
FIT5171

Project Assignment 1

Test Planning, System Setup, and

Code Understanding & Extension

Student Name ID

Student Name ID

Student Name ID

1 Test Plan Identifier

This test plan is provided by the group 34.

2 Introduction

This report is about the project which is a Java-based Web application related to rockets. Firstly is the test strategy, it introduces the type of test for this project, the testing tools, and approach. Secondly is about the process of the test environment setup. The last part is testing details to show the code. The purpose of this test is to best understand the construction of a test project and this part focuses on the testing planning and basic testing.

2.1 Test Item

The item is a Java-based Website project:
<http://www.allaboutrockets.com>

2.2 The scope and overview of the testing

The testing will be approved and reviewed by the CTO. For the whole testing continue about two weeks.

2.1 Test Tools, techniques, methods

2.1.1 Tools

- JDK8: This testing is for a Java web project, the kernel is based on JDK version 8.
- IntelliJ: The major IDE development tool. Testing will be developed and run on this IDE.
- Maven: The tool which is can be used for managing and building the Java project (Maven.apache.org, 2019)
- GitHub: provide a repository for sharing and update the document for the group.

2.1.2 Techniques and methods

Junit5: The major testing frameworks.

Test-Driven Development(TTD): is the major method which

was being used in the testing and new function development.

3 Environment and Development tools Setup

All the tools download from the official website and install them into the computer.



For the IntelliJ IDE should select the JDK version, normally, IntelliJ will use the default version for the JDK which was installed with the IDE (Figure1). Such as:

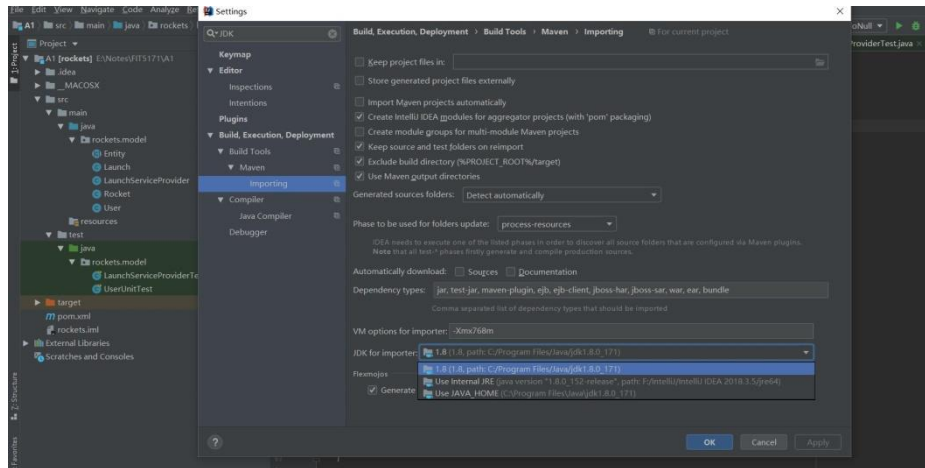


Figure1

For this project's testing has been set up to use the Maven management tool so for the IntelliJ IDE also need to set up the version of Maven(Figure2), and completed Maven configuration. (Figure3)

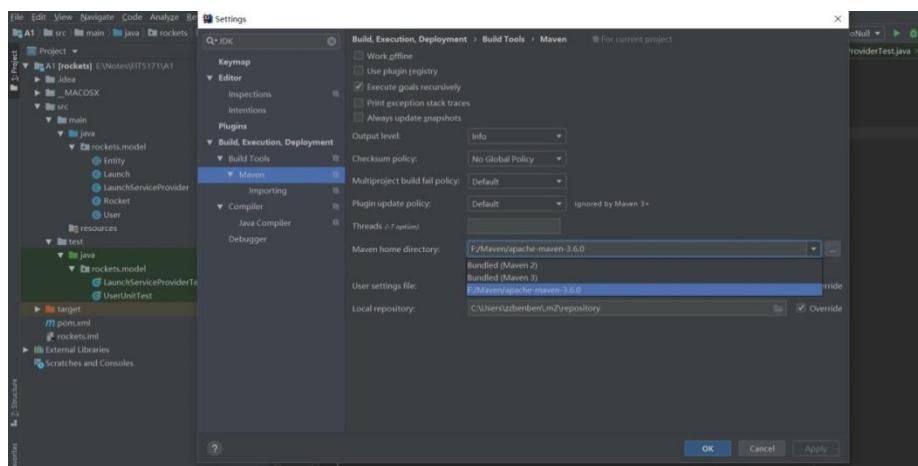


Figure2

Github as one of the tools to help the team update and sharing the testing file should be configured too in the IntelliJ IDE. Register an account and install the GitHub, then configured it in the File->Setting-> search the Git (Figure3). After the last step, changes the Github and enters the username and password.

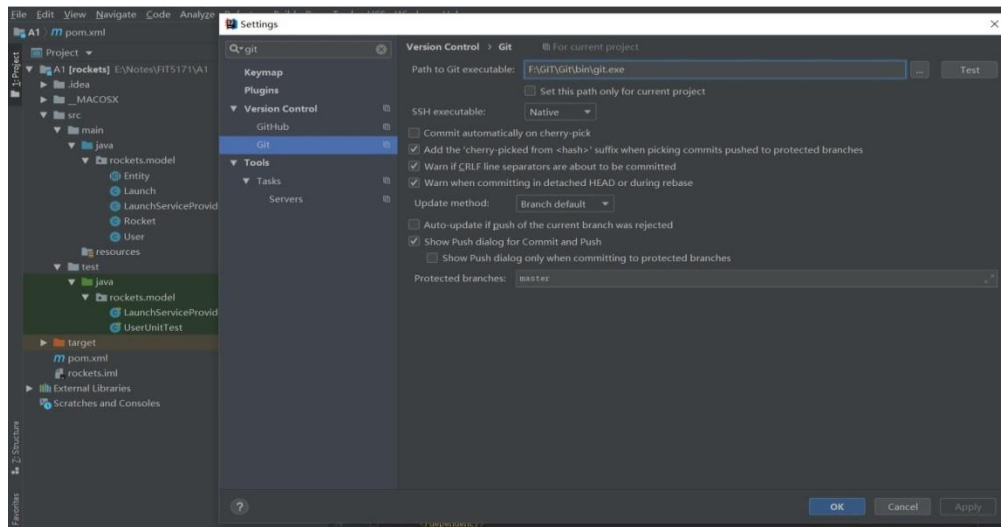


Figure3

4 Testing Details

4.1 Testing Type.

Type: Function testing

Manual/Automated: Automated testing.

4.2 Testing object

The object of this testing includes three parts: The Rocket class, The LaunchServiceProvider class, and the User class.

Rocket Class(Testing)		
Attribute	Testing	Method
MassToLEO	Not Null	NullPointerException exception = assertThrows(NullPointerException.class, () -> target.setMassToLEO(null));
MassToGTO	Not Null	NullPointerException exception = assertThrows(NullPointerException.class, () -> target.setMassToGTO(null));
MassToOther	Not null,	NullPointerException exception = assertThrows(NullPointerException.class, () -> target.setMassToOther(null));

For the Rocket class's testing is focused on these three attributes which were listed in the above table. If past the null value will throw the exception. We also add the constraints into the original class for these attributes. Another attribute Name, Country and Manufacturer had added

the constraints not null in the original class.

Add the constraints for the original Rocket class	
Attribute	constraints
massToLEO	notNull(massToLEO, "Mass cannot be null");
massToGTO	notNull(massToGTO, "Mass cannot be null");
massToOther	notNull(massToOther, "Mass cannot be null");

For the LaunchServiceProvider class, testing is based on the constructor of the class. For each attribute, testing them not null, if past the null value will throw the exception. Except for the attribute YearFounded was testing for the minimum year that can accept. When pass yearFounded less than 1900 will throw the exception.

LaunchServiceProvider(Testing)		
Attribute	Testing	Method
Name	Not Null	target = new LaunchServiceProvider(nullName, yearFounded, country); assertNull(nullName, "true");
Country	Not Null	target = new LaunchServiceProvider(name, yearFounded, nullCountry); assertNull(nullCountry, "true");
YearFounded	year(year<1990)	target = new LaunchServiceProvider(name, minusYearFounded, country); assertTrue(minusYearFounded < 1900);
headquarters	Not Null	target = new LaunchServiceProvider(name, yearFounded, country); NullPointerException exception =assertThrows(NullPointerException.class, () -> target.setHeadquarters(null)); assertEquals ("headquarters cannot be null or empty", exception.getMessage());

The constraints for the headquarters cannot be null also had added in the original class.

Add the constraints for the original Rocket class	
Attribute	constraints
headquarters	notBlank(headquarters, "headquarters cannot be null or empty"); this.headquarters = headquarters;

In this class, there will be having more than three type situations such as two attributes as the null value. But consider the first part testing of this class and the constraints which are each one of the attributes not allowed to be null, so the situation that two attributes as null cannot happen.


For the User class, the testing of the Email and Password attribute had been provided by an example. We add the constraints into the original class for these two attributes.

Add the constraints for the original User class	
Attribute	constraints
Email	notBlank(email, "email cannot be null or empty");
Password	notNull(password, "password cannot be null or empty");

5 Extra Credits

This part we add some attribute in to some original class to meet the requirement, all the attribute which we had added shows in the below table.

Attribute or Method	Class
String family;	Rocket
String series;	Rocket
String rocketCode;	Rocket
Map<String,Rocket> rocketmap	LaunchServiceProvider
addRocketToGroup(Rocket rocket)	LaunchServiceProvider

For the Rocket class we add the three attribute for each Rocket. The collection of the rocket was provided by the original class. As the requirement, each rocket may belong to a family. So we create a Mapping  for each rocket of each family and use the **addRocketToGroup** function to separate the different rocket into different group (family).We also add the constrains and testing method for the extra credits requirement.

Add the constraints for extra credit's requirement attribute	
Attribute	constraints
family;	notNull(family,"Family cannot be null");
series;	notNull(series," Series cannot be null");
rocketCode;	notNull(rocketCode," RocketCode cannot be null");

Payload(Testing)			
Attribute	Testing	Method	Class
Payload	Input Limits	shouldThrowExceptionWhenSetPayloadsAreNotSa tellitesOrSpacecrafts();	LaunchU nitTest

For Payload testing, we use the while loop to iterator the collection of payload so that check reasonable of the input value for payload attribute.

6 Team member's Contributions

Su: RocketTest

Victor: LaunchServiceProviderUnitTest and Extra Credit

Kaixiang: Report