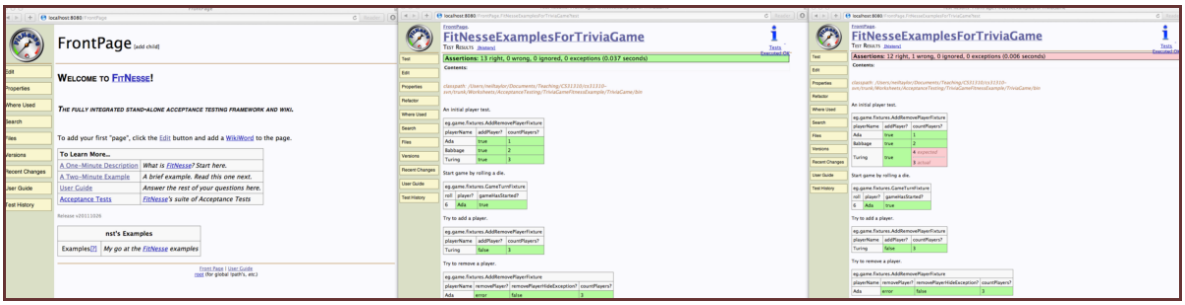
CS31310 Agile Methodologies • Worksheet 1

Automated Acceptance Testing using
FitNesse and the FIT/Slim Frameworks



Introduction

This exercise gives you the opportunity to download and experiment with a free automated testing framework called FitNesse, which was discussed briefly in the lectures. This includes Ward Cunningham's FIT test system and an alternative test system called SLIM. You will follow the steps below to install and use this framework on a simple Java application.

This worksheet is assessed and it represents 3% of your overall module assessment.

Steps

To complete this worksheet, you will follow the steps listed below. Detailed instructions for each step can be found in the following pages.

1. Download, install and run FitNesse.
2. Read the pages *A One-Minute Description* and *A Two-Minute Example*, which are found on the front page of the FitNesse Wiki that you start.
3. Read the *User Guide* and in particular the *Fixture Code* tutorial that can be found in the *Tutorial* section of the guide.
4. Download the Eclipse project from the Assignment link on Blackboard. Link the code from the TriviaGame Eclipse project and build your own example Test page.
5. Read and listen to two online resources.
6. Write a short document that includes screenshots of the pages that you have created and a brief discussion of the usefulness of tools such as FitNesse.
7. Submit your document as a PDF file to Blackboard. This should be submitted by **6pm Monday 12th November**.

1. Download, Install and Run FitNesse

FitNesse is available at www.fitnesse.org. On the front page, click on the link in the main page called *Download FitNesse and Plugins*. On the download page, look for the file link for fitnesse.jar, which was released on 2011-10-26. This is a 4Mb download.

To run FitNesse, you can use `java -jar fitnesse.jar`. By default, FitNesse will attempt to run on port 80 on your machine. However, if port 80 is already in use on your machine, you can specify another port number. In the following example, you can see that `-p 8080` is used to specify that the tool should run on port 8080.

```
$ java -jar fitnesse.jar -p 8080
Unpacking new version of FitNesse resources.  Please be patient.
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................FitNesse
(v20111026) Started...
      port:              8080
      root page:         fitnesse.wiki.FileSystemPage at ./FitNesseRoot
      logger:            none
      authenticator:     fitnesse.authentication.PromiscuousAuthenticator
      html page factory: fitnesse.html.HtmlPageFactory
      page version expiration set to 14 days.
```

The first time that FitNesse runs, it will create a directory called FitNesseRoot in the directory in which you run the tool.

2. Read the One-Minute Description and Two-Minute Example

Once FitNesse is running, open a web browser to view the content of the Wiki. If you are running the software on the standard port 80, then you can access http://localhost/. If you are running on a different port, then you should specify the port number in the URL. For example, if you are running on port 8080, you can access http://localhost:8080/.

If you are on a University machine and the access doesn't work, then you might have to change your browser settings to make sure that localhost is on the exclusion list. Sometimes, http://127.0.0.1/ will also work, but not if you are on the Aber network because it is considered a potential virus attack and therefore not allowed. Another possibility is to use the IP address of your machine.

Once you can access the Wiki, read the One and Two-Minute guides, which are listed on the front page, as shown in Figure 1.

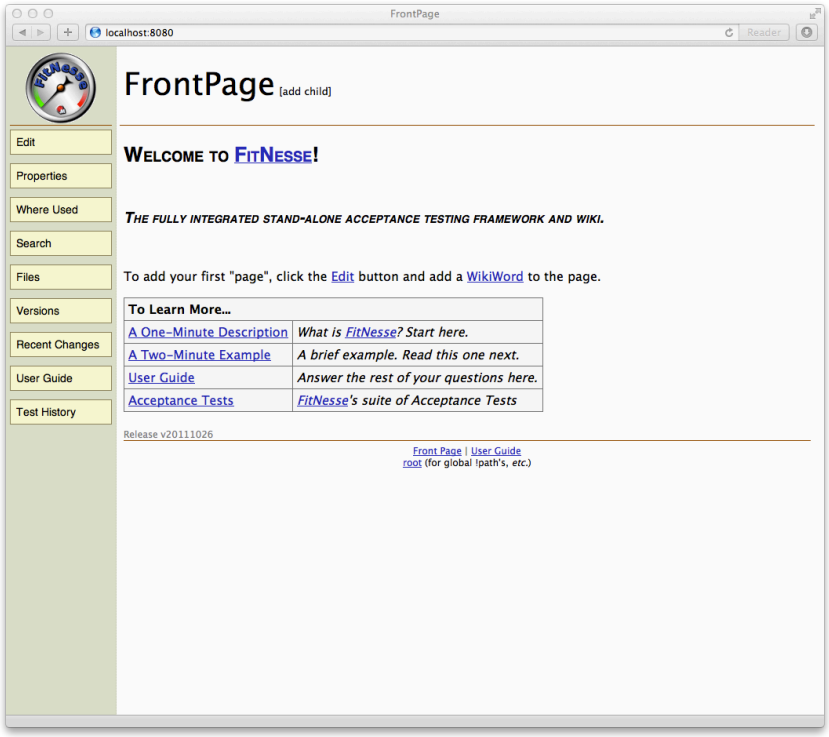


Figure 1: Front Page

3. Read the User Guide

Access the User Guide, which is also available on the front page of the Wiki. Read the text on the User Guide and then look at the *Fixture Code* tutorial, which can be found in the section *Tutorial: FitNesse Fixtures*, at the bottom of the User Guide page.

4. Build your own example Test page

In this section, you will edit the Wiki and add test tables that link to example code provided with this worksheet.

4.1. Download the code

On Blackboard, download the ZIP file that contains example code for part of a Trivia Game. The code is similar to the code in the *Fixture Code* tutorial, but there are some minor differences.

The ZIP file contains an Eclipse project. The src directory contains code in the following packages:

- *eg.game* – The example code for the game.
- *eg.game.fixtures* – The test fixtures that are accessed by FitNesse. These link to the game code in *eg.game*.
- *eg.game.tests* – A set of unit tests for the game code.

Take some time to study the code.

The code has been compiled using Java 7. The class files are provided in the bin directory. You do not need to re-compile this code in order to complete the worksheet. However, you will learn more about this general topic by looking at the code and changing a few items of code or adding additional functionality. If you make changes to the functionality, then you could try to link the changes to the test tables in the Wiki in the following sections.

Note: the class files in the bin directory will be linked to the tests in FitNesse. The path to the bin directory will be specified as a classpath in the Wiki page. There can be problems if the path has spaces in parts of the name. Therefore, locate the downloaded project in a path that does not contain spaces.

4.2 Creating a new Wiki page

On the front page of the Wiki, select the Edit button[1]. A new screen will be displayed with a text edit box. At the bottom of the text that is shown in the editor, add the following lines. Remember to change nst to your user id.

```
|!c '''nst's Examples'''|
|!c [[Examples][^FitNesseExamplesForTriviaGame]]|''My go at the FitNesse
examples''|
```

Press the Save button to commit the change. Your front page should now have an additional table, which is similar to that shown in Figure 2.

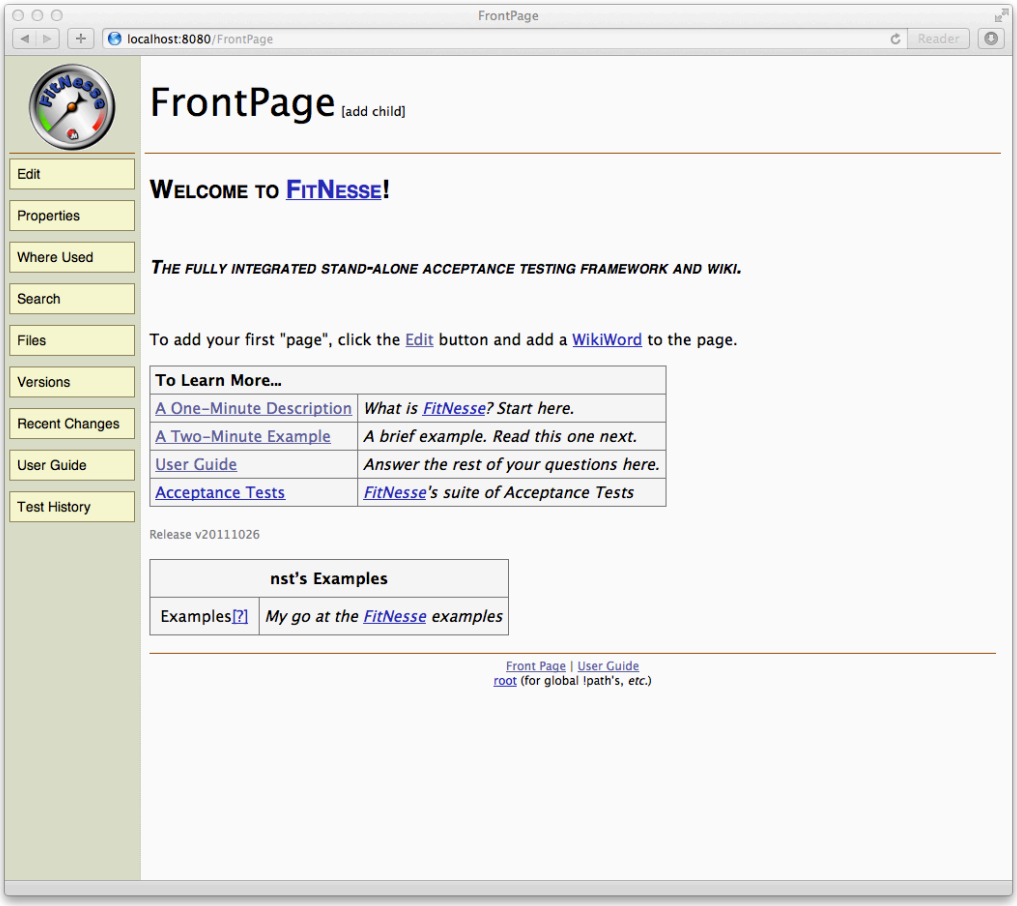


Figure 2 Front Page with edited table

Click on the [?] link next to *Examples*. A new page will be created.

4.3 Adding a Test Table Page

Click on the Edit button to show the editor for the page. At the top of the page, you need to specify the path to the class files that will be tested. If you are using Windows, the instruction will be of the form:

```
!path C:\UWA\Modules\CS31310\exampleCode\FITExamples\
```

[1] If the *Edit* button is not shown, click on the *Properties* option. Under *Actions*, select *Edit*. Finally, press the *Save Properties* button.

If you are on a Unix based platform, then an appropriate path would be:

```
!path /UWA/Modules/CS31310/exampleCode/FITExamples/
```

Whichever system you are using, the path needs to point to the bin directory in the files that you extracted from the ZIP file in step 4.1.

Within the editor, following the `!path` directive, you need to add tables that allow you to exercise the different parts of the AddRemovePlayerFixture and the GameTurnFixture. An example set of tables is shown in Figure 3 and the layout for one table is shown below. The tables should contain your own data.

```
An initial player test.

!|eg.game.fixtures.AddRemovePlayerFixture|
|playerName|addPlayer?|countPlayers?|
|Ada|true|1|
|Babbage|true|2|
|Turing|true|4|
```

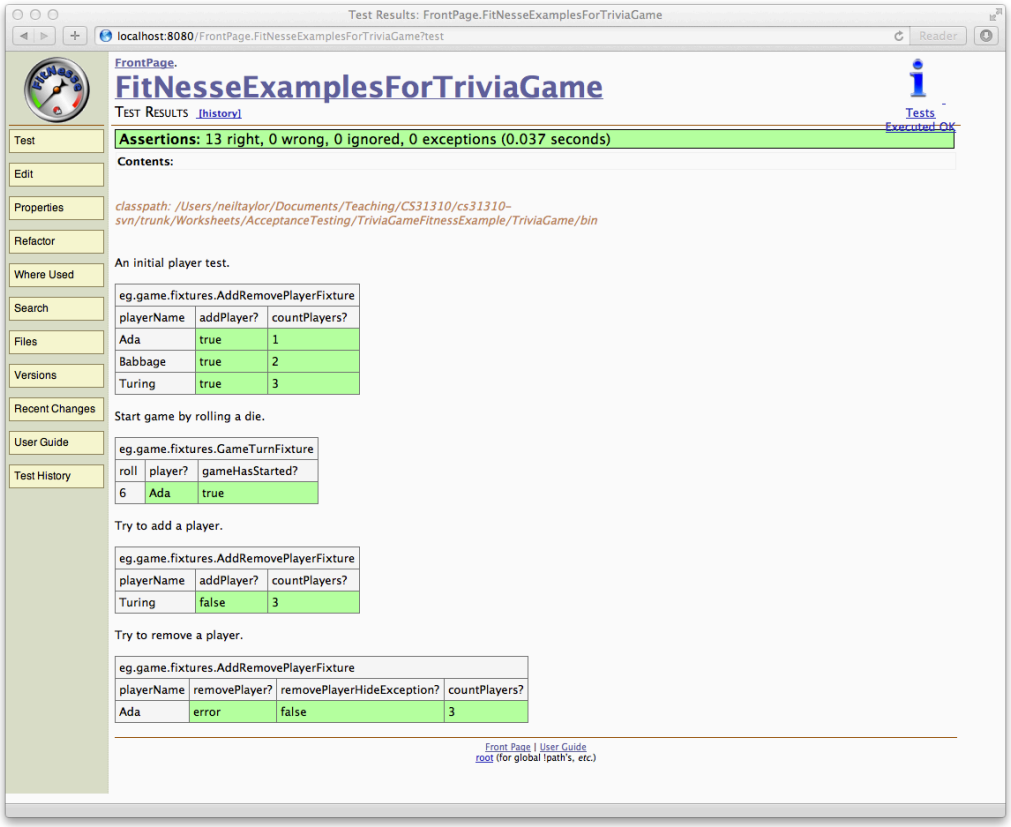


Figure 3 Test Page using Column Fixtures

You will probably find that there is no *Test* button on the page that you are editing. To activate the Test option, you need to view the *Properties* option. From the properties, select the *Test* page type. Click on *Save Properties* to commit the change. You should now see a *Test* button on the page. Click on this button to run the tests.

In the final test shown in Figure 3, you can see the structure:

- *playerName* – this is data that is passed through to the test fixture.
- *removePlayer*? – this is a method that is executed. If an exception is raised, then it is thrown from the method.

- *removePlayerHideException*? – this is a method that is executed. If an exception is raised, the method traps the error and returns true or false.
- *countPlayers*? – this is a method that is executed.

The addPlayer and removePlayer methods in Game will throw exceptions if there is an attempt to add or remove players once a game has begun. This raises the question of how we can manage that in the FitNesse tables and the test fixtures.

The *removePlayer?* Option uses a special FIT keyword *error* in the response table to indicate that an error is expected. Both removePlayerHideException and addPlayer catch any exception and return false. In the removePlayer? row, change *error* to *false*. Re-run and see what happens. When writing tests, you will need to think about the best way to highlight errors and the difference between expected errors and unexpected errors.

4.4 Demonstrate a Failing Test

You should also demonstrate what happens when at least one of the tests fails; an example is shown below.

Figure 4: A failed Test

5. 'FIT is dead, long live FitNesse'

James Shore[2] has commented on the use of automated tests in a blog and also in a podcast. Have a read of the short blog post and then listen to the podcast at the following links.

http://www.jamesshore.com/Blog/The-Problems-With-Acceptance-Testing.html

http://www.hanselminutes.com/151/fit-is-dead-long-live-fitnesse-with-ward-cunningham-and-james-shore

[2] James Shore is one of the authors of *The Art of Agile Development*.

6. Write a report

Finally, create a document that contains screenshots of your work to create test tables. Don't forget, your test tables should contain your own data items.

In your report, write a few paragraphs to consider the usefulness of tools such as FitNesse to help create and run acceptance tests. Is this useful as a way to capture documentation about the business logic? Does it complement the documentation from the code and the tests, e.g. as shown in Figure 5?

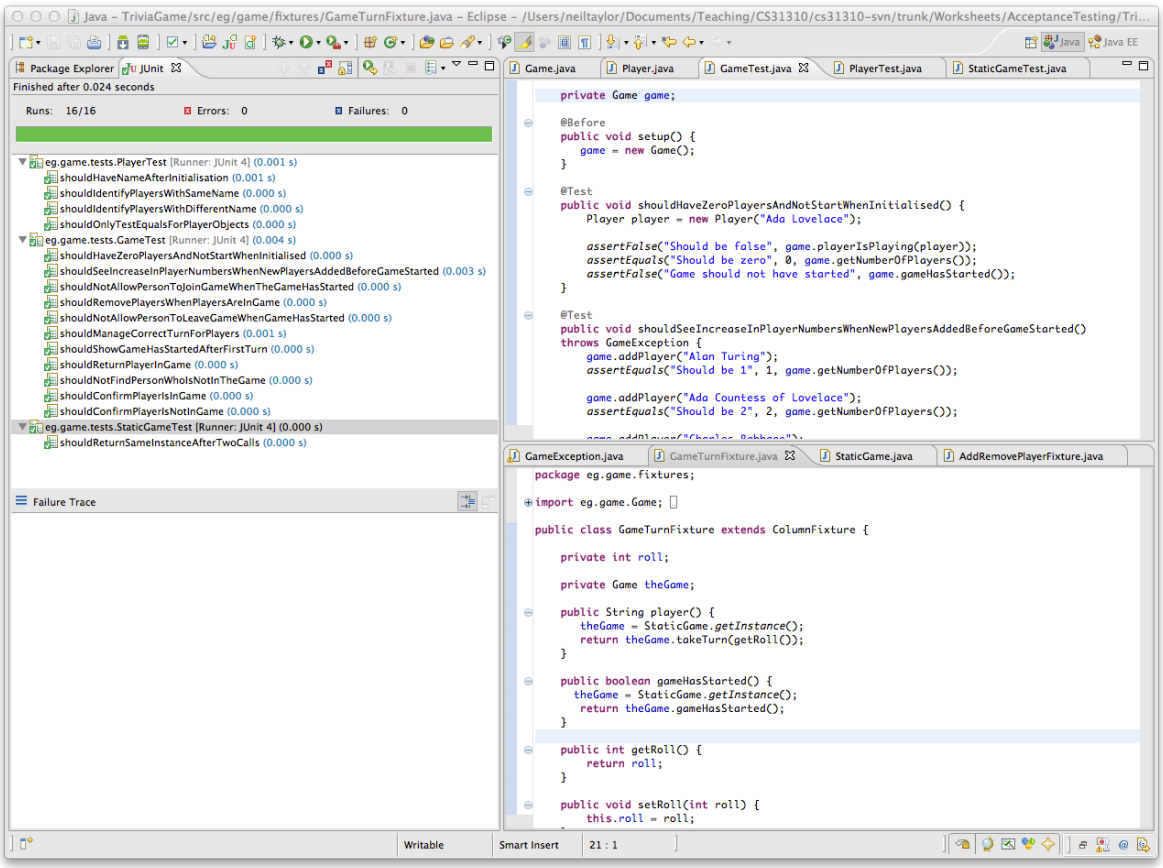


Figure 5: JUnit Tests

7. Upload Report

You should upload your report to Blackboard by **6pm Monday 12th November**. The report should be a single PDF file – no other formats will be accepted.

Marking

The worksheet is worth 3% of the module. You are being assessed on whether you attempt the worksheet and submit a sensible report with a considered view on the usefulness of such tools.

Questions?

If you have any questions about this worksheet, contact Neil Taylor (nst@aber.ac.uk).