Linnéuniversitetet
Kalmar Växjö

Report

Testing

Test planning

Author: Helena Tevar Hernandez
Handledare: Jesper Andersson
Semester: Spring 2017
Discipline: Software Technologies
Course code: DV600

Table of Contents

1 JUnit classes

Planning for the unit test of the three classes used in the library.

1.1 Summary

J-01	J-02	J-03	J-04	J-05
PASSED	PASSED	PASSED	PASSED	PASSED
J-06	J-07	J-08	J-09	J-10
PASSED	PASSED	PASSED	PASSED	FAIL

1.2 Test planning

1.2.1 Book class

Test id	J-01	*Test name*	Book Get/Set
Requirement			
Use case			
Scenario	Create Junit to test the class Book		
Pre-condition	Create an empty book and a test strings for all the variables inside the book.class and create a test for every get/set		
Post-condition	The get/set test pass		
Notes	Create a test for each get/set. We don't need more than one if we use a setter to set a string and a getter to assert that is equals to our string. Both, get and set, are tested at the same time.		
Expected result	Passed Junit test		
Actual result	Passed Junit test		
Date	July 2017	*Status*	DONE

Test id	J-02	*Test name*	Book toString
Requirement			
Use case			
Scenario	Create Junit to test the class Book		
Pre-condition	Create an empty book and a test string for all the variables inside the book.class and create a test that compares the test string with our book		
Post-condition	JUnit test completed.		
Notes			
Expected result	Passed Junit test		
Actual result	Passed Junit test		
Date	July 2017	*Status*	DONE

1.2.2 Catalog class

Test id	J-03	*Test name*	Catalog Get/Set
Requirement			
Use case			

Scenario	Create a Junit to test the class catalog getters and setters		
Pre-condition	Create a catalog in junit with different mocked list, mapper and books that will be used during the tests.		
Post-condition	JUnit test completed.		
Notes	During this test, mock different object to make the test easier to prove.		
Expected result	Set the catalog list of books with a mocked list. Assert that the mocked list is the same that you get with your catalog.		
Actual result	Test passed.		
Date	July 2017	*Status*	DONE

Test id	J-04	*Test name*	Catalog toJson
Requirement			
Use case			
Scenario	Create a Junit test for the method toJson from the class catalog.		
Pre-condition	Junit test created for the catalog.class.		
Post-condition	JUnit test completed.		
Notes	Create a list with one book and a Json string that matches the book.		
Expected result	The Json string and the method catalog.toJson() matches.		
Actual result	Test passed.		
Date	July 2017	*Status*	DONE

Test id	J-05	*Test name*	Catalog get A book
Requirement			
Use case			
Scenario	Create a JUnit test to get a book with an specific id.		
Pre-condition	Junit test created for the catalog.class.		
Post-condition	JUnit test completed.		
Notes	Mock the list used by the catalog so when we ask for a book, it will return our mocked book.		
Expected result	The book asked with an specific id matches our mocked book.		
Actual result	Test passed.		
Date	July 2017	*Status*	DONE

Test id	J-06	*Test name*	Catalog get A book null
Requirement			
Use case			
Scenario	Create a test that searchs a book with an specific but inexistent id.		
Pre-condition	Junit test created for the catalog.class.		
Post-condition	JUnit test completed.		
Notes	We mock the list and books, so when we ask the single book we have for its id, it will NOT match our id.		
Expected result	The method will not find the book asked and will return a null book.		
Actual result	Test passed.		
Date	July 2017	*Status*	DONE

Test id	J-07	*Test name*	Catalog set A book
Requirement			
Use case			
Scenario			
Pre-condition	Junit test created for the catalog.class.		
Post-condition	JUnit test completed.		
Notes	We create two Json strings. We will add the first string to create a book in our catalog and the second to modify it.		
Expected result	The second string will modify the first one.		
Actual result	Error found: ==> default: models.TestCatalog > testSetABook FAILED ==> default: org.codehaus.jackson.map.JsonMappingException: Can not deserialize instance of lnu.models.book out of START_ARRAY token When testing the API as an user, the program works. When using an Junit class, we get the error above. Fixed: The Json string was not correctly written.		
Date	July 2017	*Status*	DONE

Test id	J-08	*Test name*	Catalog add a Book
Requirement			
Use case			
Scenario			
Pre-condition	Junit test created for the catalog.class.		
Post-condition	JUnit test completed.		
Notes	Use our mocked book to add it to a empty catalog and ask for the list.		
Expected result	Error found: ==> default: models.TestCatalog > testSetABook FAILED ==> default: org.codehaus.jackson.map.JsonMappingException: Can not deserialize instance of lnu.models.book out of START_ARRAY token When testing the API as an user, the program works. When using an Junit class, we get the error above. Fixed: The Json string was not correctly written.		
Actual result			
Date	August 2017	*Status*	DONE

1.2.3 BooksDAO class

Test id	J-09	*Test name*	DAO Get/Set
Requirement			
Use case			

Scenario	Creating two JUnit test with same pattern for getters and setters of the variable catalog and list inside booksDAO.		
Pre-condition			
Post-condition	JUnit test completed.		
Notes	Create a mocked catalog to create a DAO. Repeat the same pattern with the list.		
Expected result	Create a DAO and set its catalog with a mocked one. Then compare the mocked catalog with the DAO.getCatalog() Set the list with the setter and compare it with the getter result.		
Actual result	Tests passed		
Date	August 2017	*Status*	DONE

Test id	J-10	*Test name*	DAO Write/Read XML
Requirement			
Use case			
Scenario	Create a JUnit test for the methods read and write XML.		
Pre-condition			
Post-condition	Junit test completed		
Notes	We create a simple list with a simple book as dummies. We use the method rewrite with our simple list. After that, create a list that will include what the method booksDAO.booksFromXML() returns.		
Expected result	Test passed.		
Actual result	The method booksDAO.rewrite(List<book>) actually rewrites the XML and the method booksDAO.booksFromXML() actually returns a List of books, but for some reason, the assert equals fail. The tester is unable to understand and solve this problem because both list looks the same. *==> default: java.lang.AssertionError: expected: java.util.ArrayList<[Book: Titletest - Author: Authortest - ID: 1 - Publish Date: 2017-04-10 - Genre: Genretest - Description: Descriptiontest]> but was: java.util.ArrayList<[Book: Titletest - Author: Authortest - ID: 1 - Publish Date: 2017-04-10 - Genre: Genretest - Description: Descriptiontest]>* *==> default: at org.junit.Assert.fail(Assert.java:88)*		
Date	August 2017	*Status*	Failed

2 API Testing

Test made by user interation with the library.

2.1 Summary

BAPI-01	BAPI-02	BAPI-03	BAPI-04	BAPI-05	BAPI-06
PASSED	PASSED	PASSED	PASSED	PASSED	PASSED

2.2 Test planning

Test id	BAPI-01	*Test name*	Beta Testing Fetching Books
Requirement	The user can see a list of books.		
Use case	Get Books.		
Scenario	The user click over "/books" and is able to see a list of books.		
Pre-condition	A list of books is created previously.		
Post-condition	The list of books shows in the screen.		
Notes			
Expected result	A list of books in the screen.		
Actual result	Same		
Date	August 2017	*Status*	DONE

Test id	BAPI-02	*Test name*	Beta Testing Feching Books Empty
Requirement	The user is unable to see a list of books.		
Use case	Get Books.		
Scenario	The user clicks over "/books" but no books are displayed.		
Pre-condition	Empty list of books.		
Post-condition	The server shows no books in the screen.		
Notes			
Expected result	No books in "/books"		
Actual result	Same		
Date	August 2017	*Status*	DONE

Test id	BAPI-03	*Test name*	Beta Testing Adding Books
Requirement	The user is able to add books to the catalog through the web.		
Use case	Add Books		
Scenario	The user clicks on "add" and the add books form shows up. The user fills the form and is able to add that book to the showed list.		
Pre-condition	The book added does not exist already in the catalog.		
Post-condition	The book is added to the catalog.		
Notes			
Expected result	The book is added and showed in the list.		
Actual result			
Date	August 2017	*Status*	DONE

Test id	BAPI-04	*Test name*	Beta Testing Adding Books Same
Requirement	The user is unable to add books to the catalog through the web.		
Use case	Add Books		
Scenario	The user clicks on “add” and the add books form shows up. The user fills the form and is unable to add that book to the showed list.		
Pre-condition	The book added exist already in the catalog.		
Post-condition	The book is not added to the catalog.		
Notes			
Expected result	The book is already added and showed in the list.		
Actual result			
Date	August 2017	*Status*	DONE

Test id	BAPI-05	*Test name*	Beta Testing Edit Book
Requirement	The user is able to edit books to the catalog through the web.		
Use case	Edit Books		
Scenario	The user clicks on “edit” and the edit books form shows up. The user fills the form and is able to edit that book to the showed list.		
Pre-condition	There is books to edit in the catalog.		
Post-condition	The book is edited to the catalog.		
Notes			
Expected result	The book should be modified and showed in the list.		
Actual result	The book is modified and shows in the list.		
Date	August 2017	*Status*	DONE

Test id	BAPI-06	*Test name*	Beta Testing Delete Book
Requirement	The user is able to delete books to the catalog through the web.		
Use case	Delete Books		
Scenario	The user clicks on “delete” and the books form shows up. The user is able to delete that book to the showed list.		
Pre-condition	There is books to delete in the catalog.		
Post-condition	The book is deleted to the catalog.		
Notes			
Expected result	The book should be deleted and not showed in the list.		
Actual result	The book is deleted and does not show in the list.		
Date	August 2017	*Status*	DONE

2.3 JUnit test

It was planned to make a Junit test of the API using the response of the methods, but until now the tester needs more formation and refatorization to understand and implement a test with that in consideration.

Test id	JAPI-01	*Test name*	Junit Get Books
Requirement			
Use case			
Scenario	Create a JUnit test for the resource GetBooksResource.		
Pre-condition	Create a mocked resource, with a simple list.		
Post-condition	The resource answers a Json string.		
Notes	Needs testing response 404.		
Expected result	The test should pass.		
Actual result	Test passed		
Date	August 2017	*Status*	ONGOING

Test id	JAPI-02	*Test name*	Junit Add Book
Requirement			
Use case			
Scenario	Create a JUnit test for the resource AddBookResource		
Pre-condition	Create a mocked resource, with a empty list.		
Post-condition	The resource adds a book and answers response 200.		
Notes	Needs testing response 404.		
Expected result	The test should pass.		
Actual result	Test passed		
Date	August 2017	*Status*	ONGOING

Test id	JAPI-03	*Test name*	Junit Modify Book
Requirement			
Use case			
Scenario	Create a JUnit test for the resource EditBookResource		
Pre-condition	Create a mocked resource, with a list with a book to be modified.		
Post-condition	The resource edits a book and answers response 200.		
Notes	Needs testing response 404.		
Expected result	The test should pass.		
Actual result	Test passed		
Date	August 2017	*Status*	ONGOING

Test id	JAPI-04	*Test name*	Junit Remove Book
Requirement			
Use case			
Scenario	Create a JUnit test for the resource RemoveBookResource		
Pre-condition	Create a mocked resource, with a list with a book to be deleted.		
Post-condition	The resource deletes a book and answers response 200.		
Notes	Needs testing response 404.		
Expected result	The test should pass.		
Actual result	Test passed		
Date	August 2017	*Status*	ONGOING