Unit Testing

JDK 1.7
Grails 2.4.4
Hibernate 4.3.6.1

General Idea for Controller Testing:

- One Unit Test per Controller
- Define some sample data that will be used during testing
- Define the mocked Services and their Methods that the Controller uses/calls
- Test each Controller Action in a single Test Method

Notes:

- Note 1:
  A **Controller Action** that doesn't get returned any output data from a **Service Method** shouldn't need to worry about what the Service Method it calls does, therefore we don't need to simulate any logic inside the mocked **Service Method** as well as not needing to return anything.

- Note 2:
  assert can be used in place of == and of assertEquals.
  For example, the next 3 statements can be used interchangeably:

```
name == "John"
assert name == "John"
assertEquals("John", name)  //Recommended method
```

  However, assertEquals is the recommended way to test if two variables have the same value, as it makes for clearer testing code and reduces risk of error, like using a single = instead of ==.

- Note 3:
  When comparing Lists of any type (ex. ArrayList) using assertEquals, order matters.
  For example, the first statement is false but the second is true:

```
assertEquals([1, 2], [2, 1])    //returns false
assertEquals(["John", "George"], ["John", "George"])    //returns true
```

- Note 4:
  The parameters (params) a Controller Action has when it is called can be accessed in the respective Test Method through either of the 3 ways below:

```
params
controller.params    //Recommended method
controller.params    //Recommended method
this.params
```

However, controller.params is the recommended way to reference params, as it makes for clearer testing code.

- Note 5:
  When using assertEquals always put the Expected Value as the first parameter and the Value You Are Checking as the second.
  Example:

```
String name = "John Smith"
Customer customer = new Customer(name)
assertEquals(name, customer.getName())
```

  Although it doesn't produce any different example, it makes for clearer testing code and help during debugging.

- Note 6:
  If you want to achieve Full Code Coverage with the test (meaning: test all scenarios / every piece of code) and there are if/else/switch in your Method/Action, you need to make one Unit Test per possible "branch/path".
  For example:

```
boolean flag = true
if(flag)
{
    //doSomething...
}
else
{
    //doSomethingElse...
```