Existing Test Cases

Introduction

Spring Boot currently has 794 test cases, all of which pass when run on the project. These test cases utilize Junit and Mockito to check for correct behaviors of various context, json, system, utility, and web functionalities of the system.

Test Case 1: context-AnnotatedClassFinderTests

This test case checks that the AnnotatedClassFinder class within Spring Boot is working as expected. This is an important class since it returns the first class that is described with a specific annotation. This is an interesting test case since it provides further insight into how this important class should actually behave. The class itself can be ran on a specific package or a different class. The class or package must have a configuration associated with it. If it is not previously configured, the called method will throw an exception since it will recognize that the given package or class is actually null. This test case stresses the importance of using annotations and configurations in the project.

Test Case 2: json-DuplicateJsonObjectaContextCustomizeFactoryTests

This test case checks how Spring Boot behaves if two json objects are created on one class path. This is an interesting class because we had originally assumed that the original json object would just be overwritten if a second one is created. However, the opposite actually happens. The project actually ignores the second json object and throws an exception stating that it “found multiple occurrences of org.json.JSONObject on the class path.” This is an important behavior to note since json objects are important for the overall functionality of the project.

Test Case 3: context-bootstrap- SpringBootTestContextBootstrapperTests

This test case is interesting because it is actually used as an extra check for other test cases written to test Spring Boot’s bootstrap functionality. Bootstrap compatibility is another important functionality in Spring Boot. This test case was written to ensure that other bootstrap test cases are utilizing a mock web configuration object rather than an actual instance of the object. This ensures that any issues found by the test cases are most likely due to the behavior of bootstrap, rather than the web configuration itself. The use of mocking in these test cases proves that our project relies on a dependency between bootstrap and web configurations.