");
        msg2.save();

        List<Message> joansMessages = Message.find("byTo", joan).fetch();
        assertEquals (joansMessages.size(), 2);
        Message message1 = joansMessages.get(0);
        assertEquals(message1.messageText, "Hi there - how are you");
        Message message2 = joansMessages.get(1);
        assertEquals(message2.messageText, "Where are you now?");
    }
}
```

Test Runner Runs All Tests

Tests runner

Select the tests to run, then click [Start] and pray

Start !

2 tests to run ([Bookmark this link to save this configuration](#)) - [Unselect all](#)

There are 2 unit tests,

- MessageTest

testCreateMessage	Ok	5 ms
testNoMessage	Ok	2 ms
testMultipleMessages	Ok	5 ms

- UserTest

testCreateUser	Ok	3 ms
testFindUser	Ok	3 ms



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

