Documentation of XPathBuilder

What does XPathBuilder do? It creates XPaths!

Here are examples :

```
//div[@title='Documents 1'][not(@title='Documents 2')][@wicketpath='multiFlow_panels_4_header']
//div[@title='Documents 2'][not(@title='Documents 1')][@wicketpath='multiFlow_panels_5_header']
//div[@data-path='pnlAddSource pnlAccNam txtAccountName']/input[contains(@wicketpath,'pnlAccNam_c_w_txtAccountName')][//label[contains(text(),'Account name')]]
//div[@data-path='pnl-adv-search txtSearch'][//label[text()='SEARCH TEXT']]//input[contains(@wicketpath,'txtSearch')]
//div[@data-path='lnk-adv-options']/a[contains(@wicketpath,'lnk-adv-options_script') and contains(@wicketpath,'lnk-adv-options_script')]//span[text()='Hide advanced options']
//div[@data-path='lnk-adv-options']/a[contains(@wicketpath,'lnk-adv-options_script') and contains(@wicketpath,'lnk-adv-options_script')]//span[text()='Show advanced options']
//div[@data-path='pnlStage2'][contains(@class,'widget-expanded')][not(contains(@class,'widget-collapsed'))]
//div[@data-path='pnlStage2'][contains(@class,'widget-collapsed')][not(contains(@class,'widget-expanded'))]
```

More descriptive answer would be that XPathBuilder constructs XPaths by calling method **getXPath**(*PREFIX prefix, ELEMENTS element, ACTIONS action, ATTRIBUTES attribute, String value*) and all what is needed are just few input variables...

Let's define a few XPath related properties :

PREFIX+ELEMENT[ACTION+ATTRIBUTE+VALUE]

PREFIX + ELEMENT + ACTION + ATTRIBUTE + VALUE(s) = XPath

or

ACTION + ATTRIBUTE + VALUE(s) = XPath
[ACTION+ATTRIBUTE+VALUE]

This is a really the basic concept how to create an XPath locator... Any increase of complexity is done just by repeating this simple concept over and over again, until you are able to uniquely target your needed HTML

target.

marked are XPath parts let's called it - PREFIX

//div[@data-path='pnlSetFico btnSave']//a[contains(@wicketpath,'pnlSetFico_c_w_btnSave_submit')]//span[text()='Save']]

PREFIX have two possible values to choose PREFIX.SINGLE_SLASH or PREFIX.DOUBLE_SLASH

```
public enum PREFIX {

    SINGLE_SLASH ("/"),
    DOUBLE_SLASH ("//"),
    AND ("and"),
    NOT ("not"),
    OR ("or"),
    FOLLOWING("/following-"),
    SIBLING ("sibling::"),
    DESCENDANT("descendant::"),
    CHILD("child::"),
    EMPTY(""),
    UN("un");
```

Marked are XPath locator parts let's call these ELEMENTS

//div[@data-path='pnlSetFico btnSave']//a[contains(@wicketpath,'pnlSetFico_c_w_btnSave_submit')]//span[text()='Save']]

ELEMENTS have almost unlimited variations as it depends on latest HTML coding standards.

```
public enum ELEMENTS {
```

```
    INPUT("input"),
    DIV("div"),
    A("a"),
    I("i"),
    WI("wi"),
    P("p"),
    H1("h1"),
    H2("h2"),
    H3("h3"),
    SPAN("span"),
    FONT("font"),
    OPTION("option"),
    BUTTON("button"),
    DROPDOWN("dropdown"),
    SELECT("select"),
    TEXTAREA("textarea"),
    TD("td"),
    UL("ul"),
    EM("em"),
    LI("li"),
    LABEL("label"),
    IFRAME("iframe");
```

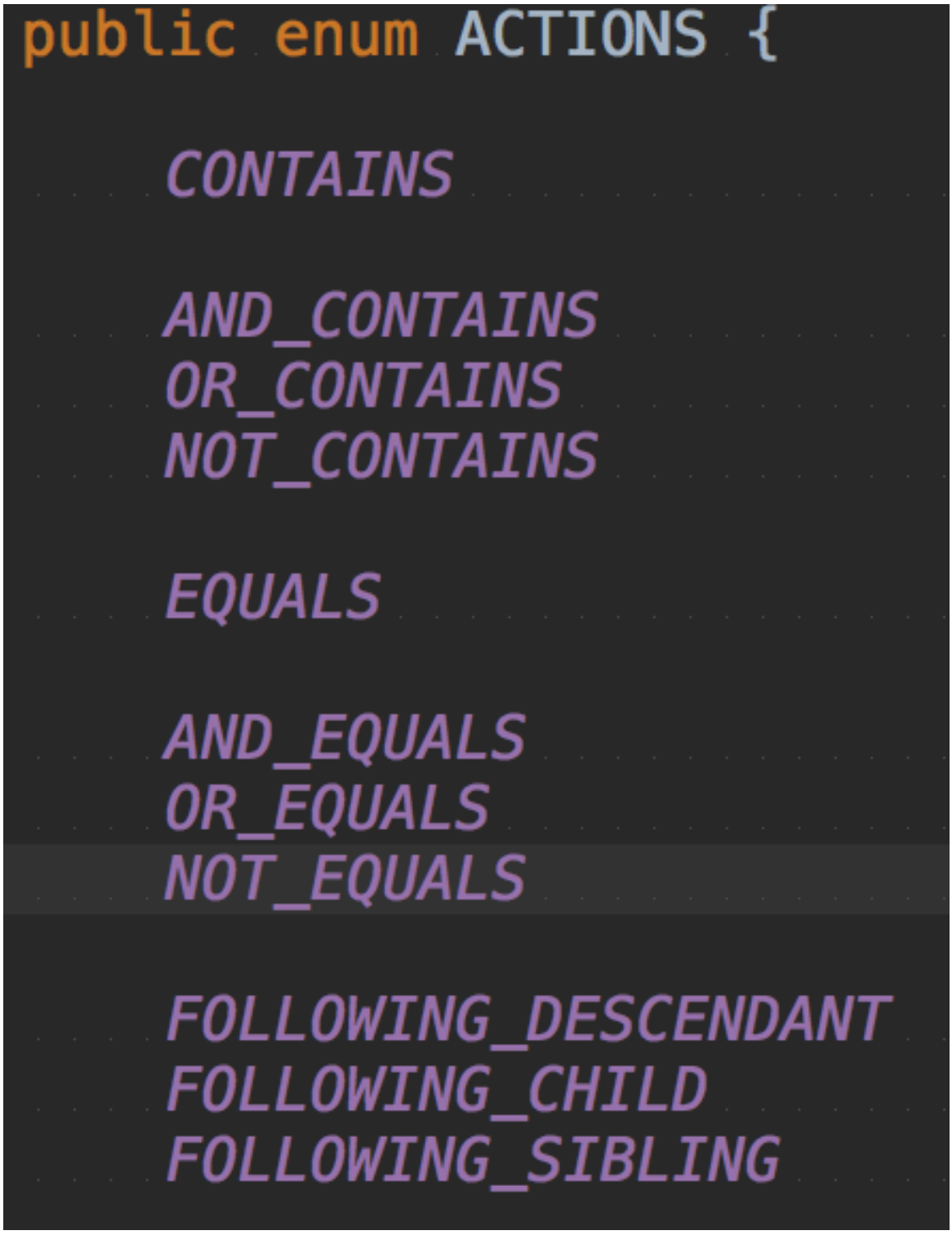
Marked are XPath locator parts let's call these ATTRIBUTES

//div[@data-path='pnlSetFico btnSave']/a[contains(@wicketpath,'pnlSetFico_c_w_btnSave_submit')][//span[text()='Save']]

ATTRIBUTES have even more possible values that are used in HTML coding.

```
public enum ATTRIBUTES {

    ID("id"),
    ARIA_DISABLED("aria-disabled"),
    HREF("href"),
    EM("em"),
    FRAG("frag"),
    CLASS("class"),
    ROLE("role"),
    DATA_REACTID("data-reactid"),
    DROPDOWN("dropdown"),
    TITLE("title"),
    TYPE("type"),
    NAME("name"),
    SELECTED("selected"),
    DISABLED("disabled"),
    STYLE("style"),
    WICKETPATH("wicketpath"),
    DATA_PATH("data-path"),
    DATA_TYPE("data-type"),
    TEXT("text()"),
    CHECKBOX("checkbox"),
    ARIA_LABEL("aria-label"),
    DATA_YTRACK("data-ytrack"),
    ANY(".");
```



And as a last are ACTIONS which actually controls how our Attributes and String values construct into valid XPath just by following several simple rules.

```
//button[contains(@id,'button_ok')]
//button[contains(@id,'button') and contains(@id,'c
//button[contains(@id,'button') or contains(@id,'ok')]
//button[not(contains(@id,'cancel_button'))]
//button[@id='button_ok']
//button[@id='button' and @id='ok']
//button[@id='button_ok' or @id='button_cancel']
//button[not(@id='button_cancel')]
```

How to use XPathBuilder in our Automation Framework ?

1. You needs to import following classes :

```
import com.r2development.leveris.utils.XpathBuilder.Enums.*;

import static
com.r2development.leveris.utils.XpathBuilder.XPath.*;
```

2. Then you are able to call getXPath methods and use Actions ,Elements & Attributes enums as bellow here :

```
String FORM_TOOLS_PANEL_HIDE =          getXPath(ELEMENTS.DIV,
ACTIONS.CONTAINS, ATTRIBUTES.WICKETPATH, "multiFlow_panels") +
getXPath_HasADescendantSpanEqualsText("Form Tools") +
getXPath(ELEMENTS.A, ACTIONS.CONTAINS, ATTRIBUTES.CLASS,
"collapse");
String FORM_TOOLS_PANEL_HIDDEN =        FORM_TOOLS_PANEL_HIDE +
getXPath(ACTIONS.CONTAINS, ATTRIBUTES.STYLE, "display: none");
String FORM_TOOLS_PANEL_NOT_HIDDEN =    FORM_TOOLS_PANEL_HIDE +
getXPath(ACTIONS.NOT_CONTAINS, ATTRIBUTES.STYLE, "display:
none");

Instead of this way which is harder to maintain even harder to
type and orientate in [I will not describe the main benefits
of usage as it is up-to you to judge] :

String a = "//div[contains(@wicketpath,'multiFlow_panels')][//
```

```
span[text()='Form Tools']]//a[contains(@class,'collapse')]";
String b = "//div[contains(@wicketpath,'multiFlow_panels')][//
span[text()='Form Tools']]//a[contains(@class,'collapse')]
[contains(@style,'display: none')]";
String c = "//div[contains(@wicketpath,'multiFlow_panels')][//
span[text()='Form Tools']]//a[contains(@class,'collapse')]
[not(contains(@style,'display: none'))]";
```

There are two basic ways how to use XPathBuilder method getXPath :

```
getXPath(); /** todo - is default xpath building method which
expects two basic ways of calling */

/** "1st way" getXPath(optional - PREFIX, mandatory -
ELEMENTS, mandatory ACTIONS, mandatory ATTRIBUTES, mandatory
String value or a XPathValues() - that will be described
later ) */
    getXPath(PREFIX.SINGLE_SLASH, ELEMENTS.DIV,
ACTIONS.CONTAINS, ATTRIBUTES.WICKETPATH, "multiFlow_panels");
    /** that will produce String ---> "//
div[contains(@wicketpath,'multiFlow_panels')]"   */

/** "2nd way" getXPath(mandatory - ACTIONS, mandatory
ATTRIBUTES, mandatory String value or a XPathValues())  */
    getXPath(ACTIONS.CONTAINS, ATTRIBUTES.TEXT, "Text to
Validate");
    /** this one will create this String --->
"[contains(text(),'Text to Validate')]"
             */
```

There are possibilities of creating an own custom methods that should reduce code redundancy. These new methods should follow naming convention as follows :

```


getXPath_DivEqualsDataPath("");
/** is equivalent to */ getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.DIV, ACTIONS.EQUALS, ATTRIBUTES.DATA_PATH, "");

getXPath_DivAndContainsDataPath("");
/** */ getXPath(PREFIX.DOUBLE_SLASH, ELEMENTS.DIV,
ACTIONS.AND_CONTAINS, ATTRIBUTES.DATA_PATH, "");
```

```
getXPath_InputEqualsName("");
/** */ getXPath(PREFIX.DOUBLE_SLASH, ELEMENTS.INPUT,
ACTIONS.EQUALS, ATTRIBUTES.NAME, "");
        getXPath_AndContainsWicketpath(PREFIX.DOUBLE_SLASH,
ELEMENTS.DIV, "", "", "");  /** */
getXPath(PREFIX.DOUBLE_SLASH, ELEMENTS.DIV,
ACTIONS.AND_CONT-9ATTRIBUTES.WICKETPATH, "");
        getXPath_DirectAButtonContainsOrContainsWicketpath("",
"");                    /** */ getXPath(PREFIX.SINGLE_SLASH,
ELEMENTS.A, ACTIONS.OR_CONTAINS, ATTRIBUTES.WICKETPATH, new
XPathValues("", ""));

getXPath_DirectAButtonAndContainsWicketpath("");
/** */ getXPath(PREFIX.SINGLE_SLASH, ELEMENTS.A,
ACTIONS.AND_CONTAINS, ATTRIBUTES.WICKETPATH, new
XPathValues("", ""));

getXPath_HasADescendant("stringText");
/** This one is only adding brackets around the variable - "["
+ "stringText" + "]" */
        getXPath_HasADescendantSpanText(ACTIONS.CONTAINS,
"");                          /** is equivalent to */
getXPath_HasADescendant(getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.SPAN, ACTIONS.CONTAINS, ATTRIBUTES.TEXT, ""));

getXPath_HasADescendantSpanEqualsText("");
/** */ getXPath_HasADescendant(getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.SPAN, ACTIONS.EQUALS, ATTRIBUTES.TEXT, ""));

getXPath_HasADescendantSpanContainsText("");
/** */ getXPath_HasADescendant(getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.SPAN, ACTIONS.CONTAINS, ATTRIBUTES.TEXT, ""));

getXPath_HasADescendantLabelEqualsText("");
/** */ getXPath_HasADescendant(getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.LABEL, ACTIONS.EQUALS, ATTRIBUTES.TEXT, ""));

getXPath_HasADescendantLabelContainsText("");
/** */ getXPath_HasADescendant(getXPath(PREFIX.DOUBLE_SLASH,
ELEMENTS.LABEL, ACTIONS.CONTAINS, ATTRIBUTES.TEXT, ""));
```

Calling of Enum values was a bit pain, but there was a huge improvement thans to this :