COMP 3111: Software Engineering

HKeInvest Sprint 3: Unit Testing

Unit Testing Requirement

Construct unit tests for all the public methods of the classes that your team uses in your web application.

Notes About Unit Testing an ASP.NET Web Application

1. Unit Testing a Web Form's Code-behind File

It is not possible to create unit tests for event handler's or other methods in a web form's code-behind file. Therefore, you should structure your code so that as much of the code as possible is in public methods in classes so that it can be unit tested. Note that, in general, it is not possible to use web controls, such as TextBox and GridView in a unit test. Therefore, you should not pass such controls as parameters to your class methods since you cannot pass them as parameters in the test method.

2. Testing Methods That Accesses a Database

Technically, if you test a method that accesses a "live" database is no longer considered to be a unit test, but an integration test because you are not testing the method in isolation and you generally cannot control the result that is returned from accessing the "live" database. Moreover, since the test project is a separate project from the web application project, the web application project itself is not run when a unit test for a method is run in the test project. Consequently, a null exception is raised if you try to access the connection string for a database in the web application since no connection strings are set.

In general, if you want to access a database when doing unit testing, you should not use the "live" database, but use a "mock" database in the test project so that you can control the result that a query returns. In general, mocking a database is a complicated process that requires specialized software.

To overcome this problem, the following workaround can be used for unit testing methods that access the ExternalDatabase.mdf and HKeInvestDB.mdf databases. (This workaround is not recommended for a real project!)

1. Create a new folder inside the App_Data folder and give it a meaningful name (e.g., TestingDatabases).
2. Copy the ExternalDatabase.mdf and HKeInvestDB.mdf databases into the folder created in the previous step.
3. Prepare the test data required for your test cases in the copies of the ExternalDatabase.mdf and HKeInvestDB.mdf databases. (Most likely you will only need to prepare data for the HKeInvestDB.mdf database as you can use the data in the ExternalDatabase.mdf database as is.)

Once the ExternalDatabase.mdf and HKeInvestDB.mdf databases to be used for testing are ready, do the following in the test project.

1. Copy the ExternalDatabase.mdf and HKeInvestDB.mdf databases that you have prepared for testing into the "bin→Debug" folder. (If the "bin" folder is not visible, select the "Show All Files" tab in the `Solution Explorer`.)
2. In the web application project, open the `Web.config` file and copy all the code starting from the tag "`<connectionStrings>`" up to and including "`</connectionStrings>`".
3. In the test project, open the "app.config" file and paste the code copied in the previous step just below the "`<configuration>`" tag.

When you run a test whose method accesses either the ExternalDatabase.mdf or the HKeInvestDB.mdf databases, it will access the corresponding databases in the test project rather than the "live" databases in the web application.

Unit Testing Grading

The grading of your unit testing will be based on the quality of your test cases and the proportion of testable code that your unit test cases cover.