HTMLUnitGenerator

Enables user friendly and powerful front end testing of web applications with minimum required effort to implement.

AUTHOR: TOMAS BJERRE, TOMASBJERRE [AT] YAHOO.COM

Updated January 6, 2012

Contents

Chapter 1

Introduction

This is the official documentation of HTMLUnitGenerator. This introduction includes three questions and three answers. They are intended to quickly get you into the context of this document.

1.1 What is HTMLUnitGenerator?

It is a compiler. It translates a user friendly DSL[1] into a more advanced test case. That way you get clear test cases that are easily maintained while at the same time powerful and easily introduced in your current test suit.

1.2 Why should I use HTMLUnitGenerator?

1. **Easy to learn and fast to work with**
   The time you need to spend reading up on HTMLUnitGenerator before being able to produce qualitative test cases using it, is very short.

   If you write your test cases using, for example, raw Java and HTMLUnit you will need to come up with some hierarchy of classes to be able to re use code. XPath:s and URL:s should typically be defined once and then referenced in all your test cases. Developing such a structure takes time as well as explaining and documenting it to your colleges.

2. **No need to document test cases**
   The Flow language (see Section 2) is simple enaugh to, itself, qualify as documentation. Anyone, even people with no previous programming experience, will understand what your test cases do. The Flow language has been design with the intention to provide only an absolute minimum number of choices to the developer, in order to keep all test cases clean and neutral.

[1] Domain Specific Language

3. **Future safe**
   HTMLUnitGenerator is an open source software. You can write your own generator if you don't want to user HTMLUnit anymore, see Section **??**. Or maybe you want to test your code using several different head less browsers. The idea is to provide several different generators with HTMLUnitGenerator[2].

[2]Yes, the name will change!

Chapter 2

Flow language

This chpater will describe the Flow language. It is the language used to express test cases.

This chapter starts with a quick look at a test case written in Flow, see Section 2.1, and continued with a complete walkthrough of the language.

2.1 Introduction

A quick example of a test case written in Flow is presented in Listing 2.1. The details of this test case is explained in this chapter.

```
1 Url SomeSite is http://www.somesite.domain/
2
3 Go to SomeSite
4 Find a with attribute href set to /some/target
```

Listing 2.1. Find href of an a tag

2.2 Defining paths

Flow uses XPath[1] to define containers, like div-tags for example.

```
1 Path searchpopup is /html/body/div[7]/div/div[9]
2 Path search is //*[@id="eventIdsearch"]
3 Path website is /html/body
```

Listing 2.2. Defining XPaths

In Listing 2.2 a div is defined as *searchpopup*, an element with an id is defined as *search* and the entire website content is defined as *website*.

[1] http://www.w3schools.com/xpath/

2.3 Defining URL:s

URL:s are used in Flow to go to exact URL:s, see Section 2.4.

```
1 Url myUrl is http://www.my.domain/page.html
2 Url myOtherUrl is http://www.my.domain/page.html
```

Listing 2.3. Defining URLs

In Listing 2.3 a URL is defined as *myUrl* and another defined as *myOtherUrl*.

2.4 Go to URL:s

The *Go to* statement is used to browse directly to a URL, see Section 2.3.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
```

Listing 2.4. Go to statement

In Listing 2.4 a URL is defined and used in a *Go to* statement. This will create a test case where a browser is opened and the URL *http://www.my.domain/page.html* is visited.

So far, no assertions on the content of the visited page are made. This means the test will never fail. In order to make assertions you may want to wait for a while after the page has been loaded, in order for JavaScripts to execute. This can be done as in Listing 2.5.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl and wait 5 seconds
```

Listing 2.5. Go to statement with wait

In Listing 2.5 the test case will sleep for 5 seconds before continuing execution. This is may work in many situations but it should be avoided. You should instead wait for a condition to be true and then continue execution. This can be achieved with *wait at most* in the *Find* statement (see Section 2.5).

2.5 Find elements

The *Find* statement is used to make assertions on the content of the current page.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find containing Hello
```

Listing 2.6. Find statement

In Listing 2.6 a web page is visited and the test will fail unless it contains "*Hello*".

If you want to match a string containing whitespaces you must add quotes to the string, see Listing 2.7.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find containing "Hello World"
```

Listing 2.7. Find statement with spaces

By default *Find* will look for the given content within the entire website, */html/body*. You may change this behaviour by adding *in [XPath]*. Listing 2.8 shows how the string *Hello World* is asserted to be found in inside an element with *id* set to *mypopup*.

```
1 Path mypopup is //*[@id="mypopup"]
2 Url myUrl is http://www.my.domain/page.html
3 Go to myUrl
4 Find containing "Hello World" in mypopup
```

Listing 2.8. Find statement in given XPath

Boolean logics, *and*/*or*, can be used to accept different combinations of containing strings. Listing 2.9 shows how the string *Hello World* is asserted to be found along with *Alternative1* or *Alternative2*.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find containing "Hello World" and containing "Alternative1" or
    containing "Hello World" and containing "Alternative2"
```

Listing 2.9. Find statement with boolean logics

Find can also be used to find tags. In Listing 2.10 an assertion is made making sure an *a* tag with attribute *href* set to */link/category/blandat* is available.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find a with attribute href set to /link/category/mix
```

Listing 2.10. Find tag statement

In Listing 2.11 the a tag is asserted to be available within the XPath *mypopup*.

```
1 Path mypopup is //*[@id="mypopup"]
2 Url myUrl is http://www.my.domain/page.html
3 Go to myUrl
4 Find a with attribute href set to /link/category/mix in mypopup
```

Listing 2.11. Find tag statement within an XPath

Tags can also be combined, as in Listing 2.12.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find a with attribute href set to /link/category/mix and h2 with
    attribute class set to alternative1 or a with attribute href set to
     /link/category/mix and h2 with attribute class set to alternative2
```

Listing 2.12. Find tag statement with and/or logics

Several attributes can be asserted to be available in the same tag. Listing 2.13 shows how tag *a* is asserted to be available with either */link/category/mix* or */link/category/other* but then attribute *class* has to be defined as *otherClass*.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find a with attribute href set to /link/category/mix or attribute a set
      to "/link/category/other" and attribute class set to otherClass
```

Listing 2.13. Find tag with several attributes

Find can combine tags and strings. Listing 2.14 shows how *a* with attribute *href* set to */link/category/mix* along with string *Alternative1* or *Alternative2* is asserted to be available.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find a with attribute href set to /link/category/mix and containing "
     Alternative1" or a with attribute href set to /link/category/mix
     and containing "Alternative2"
```

Listing 2.14. Find statement with combined tag and containing with and/or logics

The content asserted by *Find* may not be available, and then the test will fail. You may accept the content to be available within a period of time, for example JavaScripts may update the DOM. Listing 2.15 shows how *wait at most* is used to wait at most 5 seconds for the assertion to become true.

```
1 Url myUrl is http://www.my.domain/page.html
2 Go to myUrl
3 Find containing "Hello World" or wait at most 5 seconds
```

Listing 2.15. Find statement with wait at most

2.6 Click on elements

A click on an element may result in the entire page being reloaded or just a JavaScript adding some content to the DOM. Listing 2.16 shows a simple example of an element being clicked.

```
1 Path choose is /html/body/div[2]/div
2 Url baspaket is http://www.bredbandsbolaget.se/tv/kanalpaket/baspaket.
     html
3 Go to baspaket
4 Click on choose
```

Listing 2.16. Click on

After clicking the element you will probably want to add a *Find* statement. Listing 2.17 show how a waiting period can be added after the click. When waiting for content to be available it is recommended to use *Find* with *wait at most* as in Section 2.5.

```
1 Path choose is /html/body/div[2]/div
2 Url baspaket is http://www.bredbandsbolaget.se/tv/kanalpaket/baspaket.
    html
3 Go to baspaket
4 Click on choose and wait 3 seconds
```

Listing 2.17. Click on and wait

2.7 Fill in forms

When testing pages that contains forms you may need to add text to them. Listing 2.18 shows how a text field can be populated with content. The URL *http://someurl.domain/* is asserted to contain a form named *userForm* which contains a text field named *name*. The text field is populated with "*Tomas Bjerre*".

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in userForm with name as "Tomas Bjerre"
```

Listing 2.18. Fill in text field in formular

Listing 2.19 shows how multiple text fields can be populated.

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in userForm with name as "Tomas Bjerre" and gender as male
```

Listing 2.19. Fill in multiple text fields in formular

Listing 2.20 shows how the text field *name* is populated with a unique (random) string.

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in userForm with name as unique string of length 6 starting with
    My
```

Listing 2.20. Fill in text field in formular with unique string

The string in Listing 2.20 can be set to a specific length as in Listing 2.21.

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in userForm with name as unique string of length 6
```

Listing 2.21. Fill in text field in formular with unique string of specific length

The string in Listing 2.21 may only be partly random as in Listing 2.22. In Listing 2.22 the string is randomly generated but it will always start with the same sequence.

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in userForm with name as unique string of length 6 starting with
    My
```

Listing 2.22. Fill in text field in formular with unique string of specific length and start

When selecting an item from a drop down, the same code as above can be used. But it may be preferred to reference the item as the index number. Listing 2.23 shows how the option number 1 is selected.

```
1 Path Website is /html/body
2 Url SomeUrl is http://someurl.domain/
3 Go to SomeUrl
4 Fill in locationForm with floor as option number 1
```

Listing 2.23. Fill in option by index number

2.8 Re-use test cases

It is ofter preferred to store definitions of XPath:s and URL:s in seperate files. Then they can be re-used in test cases. Also there may be a serious of actions needed in order to get to a specific state, such actions may also be stored in sepereate files and enabled to be re-used. Listing 2.24 shows how test code can be made available in another test.

```
1 See includes/paths.flow
2 See includes/urls.flow
3 See includes/goToOrderflow.flow
```

Listing 2.24. See statement

2.9 Use proxy

Test cases may need to be configured to use proxies when run. Listing 2.25 shows how to set the proxy server to use.

```
1 Use proxy myProxyHost with port 8080
```

Listing 2.25. Use proxy statement