vivo-regression-test: a Test Bench Tool for the Continuous Evaluation of VIVO's Development

Sub pages

- Comparison test case scenario

Summary

1) Document Status

This document is currently being drafted

2) Perspective

The VIVO webapp is a sophisticated computer program whose evolution is guaranteed by continuous software development using an Agile software engineering methodology. In this engineering approach, the software testing process must also be managed continuously and be adaptive to new developments that occur over time. The Vivo-regression-testing tool aims to provide the software developer with a test bench integrated into its development environment.

Needs

For the IT developer, the test bench is designed to meet the following needs:

- Offer a regression test ecosystem by simulating user actions from the VIVO user interface.

- Enable regression test design
- Allow regression testing to be performed
- Provide unified tools for managing the execution of VIVO instances
- Provide an environment for running comparative tests on multiple instances of VIVO at the same time
- Running Regression Tests in Batch Mode

What we want to test:

In the context of vivo-i18n

- User interface consistency as corrections are made to the various windows
- Integrity of data in the triplestore when updating through the user interface in a multilingual context
- Ontological cohesion and user interfaces between the original version (1.11.1) of VIVO and its corollary en_US of VIVO-i18n (1.11.2-SNAPSHOT)

Usage context

The test bench usage is illustrated according to the process described below

Test case building process illustration	Description
blocked URL	**Step 1)** Selenium includes a recorder that captures the interaction between the user and the browser. The capture of the interaction is stored in the form of a Selenium script. **Step 2)** The translation of the Selenium script is translated into a JUnit Test Case to which it is necessary to make adjustments for the interpretation of the actions. **Step 3)** The Java Unit Test is suited for a generalized use in multiple language contexts. **Step 4)** The test bench encapsulates a set of features and properties that are necessary to parameterize the execution of test cases. This step aims to integrate the test case into these execution parameters. **Step 5)** Once integrated into the test bench, the test case can be triggered by the development in a singular way or in batch mode according to the needs of the moment.

---

3) Technical Notes

Where you can find the Test bench (vivo-regression-test) tool

the regression testing tool available in this github directory https://github.com/vivo-community/vivo-regression-tests

Componants architecture

The figure below shows the architecture of the VIVO-regression-test tool's main components. The architecture is divided into two parts: 1) the client-server side and 2) the client-side.

On the server side, the architecture includes a TOMCAT server running multiple instances of VIVO. The management of these instances is not properly part of this tool, it belongs mainly to the UQAM-DEV tool. On the other hand, the test bench is also an Eclipse project which can be integrated into UQAM-DEV. The availability of multiple concurrent VIVO instances ensures the development of comparative tests between different VIVO implementations (e.g., unilingual VIVO (actuel version 1.11.1)). VIVO i18n, Other VIVO configurations to be evaluated).

On the client side, each test case is an instance of both the Selenium and TestNG frameworks. Selenium consists of a robot programming interface that simulates human interactions on a web site through a web browser while TestNG is used to schedule the progress of the regression test. Each interaction through the user interface is evaluated and measured to validate the accuracy and consistency of the information produced by the VIVO instance being tested. During possible updates (addition/deletion/modification) of data, the test case can evaluate the accuracy of the data stored in the triplestore through the Apache Jena APIs. The 'Main Test Runner' is the main application that coordinates the execution of suites (aTestSuite) and test cases (aTestCase). The MainTestRunner exists in two forms. The first form starts its execution by the Maven utility and is set up by configuring the pom.xml and settings.xml files. The second form is a Java application which is parameterized by the configuration of the TestRunner.propreties file.

blocked URL

Test Bench Java Package Structures

The figure below shows the structure of the Java packages in the test bench. Each package is associated with a label describing the package's usefulness.

blocked URL

Unit Test Java Class Diagram

The figure below shows the java class structure for the *ResearchOverviewToPersonUnitTest_{$linguistic-context}* test case. The test case is particularized for each language context to be evaluated (en_CA, en_US, fr_CA and orig). Each test case is an instance of the abstract class (e.g. ResearchOverviewToPersonUnitTest.java) that contains the test sequence to be performed. Finally, the TestBenchModel class is the super abstract class from which all the test cases of the test bench are derived. This abstract class encapsulates all the functionalities and properties of the test bench. The MainTestRunner is responsible for the execution of each test case,

blocked URL

Utilities Resources Files and Java Class Diagram

The diagram below shows the utilities classes and resources files necessary for test bench operation.

blocked URL

Phases Structure Pattern of a Java TestNG Class

In general, each test case is divided into 5 phases in addition to the setUpBeforeClass et tearDownAfterClass

Five phases pattern of testCase	Description
setUpBeforeClass	• Opening Silenium Driver • Choosing the Vivo tested instance (I18n or non-i18n instance) • Purge Tripestore from instance statement (previus sample data manipulation • loading sampleData
Phase 1	• login from the UI
Phase 2	• Adding Data by UI • verify in the triplestore if the data is correctly added
Phase3	• Modifying Data by UI • verify in the triplestore if the data is correctly modified
Phase 4	• Delete Data by UI • verify in the triplestore if the data is correctly Deleted
Phase 5	• Log out
tearDownAfterClass	• Closing Selenium driver

4) Running A Test Case in the TestBench under UQAM-DEV

Prerequisite

- UQAM-DEV installed under Windows-10
- GITBash In Windows Case
- Include the appropriate Selenium driver in the ./lib directory. The current distribution contains the driver for Firefox 64 Bits under Windows 10.
- A complete installation of VIVO 1.11.1 and 1.11.2-SNAPSHOT (VIVO-i18n)

Introduction

Faire le sommaire des étapes

Step 1: Installing:

Using Windows GitBash

UQAM-DEV

This is not a mandatory step. On the other hand the references to the directories that will depend on C:/UQAM-DEV, which is its installation directory.

For complete installation, following: 2. **Installing UQAM-DEV**

OR FROM SCRATCH

Install Tomcat and SOLR as indicated in the Lyrasis documentation. **https://wiki.lyrasis.org/display/VIVODOC111x/Installing+VIVO**

Or these documents more specific to this installation:

https://wiki.lyrasis.org/display/VIVO/1%29+Starting+with+basic+installation%3A+Java-Maven-Solr-GIT

https://wiki.lyrasis.org/display/VIVO/3%29+Installing+Tomcat+8.5+for+Vivo-1.11.0-i18n

Variables configuration

For UQAM-DEV installation

```
    export REG_TEST_HOME=/c/UQAM-DEV                    # Home regression test directory
export REG_TEST_GIT=$REG_TEST_HOME/GIT              # GIT extraction directory
export JAVA_HOME=$REG_TEST_HOME/jdk                 # Java Jdk 1.8
export SOLR=$REG_TEST_HOME/solr-7.7.2               # Solr 7.7.2 installation directory
export TOMCAT=$REG_TEST_HOME/apache-tomcat-8.5.11   # Tomcat installation directory
export VIVO_HOME_I18N=$REG_TEST_HOME/vivo_i18n      # Vivo-i18n Home directory
export VIVO_HOME_ORIG=$REG_TEST_HOME/vivo_orig      # Vivo original (non-i18n) Home directory
```

For Other installation

```
    export REG_TEST_HOME=YOUR_VALUE  # Home regression test directory
export REG_TEST_GIT=YOUR_VALUE   # GIT extraction directory
export JAVA_HOME=YOUR_VALUE             # Java Jdk 1.8
export SOLR=YOUR_VALUE                          # Solr 7.7.2 installation directory
export TOMCAT=YOUR_VALUE                 # Tomcat installation directory
export VIVO_HOME_I18N=YOUR_VALUE # Vivo-i18n Home directory
export VIVO_HOME_ORIG=YOUR_VALUE # Vivo original (non-i18n) Home directory
```

Example for Linux

```
    export REG_TEST_HOME=/opt/tomcat
export REG_TEST_GIT=/opt/tomcat/GIT
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export SOLR=/opt/solr
export TOMCAT=/opt/tomcat
export VIVO_HOME_I18N=/opt/tomcat/vivo_i18n
export VIVO_HOME_ORIG=/opt/tomcat/vivo_orig

# You can also source the ./linux_var_dir.sh file
```

Step 2: Extracting the appropriate GitHub Repos

While waiting for the integration of the development branches to the Master branch, it is necessary to perform the extractions from the following repo:

Repos are retrieved from the **$REG_TEST_GIT**

```
    cd $REG_TEST_GIT
```

Repo name	GIT clone commans
Vitro	`git clone --single-branch --branch  feature-1801-regtest https://github.com/MichelHeon/Vitro.git`
Vitro-languages	`git clone --single-branch --branch  feature-1801-regtest https://github.com/MichelHeon/Vitro-languages.git`
VIVO	`git clone --single-branch --branch  feature-1801-regtest https://github.com/MichelHeon/VIVO.git`
VIVO-languages	`git clone --single-branch --branch  feature-1801-regtest https://github.com/MichelHeon/VIVO-languages.git`
Vivo-regression-test	`git clone --single-branch --branch  feature-1801-regtest https://github.com/MichelHeon/vivo-regression-tests.git`

Step 3 Installing VIVO's (orig and i18n)

Step 3 a: Installing VIVO 1.11.1 (Original VIVO) from the CLI

The installation procedure below refers to the official Lyrasis installation guide, available here.i: https://wiki.lyrasis.org/display/VIVODOC111x/Installing+VIVO

	Description	Bash command

1	Vivo Home and Vivo solr configuration
	
	from the directory **`$REG_TEST_GIT`/vivo-regression-tests/vivo-config-proposal /vivo_orig ,** copy **solr** and **vivo_home** in aprropriate directory
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-config-proposal/vivo_orig`
		`mkdir -p $VIVO_HOME_ORIG/home/config`
		`cp -r vivo_home/* $VIVO_HOME_ORIG/home/config`
		`sudo -u solr cp -r solr/server $SOLR`

2	rom the directory **C:\UQAM-DEV\GIT\vivo-regression-tests**
	
	Installing VIVO 1.11.1 original
		For UQAM-DEV installation
		
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-installer-orig`
		`mvn install -s UQAM-DEV-RgTest_orig_settings.xml`
		
		For OTHER installation
		
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-installer-orig`
		`# Edit and change attributes values to appropriate settings of`
		`# NON_UQAM-DEV-RgTest_orig_settings.xml file`
		`vi NON_UQAM-DEV-RgTest_orig_settings.xml`
		`mvn install -s NON_UQAM-DEV-RgTest_orig_settings.xml`

Step 3 b: Installing VIVO 1.11.2-SNAPSHOT (VIVO-i18n) from the CLI

	Description	Bash command

1	Vivo Home and Vivo solr configuration for i18n
	
	from the directory **`$REG_TEST_GIT`/vivo-regression-tests/vivo-config-proposal /vivo_i18n ,** copy **solr** and **vivo_home** in aprropriate directory
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-config-proposal/vivo_i18n`
		`mkdir -p $VIVO_HOME_I18N/home/config`
		`cp -r vivo_home/* $VIVO_HOME_I18N/home/config`
		`sudo -u solr cp -r solr/server $SOLR`

2	From de vivo installer for i18n: cd **$REG_TEST_GIT/vivo-regression-tests/vivo-installer-i18n**
	
	install the vivo-i18n (solr and vivo_home are pre-configured with the UQAM-DEV installation)
		For UQAM-DEV installation
		
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-installer-i18n`
		`mvn install -s UQAM_DEV_RgTest_i18n_settings.xml -Dmaven.test.skip=true`
		
		For OTHER installation
		
		`cd $REG_TEST_GIT/vivo-regression-tests/vivo-installer-i18n`
		`# Edit and change attributes values to appropriate settings of`
		`# NON_UQAM_DEV_i18n_RgTest_settings.xml file`
		`vi NON_UQAM_DEV_i18n_RgTest_settings.xml`
		`mvn install -s NON_UQAM_DEV_i18n_RgTest_settings.xml -Dmaven.test.skip=true`

Step 3 c: Checking the installation

This section is used to verify the co-execution of VIVO-ORIG and VIVO-i18n.

	Description	Bash commands

1	Starting solr
	
	Make sure your JAVA_HOME is set up correctly e.g.
	`export JAVA_HOME=C:/UQAM-DEV/jdk`
		`cd $SOLR/bin/`
		`./solr.cmd start`
		
		`# Start http://localhost:8983/solr/#/ in your WebBrowser`
		`# and look for vivocore_i18n and vivocore_orig in the`
		`# core selector dropdown menue`

2	Start TOMCAT	`cd $REG_TEST_HOME/apache-tomcat-8.5.11/bin/` `./catalina.bat jpda start`
3	Verify VIVO i18n at http://localhost:8080/vivo/	blocked URL
4	Verify VIVO otiginal at http://localhost:8080/vivo_orig/	blocked URL

Step 4: Configure property files in preparation for running a test case

	Description	Commands
1	**a) Setup VIVO_ORIG** Assign the password for for the user admin	blocked URL — On the page of http://localhost:8080/vivo_orig/ click continue blocked URL — In the login passord enter: email: vivo@uqam.ca passwd: rootPassword blocked URL — Enter the new passord e.g.: Vivo2435....

2	Edit the file C:\UQAM-DEV\vivo_orig\home\config\runtime.properties	Make sure the variable rootUser.emailAddress matches the login you entered in the last step.
		
		Don't worry about the variables below, they are not enabled for version 1.11.1 of VIVO.
		
		`rootUser.passwordChangeRequired`
		`rootUser.password`

3	**b) Setup VIVO i18n**	edit C:\UQAM-DEV\vivo\home\config\runtime.properties
		
		Validate that the variables are correctly configured
		
		`rootUser.emailAddress = vivo@uqam.ca`
		`rootUser.passwordChangeRequired = false`
		`rootUser.password = Vivo2435....`
		`RDFService.languageFilter = true`
		`languages.selectableLocales = fr_CA, en_CA, en_US, de_DE`

4	Restarted Tomcat if changes have been made to any of the runtime. properties files.	

Step 5: Setting up the test bench

Description	Commands

a) testbench runtime.properties
 	
Edit the file C:\UQAM-DEV\GIT\vivo-regression-tests\src\main\resources\runtime.properties
	Make sure that the content (especially the credential section) corresponds to the VIVO login values.
	
	runtime.properties
	
	`#`
	`# Credential for connecting to Vivo`
	`#`
	`vivo.i18n.rootlogin=vivo@uqam.ca`
	`vivo.i18n.password=Vivo2435....`
	`vivo.orig.rootlogin=vivo@uqam.ca`
	`vivo.orig.password=Vivo2435....`
	`# ...`

b) TestRunner properties
 	
Edit the file C:\UQAM-DEV\GIT\vivo-regression-tests\src\main\resources\TestRunner.properties
	Select the test case you want to run
	
	e.g.:
	
	TestRunner.properties
	
	`ResearchOverviewToPerson=true`
	`EmailAddress=false`
	`HeadOfFaculty=true`

c) Maven profile
 	
Edit the file C:\UQAM-DEV\GIT\vivo-regression-tests\settings.xml
	Select the test case you want in the activeProfile section. Seul une profile peut être sélectionné
	
	e.g.:
	
	settings.xml
	
	`<activeProfiles>`
	`<activeProfile>ResearchOverviewToPersonTest</activeProfile>`
	`<!-- <activeProfile>EmailAddressTest</activeProfile> -->`
	`<!-- <activeProfile>HeadOfFaculty</activeProfile> -->`
	`</activeProfiles>`

d) For Linux, Installing geckodriver	`#Edit the texbenvh runtime.properties`
	`vi $REG_TEST_GIT/vivo-regression-tests/src/main/resources/runtime.properties`
	
	`# Comment the line selenium.driver.location=./lib/geckodriver-v0.26.0-win64/geckodriver.exe`
	`# and`
	`# uncomment selenium.driver.location=./lib/geckodriver-v0.26.0-linux/geckodriver`

Step 6: Running a test case

Description	Command

a) TestRunner Java runtime mode	`export DISPLAY=:0 (For Linux user)`
	`cd $REG_TEST_GIT/vivo-regression-tests`
	`mvn install`
	`mvn exec:java`

b) Test case running by profiles setting in settings.xml	`export DISPLAY=:0 (For Linux user)`
	`cd $REG_TEST_GIT/vivo-regression-tests`
	`mvn install`
	`mvn test -s settings.xml`

c) Test case running by profiles in options	`export DISPLAY=:0 (For Linux user)`
	`cd $REG_TEST_GIT/vivo-regression-tests`
	`mvn install`
	`mvn test -P EmailAddressTest,HeadOfFacultyTest,ResearchOverviewToPersonTest`

Done