

# **Integrity Setup Testing with FitNesse**

## **Documentation**

V1.0

**Volker Eckardt**  
**veckardt@ptc.com**



## Content

<b>Introduction</b>	<b>3</b>
<b>Setup Test with FitNesse</b>	<b>3</b>
<b>Execute FitNesse Tests</b>	<b>4</b>
<b>Statistic</b>	<b>6</b>

## Introduction

The approach to test web based applications with automated tools is very common. In this document we describe how we used FitNesse to test the Integrity setup.

The main achievements are:

- Standardized Integrity Setup Test
- Including:
  - Item creation and update, validate state changes
  - relationship creation
  - and more

## Setup Test with FitNesse


FitNesse allows you to define test cases in a very comfortable way. Mainly you use a wiki to organize your tests, and a build in functionality to execute them.

### Define FitNesse Tests

To define the tests, create a table based structure and place your commands right there. Each input value usually gets a dedicated column, each output value is represented by a name with a question mark.

The class which is used to execute the commands is listed right above the table, in this case it's the class integrityAPITest.integrity.

You can use variables to store results and use them in other rows. In this case we use IDTP as a placeholder for the Test Plan-ID, and ask for the current state via viewissue in the second row in section 2.



FrontPage

## TestPlan

---

**Contents:**

*variable defined: TEST\_SYSTEM=slim*  
*classpath: D:\Users\VECKARDT\Documents\NetBeansProjects\IntegrityAPITest\dist\IntegrityAPITest.jar*

**1. Initialization - Find Items already there, to use them later**

Search for the project item with Summary "Project for Release 1" and return the ID

integrityapitest.integrity		
Type	Summary	findIssue?
ALM_Project	Project for Release 1	\$IDP=

**2. Positive Tests - Create and Check the Status for three different Test Items**

We try to create 3 test items, and return via viewissue the current states.

integrityapitest.integrity										
Module	Command	Type	Summary	Description	Project	fieldName	Selection	Result?	Message?	Value?
im	createissue	ALM_Test Plan	plan 1	descr				\$IDTP=		
im	viewissue					state	\$IDTP	0		ALM_Submit
im	createissue	ALM_Test Objective	objective 1	descr	/ALM_Projects/Release1			\$IDTO=		
im	viewissue					state	\$IDTO	0		ALM_Proposed
im	createissue	ALM_Test Suite	suite 1	descr	/ALM_Projects/Release1			\$IDTS=		
im	viewissue					state	\$IDTS	0		

### 3. Positive Tests - Edit and View the Test Plan

We edit the test plan item, roll the status forward, and add the needed relationships such as project and test object. Test Object is mandatory as soon as we reach the "In Progress" state. Finally, we validate that the item has still the expected values in "type" and "description".

integrityapitest.integrity								
Module	Command	State	addRelationships	fieldName	Selection	Result?	Message?	Value?
im	editissue	ALM_In Review	ALM_Test Plan For:\$IDP		\$IDTP	0		
im	editissue	ALM_In Progress			\$IDTP	-1		
im	editissue	ALM_In Progress	ALM_Test Objectives:\$IDTO		\$IDTP	0		
im	viewissue			Type,Description	\$IDTP	0		ALM_Test Plan,descr
im	viewissue				\$IDTP	0		>1000

### 4. Negative Tests - Just to show how it works

In both cases we assume an error message from the api. Therefore, the expected result is -1.

integrityapitest.integrity								
Module	Command	Type	Summary	Description	addRelationships	Selection	Result?	Message?
im	createissue	ALM_Project Plan	summ	descr			-1	
im	editissue				ALM_Test Plan For:101	1161	-1	

### 5. Clean Up - Delete Issues just created


Just to keep the database size small, and remove the test data immediately after testing.

integrityapitest.integrity				
Module	Command	Selection	Result?	Message?
im	deleteissue	\$IDTP	0	
im	deleteissue	\$IDTO	0	
im	deleteissue	\$IDTS	0	

## Execute FitNesse Tests

Executing FitNesse tests can be achieved by simply clicking the test button. Automatically FitNesse starts and performs the test from top to bottom. Each row will be executed, and the results tracked and marked with colors.

Such a test result may look like this:



[FrontPage](#)  
**TestPlan**

Output Captured
Test
Edit
Add
Tools

**Failure Navigator**  
< 1 of 2 >

**Assertions:** 22 right, 2 wrong, 16 ignored, 0 exceptions (2.022 seconds)

**Contents:**

variable defined: TEST\_SYSTEM=slim  
classpath: D:\Users\VECKARDT\Documents\WetBeansProjects\IntegrityAPITest\dist\IntegrityAPITest.jar

**1. Initialization - Find Items already there, to use them later**

Search for the project item with Summary "Project for Release 1" and return the ID

integrityapitest.integrity		
Type	Summary	findIssue?
ALM_Project	Project for Release 1	\$IDP<- [101]

**2. Positive Tests - Create and Check the Status for three different Test Items**

We try to create 3 test items, and return via viewissue the current states.

integrityapitest.integrity										
Module	Command	Type	Summary	Description	Project	fieldName	Selection	Result?	Message?	Value?
im	createissue	ALM_Test Plan	plan 1	descr				\$IDTP<- [1353]	Item with ID '1353' has been created	BLANK
im	viewissue					state	\$IDTP-> [1353]	0	Command 'im viewissue': ok	ALM_Submit
im	createissue	ALM_Test Objective	objective 1	descr	/ALM_Projects/Release1			\$IDTO<- [1354]	Item with ID '1354' has been created	BLANK
im	viewissue					state	\$IDTO-> [1354]	0	Command 'im viewissue': ok	ALM_Proposed

im	createissue	ALM_Test Suite	suite 1	descr	/ALM_Projects/Release1			\$IDTS-<[-1]	Could not save item: MKS124312: Can not create an Item of Type ALM_Test Suite.	BLANK
im	viewissue					state		\$IDTS->[-1]	[-1] expected [0] Item -1 does not exist.	BLANK

### 3. Positive Tests - Edit and View the Test Plan

We edit the test plan item, roll the status forward, and add the needed relationships such as project and test object. Test Object is mandatory as soon as we reach the "In Progress" state. Finally, we validate that the item has still the expected values in "type" and "description".

integrityapitest.integrity									
Module	Command	State	addRelationships	fieldName	Selection	Result?	Message?	Value?	
im	editissue	ALM_In Review	ALM_Test Plan For:\$IDP->[101]		\$IDTP->[1353]	0	Command 'im editissue': ok	BLANK	
im	editissue	ALM_In Progress			\$IDTP->[1353]	-1	Could not save modified item 1353: MKS124147: The following fields which are mandatory in the state ALM_In Progress have not been filled in: Test Objectives	BLANK	
im	editissue	ALM_In Progress	ALM_Test Objectives:\$IDTO->[1354]		\$IDTP->[1353]	0	Command 'im editissue': ok	BLANK	
im	viewissue			Type,Description	\$IDTP->[1353]	0	Command 'im viewissue': ok	ALM_Test Plan,descr	
im	viewissue				\$IDTP->[1353]	0	Command 'im viewissue': ok	1353>1000	

### 4. Negative Tests - Just to show how it works

In both cases we assume an error message from the api. Therefore, the expected result is -1.

integrityapitest.integrity									
Module	Command	Type	Summary	Description	addRelationships	Selection	Result?	Message?	
im	createissue	ALM_Project Plan	summ	descr			-1	Could not save item: MKS124066: Type "ALM_Project Plan" does not exist.	
im	editissue				ALM_Test Plan For:101	1161	-1	Could not save modified item 1161: MKS124314: Relationship from 1161 -> 101 already exists for field Test Plan For	

### 5. Clean Up - Delete Issues just created

Just to keep the database size small, and remove the test data immediately after testing.

integrityapitest.integrity				
Module	Command	Selection	Result?	Message?
im	deleteissue	\$IDTP->[1353]	0	Command 'im deleteissue': ok
im	deleteissue	\$IDTO->[1354]	0	Command 'im deleteissue': ok
im	deleteissue	\$IDTS->[-1]	[-1] expected [0]	MKS124066: Item "-1" does not exist.

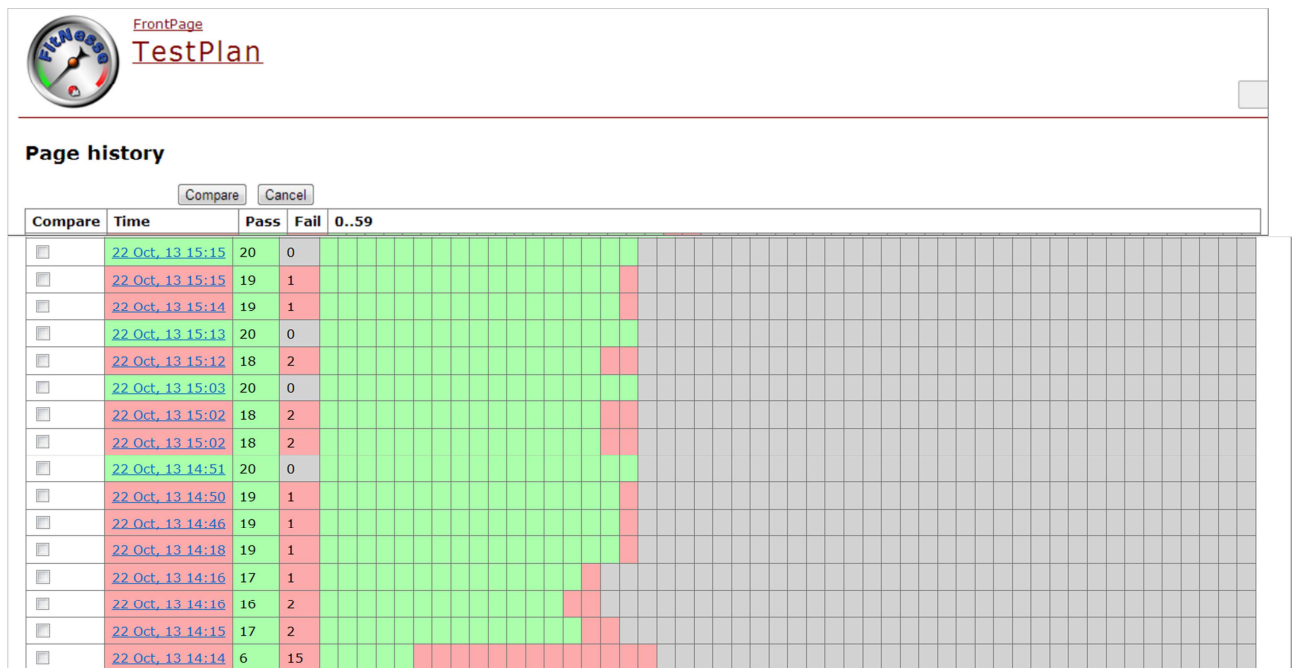
We can see that mainly all tests have been executed successfully, but two completed with errors. In fact the create issue for a test suite did not work, and therefore also the cleanup can't be performed correctly.

To validate the result, we use not only the result, but also the message and in addition the value returned from viewissue.

## Statistic

FitNesse provides a good set of statistical test information. You can also compare test session results with each other.

*Details for each session:*



*Summary view:*

