How to make
Web applications
fit for
automated GUI tests

Guideline

AUTHOR	Torsten Zelger
VERSION	V 1.2
VERSION DATE	May 23, 2004

Contents

1 Introduction

Rational Robot uses recognition methods to identify components in the application-under-test. These recognition methods are saved as arguments in scripts so Robot can correctly identify these component during playback.

For example, Robot can identify a Java Button by the visible text displayed on the button.

If the text changes after the script has been recorded, the script may fail when it is played back.

The best way to make sure that Robot recognizes a component is to assign a unique name to the GUI element.

Once the name is provided do not change it

This however is done in different ways depending on the technology of the AUT (application under test).

For **Win32** applications built with Visual C++ usually the ID is provided automatically through the use of the resource.h file. Unique here is fine if the scope is within a dialog. Visual Studio however guarantees uniqueness all over the project.

For **.NET** applications the property name is also provided automatically when using Microsoft Visual Studio .NET. Unique for .NET means "within a dialog". Redundant names can be used for different dialogs.

For **Java** applications and applets the way of providing unique recognition is to add a name to the component/control. Unfortunately the IDEs do not generate the name automatically. A JAVA-application can run without a providing custom name. Chapter 4 explains what a developer must do in order to make Java-Applications testable with Rational Robot. Unique for Java means "unique within a applet or a simple dialog".

HTML provides "id" and "name" attributes to identify a HTML element. Unique for HTML document means "unique within a webpage". On other webpages the same name can be used again.

If you are a developer think about what happens if your neighbour changes the interface of a DLL that is called within your code. If your neighbour does not inform you about the changes, you will probably not know during compile time because the module is loaded dynamically into memory during execution. If the error occcurs at one of your clients it is probably too late to fix the problem within an acceptable timeslice. This is no different from the view of test automation.

2 It is all about uniqueness

This chapter demonstrates the need of unique object recognition mechanisms in order to keep maintenance of automated testscripts manageable. If you read this chapter you are aware of almost all concerns of a software tester given the job to do GUI Test Automation.

2.1 Automated verification

As an example, an automated testcase might contain the verification whether the white Chevy Impala 1959 is loaded and ready.


Chevy 1950


Chevy 1957


Chevy 1958


Chevy 1959

If we browse the list of pictures using Robot to record the actions, the following script is beeing generated:

```
'// HTMLImage without ID-Tags
Window SetContext, "Caption=HTML - Microsoft Internet Explorer", ""
HTMLImage Click, "Index=1", "Coords=117,81"
HTMLImage Click, "Index=2", "Coords=96,62"
HTMLImage Click, "Index=3", "Coords=84,43"
HTMLImage Click, "Index=4", "Coords=88,40"
```

In order to verify whether the white Chevy Impala is loaded a test automator writes the following code:

```
Dim vValue As Variant
Dim strObjRecognition As String
strObjRecognition = ".\;Type=Window;Caption={*icrosoft Internet Explorer*};\;Type=Image;Index=4"

if (SQAGetProperty(strObjRecognition, "readystate", vValue) = sqaSuccess) then
    SQALogMessage sqaPass, "Yes, White Chevy Impala is ready", ""
else
    SQALogMessage sqaFail, "No, White chevy Impala is not loaded", ""
end if
```

The above script has a major drawback. Consider the situation where in a later version of the AUT an additional column will be introduced (Chevy 51). In addition consider the situation where for some unknown reasons the white Chevy is no longer displayed as expected (see below), instead a red cross is displayed, indicating the image could not be located and loaded.


Chevy 50


Chevy 51


Chevy 57


Chevy 58


Chevy 59

Although we expect the test script to report a failure it actually passes successfully because the script only verifies whether in cell 4 the picture within is ready. But there is a different picture in cell 4 and the white Chevy is lost. This failure goes undetected during regression test.

In order to avoid scripts to report so-called "false-positives" and/or "false-negatives" it is highly recommended to introduce a unique identifier for all objects that are subject to be verified.

2.2 What developers can do

For this specific example provide each <img>-element with a unique "id"-attribute. Typically these are not inserted automatically in your IDE and must therefore setup manually by the developer (see example below)

```
<td width="25%"><img id="id-chevy57" border="0" src="chevy57.jpg" width="177" height="119"
<td width="25%"><img id="id-chevy58" border="0" src="chevy58.jpg" width="177" height="118"
<td width="25%"><img id="id-chevy59" border="0" src="chevy59.jpg" width="177" height="119"
<td width="25%"><img id="id-chevy50-painted" border="0" src="chevy50-painted.jpg" width="1
tr>
```

Having done this the automated test script can be adapted as follows:

```
Dim vValue As Variant
Dim strObjRecognition As String
strObjRecognition = ".\;Type=Window;Caption={*icrosoft Internet Explorer*};\;Type=Image;HTMLId=id-chevy59"

if (SQAGetProperty(strObjRecognition, "readystate", vValue) = sqaSuccess) then
    SQALogMessage sqaPass, "Yes, White Chevy Impala is ready", ""
else
    SQALogMessage sqaFail, "No, White chevy Impala is not loaded", ""
end if
```

Because we use a unique identifier for the Chevy Impala the error is now detected during regression test.

What counts for pictures is no different for other GUI components such as PushButtons, Links, etc.., therefore developers must provide unique names for those as well.

How to proceed for each different control is subject for short explanation within this document.

> If no mechanism is provided in order to uniquely identify objects within a web-page, a lot of scripts need to be adapted just because a component was moved from one location to another. Scripts must be changed even though there was no change in the business logic.

2.3 Scope of unique ID/Names

It is not meant to burden developers with another hard to achieve guideline. Therefore be aware that the scope of "uniqueness" is only meant within a simple webpage or dialog or applet which generally is easy to implement.

3 HTML

Robot makes use the of the standard HTML attributes "id" and/or "name". These attributes provide an element with an individual name.

If HTML elements are not provided with either "name" and/or "id" attributes they must be introduced with either "id" or "name" or both in order to provide a unique mechanism for Robot to touch the elements ie for verification or performing actions.

The "id" attribute must always start with a letter "A-Z" or "a-z". Subsequent letters must always consist of either:

- A-Z
- a-z
- 0-9
- - (dash)
- _ (underscore)
- : (colon)
- . (dot)

Special chars, blanks and other punctuation than noted above are not allowed

The id and the name attribute can be provided for ALL HTML tags except for

- <base>
- <head>
- <html>
- <meta>
- <script>
- <stype>
- <title>

The lenght of "id" attribute value must not exceed 78 characters

3.1 Editfield

It is recommended to use the "name" attribute to uniquely identify an EditField although it is possible to use the "id" attribute instead/in addition.

```
<input type="text" id="id-lastname" size="20" value="vw" name="lastname">
<input type="text" id="id-firstname" size="20" value="vw" name="firstname">
```

3.2 CheckBox

It is recommended to use the "name" attribute to uniquely identify a CheckBox although it is possible to use the "id" attribute instead/in addition.

```
<input type="checkbox" id="id-visa" name="VISA" value="ON">MyCheck>VISA
<input type="checkbox" id="id-master" name="MASTER" value="ON">MyCheck>MASTER
```

3.3 RadioButton

The name of a bunch of radio-buttons typically is the same in order to bring them together (one click moves the dot from one radio-button to another). In order to make Robot recognize which radio-button was selected it is **necessary** to add an ID-Tag.

The "name" attribute is used to group several radio-buttons. It is recommended to use the "id" attribute to uniquely identify a particular radio-button item.

```
<input type="radio" id="id-female" value="female" name="sex" >Female
<input type="radio" id="id-male" value="male" name="sex" >Male
```

If no "id" attribute is provided for RadioButtons the only useful way to make Robot identify a particular RadioButton is through using its label. Unfortunately the label may be subject of change ie when testing the same webpage in a different language.

3.4 HTMLLink

It is recommended to use the "name" or the "id" attribute to uniquely identify a HTMLLink.

```
<a id="id-amazon" href="http://www.amazon.com">[amazon]</a>"
<a id="id-ebay" href="http://www.ebay.com">[ebay]</a>"
<a name="myGoogle" href="http://www.google.com">[google]</a>"
```

If no "id" or "name" attributes are provided Robot is still able to recognize the target link element through using its label. However the label can change according to new requirements and if the webpage is shown in a different language.

3.5 HTMLImage –(Link)

If a link is placed upon an image the HTML-Code consists of an <img>-tag embedded in an <a> tag.

To make the link/image unique for Robot it is NOT necessary to provide an "id" or "name" attribute for both. Robot only recognizes the inner most element if all elements have unique tags. In the following example the red part is not necessary.

```
<a id="id-lnk-cars" href="/cars.html"><img src="cars.gif" id="id-pic-cars"</a>
<a id="id-lnk-boats" href="/boats.html"><img src="boats.gif" id="id-pic-boats" </a>
```

If however the embedded <img> element has no associated "id/name" attribute you must introduce them for the <a> element.

3.6 PushButton

It is recommended to use the "name" attribute to uniquely identify a PushButton although it is possible to use the "id" attribute instead/in addition.

```
<input type="button" id="id-btn-swiss" value="Swiss" name="btn-swiss">
<input type="button" id="id-btn-german" value="German" name="btn-germ">
<input type="button" id="id-btn-brit" value="British" name="btn-brit">
```

If no "id" or "name" attributes are provided for PushButtons the only useful way to make Robot identify a particular PushButton is through using its visible text. Unfortunately the text may be subject of change ie when testing the same webpage in a different language or when the text is changed in a later release.

3.7 ListBox

It is recommended to use a unique "name" and/or "id" attribute to identify a listbox. It is not necessary to provide both.

```
<select id="id-type" size="6" name="type">
    <option value="001">Sedan</option>
    <option value="002">Convertible</option>
    ....
```

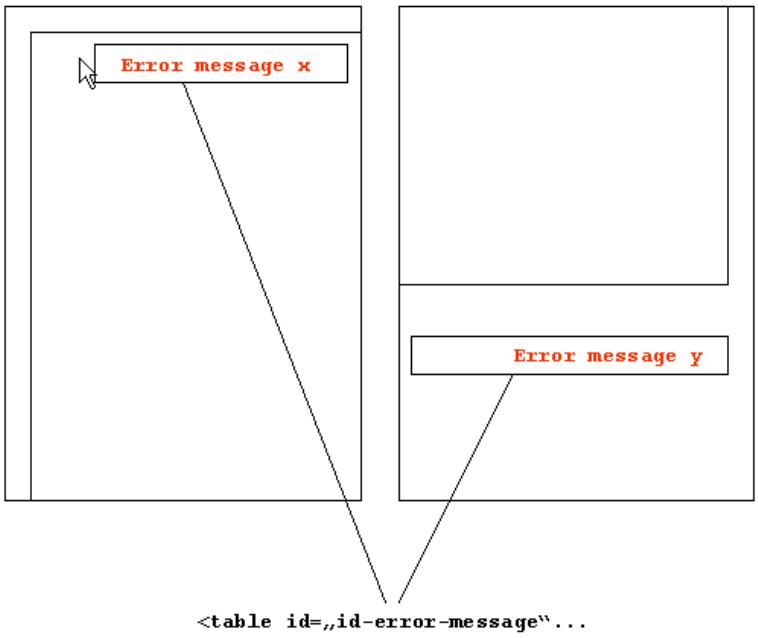
For the items wihin this the list (option) it is recommended but not mandatory to provide attribute "value" with a unique identifier that is consistent in different languages because Robot cannot recognize a unique "id" for each option (see RadioButton, there you can).

Robot can access an item through the index (relative position) or visible text. If the value is unique a loop can be scripted to parse through the list of items until the correct "value" is found and then select the list item based on the text returned for this value.

3.8 Table

In theory "id" and "name" attributes can be introduced for all HTML elements except <base>, <head>, <html>, <meta>, <script>, <style>, <title>

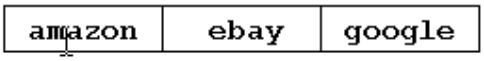
However when it comes to testing a displayed error-message/text provide a unique identifier for the table that contains the error. The same ID should be used in all webpages that produce error-message of the same type.



In webpages typically a lot of tables are cascaded and re-arranged what makes recognition impossible, therefore it is very important – for some tables - to provide at least one id in order to enable Robot to easily verify a text that resides within that table or cell

3.9 Map/AreaTags

In the example below you may expect the tabs as typical web-links where each of the links is a separate picture. In fact the above object can be implemented in different ways. One is to separate each link in a different picture (with unique ID-Tags), another is to summarize all those links in one single picture object only.



The recommended way of HTML-Implementation looks as follows:

```
<html>
<body bgcolor="#FFFFFF" text="#000000">
<img id="id-main-menu" src="aut-gd-area-sample.gif" width="344" height="40"
usemap="#Map1" border="0">
<map id="id-map1" name="Map1">
  <area id="id-area-amazon" shape="rect" coords="4,6,116,31"
href="http://www.amazon.com">
  <area id="id-area-ebay" shape="rect" coords="116,6,227,31"
href="http://www.ebay.com">
  <area id="id-area-google"   shape="rect" coords="227,6,338,31"
href="http://www.google.com">
</map>
</body>
</html>
```

If an Area-Tag has an id-attribute, Robot recognizes "clicks" as follows:

```
HTML Click, "Type=HTMLMapArea;HTMLId=id-area-amazon", "Coords=45,35"
HTML Click, "Type=HTMLMapArea;HTMLId=id-area-ebay", "Coords=45,35"
HTML Click, "Type=HTMLMapArea;HTMLId=id-area-google", "Coords=45,35"
```

If the HTML code for maps does not provide a unique id for each area-tag Robot recognizion-mechanism is different:

```
HTMLImage Click, "HTMLId=id-main-menu", "AreaIndex=1"
HTMLImage Click, "HTMLId=id-main-menu", "AreaIndex=2"
HTMLImage Click, "HTMLId=id-main-menu", "AreaIndex=3"
```

The good news is that now it recognizes the the map-id, which was not the case in our first example. However the bad thing is that the recorded script heavily depends on a stable arrangement of the buttons.

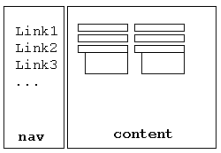
If one day another "button" is inserted in between "Amazon" and "Ebay", statement 2 will no longer link you to "Ebay" instead statement 3 will. Statement 3 was originally recorded to link you to Google. In the regression test it will not.

As it is labor-intensive for a developer to re-arrange buttons in an area (need to change the image and the HTMLMapArea) this could be a indication that the likelihood of changes is low and you are safe without any "id" attributes within the map.

However if the image is re-arranged automatically through the use of a modern content management-system, the likelihood of change is much higher. Therefore talk to the developer to find out what might be the best approach for your particular AUT.

3.10 Frames

Remember the name/id attribute must be unique within a single webpage only. When frames are used to make several webpages visible all at once (see picture) it is mandatory to provide a "name" and/or "id" attribute for each frame. Robot uses that "name" or "id" to focus the target webpage prior to perform actions on a GUI component that resides within a frame.




```
...
<frameset cols="250,*">
   <frame id="id-nav" src="nav.htm" name="Nav">
   <frame id="id-content" src="content.htm" name="content">
</frameset>
...
```

If a unique identifier is set for a frame, Robot is able to identify the frame as follows:

```
Window SetContext, "Caption={*icrosoft Internet Explore*}", ""
Browser SetFrame,"Type=HTMLFrame;HTMLId=id-content",""
ListBox Click, "Type=ListBox;Name=name-my-cars", "Text=BMW"
```

If there is no unique identifier for a frame, the Robot script is not very resilient if changes are done related to the composition:

```
Window SetContext, "Caption={*icrosoft Internet Explore*}", ""
Browser SetFrame,"Type=HTMLFrame;Index=1",""
ListBox Click, "Type=ListBox;Name=name-my-cars", "Text=BMW"
```

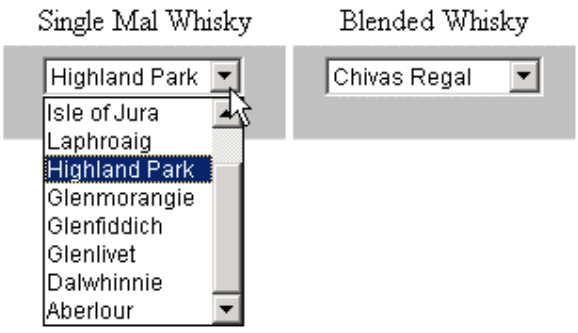
If you move the navigation page to the right, the script will fail!

3.11 Embedded Frames

Embedded frames (iFrame) are no different from the view of Robot. Even though the source of the embedded frame may not be visible for a typical user all the GUI elements can be read by Robot without restriction.

4 JAVA

Besides Win32 and HTML Robot also supports JAVA for GUI Test Automation. However to enable object recognition of Java components you have to follow simliar guidelines in order to make components testable automatically.



If the JAVA GUI component exports a public String getName() method and this method returns a name that starts with a dot (.) character, Robot uses this name to uniquely identify the component. The **DOT** prefix is necessary to make sure that the name has been explicitly set by the user and not set to a default value by the browser.

This recognition method is available with all java.awt. Component derived classes. If the component contains an accessibleContext.accessibleName property, Robot uses it to recognize the component. This recognition method is available only with JFC-derived classes.

By assigning unique names to your Java components, you can make your scripts more resilient.

The example below demonstrates how Robot recognizes named GUI components:

```
Window SetContext, "Caption=Microsoft Internet Explorer", ""
Browser SetApplet,"Type=JavaWindow;Name=.id-applet-singlemalt",""
ComboBox Click, "Type=ComboBox;Name=.id-list", ""
Browser SetApplet,"",""
ComboListBox MakeSelection, "Type=JavaWindow;Name=.id-applet-singlemalt;;\;
   Type=ComboListBox;Name=.id-list", "Text=Highland Park"
```

If a unique name is not provided or does not start with the "." as a prefix Robot produces the followig output:

```
Window SetContext, "Caption= Microsoft Internet Explorer", ""
Browser SetApplet,"JavaClass=MyApplet",""
ComboBox Click, "Type=ComboBox;Index=1", ""
Browser SetApplet,"",""
ComboListBox Click, "JavaClass=MyApplet;\;Type=ComboListBox;Index=1", ...
```

The following piece of code demonstrates how to name the Java GUI components:

```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;

public class MyApplet extends Applet implements ActionListener {

        private Choice          m_pChoice;

        public void init()
        {
                this.setName(".id-applet-singlemalt");

                m_pChoice  = new Choice();
                m_pChoice.setName(".id-list");

                m_pChoice.addItem("Talisker");
                m_pChoice.addItem("Tobermory");
                m_pChoice.addItem("Isle of Jura");
                m_pChoice.addItem("Laphroaig");
                m_pChoice.addItem("Highland Park");
                m_pChoice.addItem("Glenmorangie");
                m_pChoice.addItem("Glenfiddich");
                m_pChoice.addItem("Glenlivet");
                m_pChoice.addItem("Dalwhinnie");
                m_pChoice.addItem("Aberlour");

                add(m_pChoice);

        }

        public void actionPerformed(ActionEvent event) {
                repaint();
        }
}
```

As the above example shows we need to also name the applet itself, because there is no option to provide the HTML tag "applet" with an "id/name" attribute:

```
<applet name="my-singlemalt" id="id-singlemalt" code="MyApplet.class" ...
<applet name="my-blended" id="id-blended" code="MyApplet2.class" ...
```

Therefore you identify the applet by assigning a name to the applet (this.setName...)

Scripts based on the fact of missing unique names are very vulnerable when it comes to later changes. If the developer introduces an additional combolistbox item before the existing item, this new item will become recognized as "index=1".

In this case the script unwantedly touches the wrong listbox-selection which will result to a failure or it produces so-called "false negatves" where in fact the application works as designed.

5 Third party ActiveX controls

5.1 Background

There are a large number of third-party control vendors creating and revising ActiveX controls – toolbars, grids, and widgets of all kinds – every day.

Although Robot's built-in Visual Basic support allows it to collect some information about these ActiveX controls, truly robust and resilient scripts often require a more extensive interaction than can be supplied generically.

To better serve the needs of customers using Rational Robot to test applications that incorporate new and recently updated controls, Rational has developed a mechanism to extend the capabilities of Rational Robot. It includes a Java proxy interface that allows users to construct a proxy for certain kinds of Java controls. A similar mechanism was added for ActiveX controls.

These proxies implement a specific set of methods that are called by Rational Robot to collect information about the control when scripts are recorded and to exercise the control during playback.

During script recording, Robot intercepts actions being performed against all controls. When an unrecognized ActiveX control is acted upon, Robot checks the Windows Registry for a proxy that is able to answer for the control.

If such a proxy is found, Robot provides the proxy with a pointer to the instance of the ActiveX control that it is handling. Robot then queries the proxy about the control through a standard interface that must be exposed by the proxy. The proxy writer, based on his knowledge of the control, implements the methods of this interface to return the requested information to Robot.

5.2 If the source code is not present

Rational Robot's proxy support for ActiveX controls offers to organizations to make custom controls known and testable with Robot

The paper "Extending Rational Robot Support for New ActiveX Controls" (published at RDN) focuses on the case in which the individual creating the proxy **does not have direct access to the control's source code.**

Control vendors can use the proxy mechanism to add value to their products by quickly and easily developing a proxy and making it available to their customers. Control consumers can ensure effective and efficient functional testing by developing their own proxies – on their own schedule – for any and all ActiveX controls used in their software.

5.3 If the source-code is present

Individuals and organizations that have access to a particular control's source may not need to write a proxy, and may choose instead to enhance the control itself so it can be accessed directly by Rational Robot. If, for example, a control already implements the **IAccessible interface for Active Accessibility,** then Rational Robot only needs the control to implement a relatively simple new interface, ITestProxy, to access it directly.

5.4 Conclusion

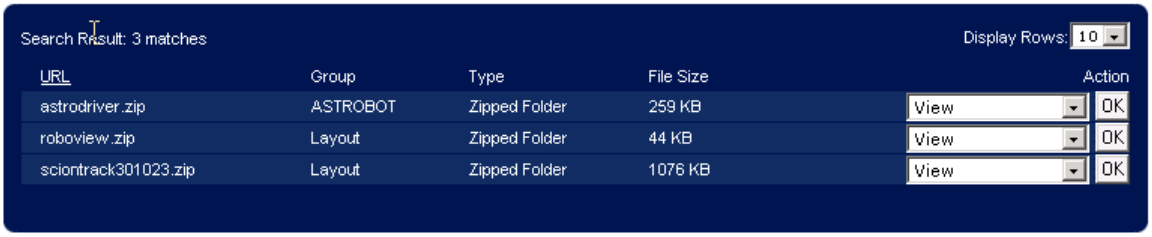
Before considering the implemention of any third party controls into your source-code be aware that the risk is high of not being able to perform automated tests against the control Only a small set of proxies are available for free (ie. Infragistics) and it may cost additional money to develop your own proxies. This counts for all kinds of customized controls (to Java as well)

6 Other issues

6.1 Where output is dynamic (unpredictable)

Even though all requirements have been met in order to make webbased applications ready for test automation there is still a challenge to verify dynamic output.

The goal of the following sample test is to verify whether "roboview.zip" is listed as a result and after that an operation is performed on the appropriate listbox to the right.



Let's assume the first test was successful and now it comes to perform a listbox selection with Robot:

```
ComboBox Click, "Name=my-action-cmb;Index=2", ""
ComboListBox Click, "Name=my-action-cmb;Index=2", "Text=CheckOut"
```

Now although the ComboBoxes have names, the names are not unique within the web-page because the same control is shown three times, each of them stands for a different record.

This makes our script vulnerable to changes. The sort order can be different under various conditions. E.g. if the underlying SQL-statement in the server does not provide an "ORDER-BY" clause the produced output is unpredictable because it varies among different RDBMS vendors.

The sort order of course can also be changed by purpose. Or an additional record can be inserted in a next release.

These events make our script to fail or report failures when in fact the application works as designed.

Therefore you must provide a way to introduce a unique string. This can be the primary-key of the record or a unique name for the appropriate record.

Example 1

Each time an automated test inserts a record such as "roboview.zip", the script must remember the name. When it comes to test whether the item is shown in the list, it simply needs to use that name as part of the object recognition:

```
ComboBox Click, "Name=my-action-cmb_roboview_zip ", ""
ComboListBox Click, "Name=my-action-cmb_roboview_zip", "Text=CheckOut"
```

Example 2

Rational Robot allows for direct read/write database access using ODBC. Therefore each time a record such as "roboview.zip" is entered, the automated scripts can read the primary key and remember this string for the next test to verify whether it is shown in the list or not.

```
ComboBox Click, "Name=my-action-cmb_4495", ""
ComboListBox Click, "Name=my-action-cmb_4495", "Text=CheckOut"
```

6.2 HTML-Title

A simple thing like making sure there is a unique title for each HTML page will allow Robot to verify whether or not the right page is focused.

6.3 Object mapping

Unfortunately Robot does not provide an object mapping mechanism, therefore the object-recognition as we saw in the previous chapters is hard-coded into the script. Each time the object recognition of a particular GUI element changes scripts have to be adapted.

It is recommended to shift the recognition to external sources (e.g. an INI-File).