CS580 Advanced Software Engineering
Assignment 6 - Unit & Integration Test

Due Date
Monday, November 17, 2014

Score
20

Questions and Directions

In this assignment, you will be performing both unit test and integration test for some of the features you have already implemented in iphoto-web.

1. Unit Test

Please create a jUnit test class in `src/test/java` project folder to test the all the functions related with photo management and filtering (total 5 features). If you have created the interface/class such as `PhotoManager`, you should perform the unit test again this class directly with a test class. If you put all the code inside the `WebController`, just do the test against the `WebController` directly.

Since most of the features related with photos are implemented locally in the disk, it really eases the testing process, which means that you probably do not have to use Mockito to do the unit test. However, depending how you implemented these features, you might need to use Mockito to simplify the testing process. Therefore, using Mockito is not required, although strongly recommended to try.

Make sure you have the right dependencies in your `pom.xml`:
for jUnit:

```
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
</dependency>
```

for Mockito:

```
<dependency>
      <groupId>org.mockito</groupId>
      <artifactId>mockito-all</artifactId>
      <version>1.9.5</version>
</dependency>
```

You need to design strong and effective test cases. We do not require any of the coverage criteria (e.g., path coverage, condition coverage). However, try to make your tests reasonable and effective in verifying errors or bugs.

You should be able to run all the unit tests in both Eclipse and Maven command line. I will be running `mvn verify` to check the status of your unit tests.

2. Integration Test

Similar to Question 1, you also need to create integration tests for the same set of the features - photo management and photo filtering (total 5 tests).

Basically, you will need to create a `WebControllerIntegrationTest` class in the test folder and perform integration tests against each HTTP API.

You should enable the automatic setup of the server before running the integration tests and automatic shutdown of the server after all the tests. The easiest way to do it is to use the same approach as we showed in the class with Spring Boot Integration Test framework (http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html)

Again, the integration tests should be runnable in both Eclipse and Maven command line. I will be running `mvn verify` to check the status of your integration tests.

Getting Help

Please let me know if you need to meet to discuss any problems that you may have.

"Software testers do not make software; they only make them better."