Introduction

This short manual is intended to provide the basic mechanics of the automation framework developed by Randy Hildebrand and the VETSNET Automation Team for the GUI-based applications for VETSNET, namely FAS, SHARE, and AWARDS. Looking forward to the future, some consideration is also given to how this framework could be used for web-based implementations of these programs.

The primary programs to be used include Rational Functional Tester, various adapters for Rational Functional Tester (RFT), and Rational Quality Manager (RQM). RFT is the VA-approved program for automation. The Java version of RFT is the one that will be used throughout. Depending on needs, some Windows batch programming and UNIX shell programming have also been used in conjunction with RFT and/or RQM.

Since Java is primarily an object-oriented (OO) programing language, an OO approach has been utilized in the development of the framework, although more functional aspects of the language have been introduced with Java 8.

The framework has primarily used IBM’s implementation of the Java language. Using a Java-only approach frequently is not able to correctly identify objects and/or is significantly slower than the framework approach used herein.

RFT approaches the applications much as an end user. The framework includes all the individual clicks, shifting of cursor focus, and filling in of fields that a typical end user would do.

We will primarily be using the Functional Test (default) perspective, except for the Interface module.

Objects

# RFT Objects vs. VB6 Objects

The VETSNET applications have been developed using VB6. RFT interprets these objects in its own terms. The framework explicitly converts these VB6 objects to RFT objects so that it can deal with them. An individual object library is developed for each RTC/RM story to reduce confusion between similarly named objects. A conversion library appears below.

|  |  |  |
| --- | --- | --- |
| VB6 Object | RFT Object | Appearance |
| ThunderRT6TextBox | TextGuiTestObject |  |
| ThunderRT6CommandButton | GuiTestObject |  |
| TreeView20WndClass | GuiSubitemTestObject |  |
| ThunderRT6ComboBox | TextSelectGuiSubitemTestObject |  |
| ThunderRT6OptionButton | ToggleGUITestObject |  |
| ThunderRT6CheckBox | GuiTestObject |  |
| ThunderRT6ListBox | SelectScrollGuiSubitemtestObject |  |

Example: if the user story stated to click the clear button, the way to describe that in an RFT library would be:

public GuiTestObject clear(){

TestObject to=clearwindow().find();

return new GuiTestObject(to);

}

Explanation:

public indicates that other java classes can reference this object

GuiTestObject is the object return type

clear() is the name of the object as used in the interface

TestObject declares a generic test object and assigns it to the variable to

clearwindow() is a reference to the object as contained object map. This name can be changed but has be different from the name of the object as used in the interface

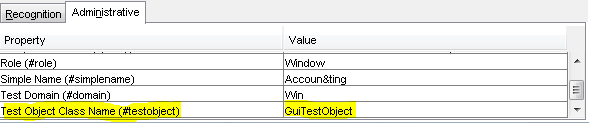
find() finds the object described in the object map

return returns the object

new sets aside memory for the object

GuiTestObject(to) converts the generic test object to (in this case, clearwindow()) to a GuiTestObject

NOTE: on the administrative tab of the TestObject Map the correct association can be seen under the Test Object Class Name:



The value cannot be successfully changed.

Each object dealt with in the user story needs its own object definition as described herein.

# RFT Objects vs Web-Based Objects

Web-based objects do not explicitly require an object map and a suggested approach is contained herein.

RootTestObject root=getRootTestObject();

public GuiTestObject select(){

Property[] p=new Property[2];

p[0]=new Property(“.class”,”Html.A”);

p[1]=new Property(“.name”,”Html link”);

TestObject found=root.find(atChild(p));

return new GuiTestObject(p);

}

# The Object Map

# The Object Library

# Ensuring Objects Continue to Function Across Builds

# Non-Readable Objects

## MSFlexGrid

## Reports

Interfaces

# Java 7-Type Interfaces

# Java 8-Type Interfaces

# Interfaces and Adapters

## RQM and RFT

## RQM and UNIX

Executing Code

# Order

# Implements

# Imports

# Verifications

# Greyed-out Objects

Add-Ons

# Apache POI

# TestNG

# Others