JAVA : (50/50) – Self learning videos

**Java** ,C#, Ruby, Python,Perl, Php,js

* Basic Syntax - class, variables, methods, return types for methods
* Looping and Conditional Statements in java
  + For
  + While
  + Do,while
  + Foreach loop
  + If
  + If else
  + switch
* Types of variables –
  + local variables
  + global variables
  + static variables
  + final variables or constants
* Difference between **classes** and **interfaces**
* **OOPS Concepts - abstraction, encapsulation, polymorphism , inheritance**
* **String –** String functions (atleast one program)
* **Exception Handling –** 
  + Try
  + Catch
  + Throw
  + Throws
  + Finally
  + User-defined Exceptions
* **Threads ( Optional )**
* **Inner classes – annonimus inner classes**
* **Collections –** List , Set, Map, Queues
  + **Similarities**
  + **Differences**
* **File Handling –** Read a file, Write a file. ( text, properties )
  + **InputStream**
  + **OutputStream**
* **Generics - Must**

Selenium – Browser Automation Tool

* Selenium IDE (record and playback option)
* Selenium RC ( Selenium 1.0 )
* Selenium WebDriver ( Selenium 2.0, 3.0 (3.141.59), 4.0 )
* Selenium Gird (Distributed Execution)

Frameworks

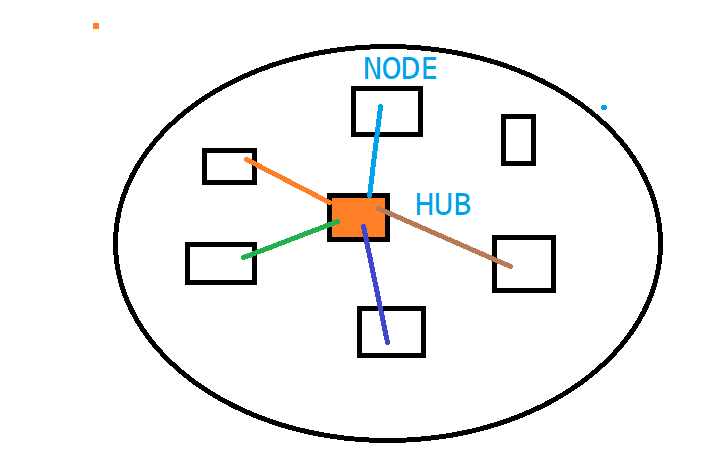
* Function Driven Automation Framework
* Data Driven Automation Framework
* Keyword Driven Automation Framework
* Hybrid Driven Automation Framework
* TestNG
  + Annotations
  + How to generate test execution reports
  + Control test annotation execution
  + Execute the tests in parallel ( 2 or n )
  + How to achieve data driven testing in testNG
  + How to use xml file for test execution
  + How to create batch file
  + How to create groups in testNG
* Page Object Model
* BDD ( if time permits )

Maven – Build Automation Tool

* What is maven
* Why it is required
* How to create a maven project
* How to add dependencies
* Life cycle of maven
* How to execute maven using command prompt
* Verify the reports

Jenkins – High level (CI \ CD) - devOps

Selenium :

* Free ,open source
* Supports multiple programming languages (**Java** , C#, Ruby, Python, Perl, php, js )
* Platform independent - (linux, Windows, MAC.. )
* Multiple Browsers ( almost all popular browsers )
* No Dedicated machine is required \*\*\*\*\*\*\* \*\*\*
* Parallel Execution ( at a time you can open more than one browser )
* Distributed Execution
* 

Testing –

* Process
  + Functional Testing
  + Integration Testing
  + System testing
  + UAT
  + Adhoc Testing
  + Usability Testing
  + Environment Testing
  + Smoke Testing
  + Sanity Testing
  + Compatibility testing
  + **Regression Testing – application altered** 
    - **new feature is added to the application,**
    - **existing feature is removed**
    - **existing feature is enhanced**
  + **Drawbacks of Manual Testing :**
    - **Time Consuming (9-6)**
    - **Person Mood**
    - **Domain Knowledge**
    - **Boring**
    - **Max one time, two, third,....... not possible**

***Why Automation*** –

We relay on tools to test the application.

* + - 24X7
    - N –number of times you can execute your script
    - Process
    - No need to have domain knowledge / **tool knowledge**
    - Save lot of time and effort

What are the Different Tools –

* + QTP \ UFT (Paid tool from HP)
  + Telerik
  + Selenium
  + Sahi
  + Robotium
  + EggPlant
  + TestComplete
  + Visual Studio
  + SoapUI
  + AutoIT
  + Sikuli

Why Selenium :

* Free ,open source
* Supports multiple programming languages (**Java** , C#, Ruby, Python, Perl, php, js )
* Platform independent - (linux, Windows, MAC.. )
* Multiple Browsers ( almost all popular browsers )
* No Dedicated machine is required \*\*\*\*\*\*\* \*\*\*
* Parallel Execution ( at a time you can open more than one browser )
* Distributed Execution (GRID)

POC – Proof of concept (using Record and playback )

* Identify 2 – 3 automation tools
* End-to-End Tests
* Pros/Cons
* Customer selection

Selenium IDE :

* To perform POC on an application
* **Chrome** and **Mozilla** as an add-on

Demo Application :

**actiTime**

Day-3

Selenium IDE :

|  |  |
| --- | --- |
| Command | Description |
| Open | Launch the application |
| Click | Performs mouse left click operation |
| Type | Performs keyboard type operation |
|  |  |

Target : means webElement

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Identifier** | **Description** | **Priority** |
| 1 | Id | Property of a html tag | 1 |
| 2 | Name | Property of a html tag | 2 |
| 3 | Tagname | Property of html tag |  |
| 4 | Classname | Property of html tag |  |
| 5 | linkText | Text on WebElement | 3 |
| 6 | partialLinkText | Partial Text of a WebElement | 3 |
| 7 | cssSelector | Expression (parent to child in HTML DOM) |  |
| 8 | xpath | Expression (either parent to child or vice versa in HTML DOM) | 3 |

Validations in Selenium IDE :

* Assert
* Verify

---------------------

Expressions:

Css :

* Htmltag[attribute = ‘value’]
  + input[id='username']
* htmltag#ValueOfIDAttribute OR # ValueOfIDAttribute
  + input#username
* htmltag. ValueOfClassAttribute
* Traversing from parent to child
  + htmlTag[attribute=’value’] > htmltagofchildElement
  + a#loginButton >div

Xpath: Relative xpath

Install xpathhelper addon on your chrome

Install xpath checker addon on your firefox

1. Basic Xpath:
   1. //htmltag
2. Apply filters
   1. //htmltag[@attribute=’value’]
      1. //a[@id='loginButton']
      2. //label[@for='keepLoggedInCheckBox']
3. AND and OR and not
   1. //htmltag[@attribute1=’value1’ and @attribute2=’value2’]
      1. //input[@class='textField' and @name='username']
      2. Search both username and password –

//input[(@class='textField' and @name='username') or @class='textField pwdfield']

iii.Using not in redbus.in

//div[@id='rb-calendar\_return\_cal']//td[text()='30' and not(contains(@class,'past day'))]

* 1. //htmltag[@attribute1=’value1’ or @attribute2=’value2’]

1. Functions
   1. text():
      1. //htmltag[text()=’text preset out side the html tag element’]
         1. //div[text()='Login ']
   2. contains(arg1, arg2)
      1. arg1 - attribute or function
      2. arg2 – partial value
      3. //htmltag[contains(@attirbure=’partialValue’]
         1. //img[contains(@src,'timer')]
      4. //htmltag[contains(text()=’partialValue’]
         1. //label[contains(text(),'Keep')]
   3. starts-with(arg1,arg2)
      1. arg1 - attribute or function
      2. arg2 – partial value
      3. //htmltag[starts-with(@attirbure=’partialValue’]
         1. //button[starts-with(@id,'ext-gen')]
      4. //htmltag[starts-with(text()=’partialValue’]
         1. //td[starts-with(text(),'Enter')]

Some more examples for above techniques :

Google search results - //h3[@class='LC20lb' or @class='sA5rQ']

//h3[(@class='LC20lb' or @class='sA5rQ') and contains(text(),'Intellipaat')]

1. Parent to child :
   1. //htmltag[@attribute=’value’]/htmlTagOfChild
      1. //a[@id='loginButton']/div
   2. //htmltag[@attribute=’value’]//htmlTagOfChild
      1. //div[@id='rb-calendar\_onward\_cal']//td[text()='21' and (@class='current day' or @class='wd day' or @class='we day')]
      2. //div[@id='rb-calendar\_return\_cal']//td[text()='30' and not(contains(@class,'past day'))]
2. Child to parent (DEPENDENT AND INDEPENDENT ) :
   1. //htmltag[@attribure=’value’]
   2. //parentHtmltag[htmltag[@attribure=’value’]]
      1. //tr[th[text()='Directed by']]//a
      2. Make my trip :

      i.      //div[div[div[h3[text()='Super Saver Bali (4N)']]]]//p[@class='font26 blackText latoBold appendBottom5']

7.        Axes functions:

a.       Traversing to Following sibling (APP – Wikipedia):

                                                               i.      //th[text()='Directed by']/following-sibling::td

b.      Traversing to preceding sibling:

                                                               i.      //span[text()='Production']/preceding-sibling::span

c.       Finding all links next to perticular element:

                                                               i.      //li[@class='toclevel-1 tocsection-4']//span[text()='Soundtrack']/following::a

d.      Finding all links before perticular element:

                                                               i.      //li[@class='toclevel-1 tocsection-4']//span[text()='Soundtrack']/preceding::a

e.       Ancestor

                                                               i.      //h3[text()='Spanish Delight 2020 (Budget Special Group Package)']/ancestor::div[@class='boxShadow bdr packageListing pointer packageDetailsBox']//p[contains(@class,'black')]

f.        Child

                                                               i.      Xpath/child::childtag

                                                             ii.      //th[text()='Directed by']/parent::tr/td/child::a

g.       Parent

                                                               i.      Xpath/parent::patenttag

                                                             ii.      //th[text()='Directed by']/parent::tr/td/child::a

--- Selenium Setup---

1. JDK 1.8

2. Eclipse - <https://www.eclipse.org/downloads/packages/>

3. Download selenium jar file ( libraries) and attach it to eclipse project

4. Download corresponding driver executable

WebDriver<I>

RemoteWebDriver

OperaDriver

FireFoxDriver

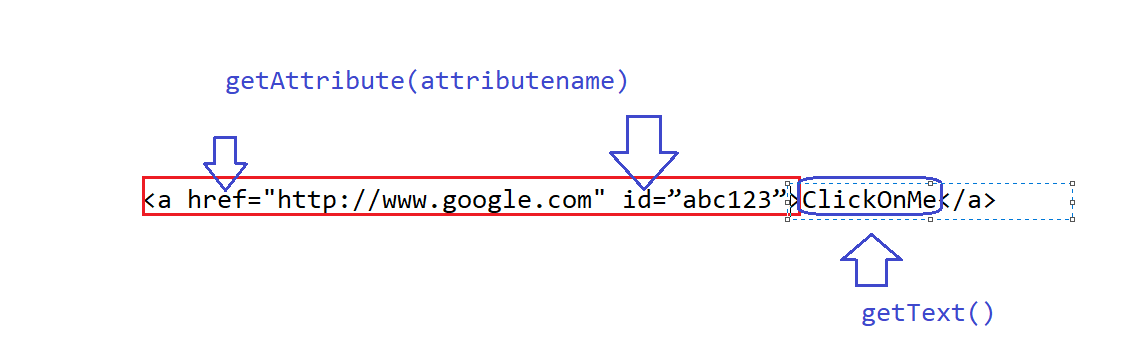
ChromeDriver

4.

java

Getting Text at Run Time:

<a href=[www.google.com](http://www.google.com) id=”abc123”>ClickOnMe</a>

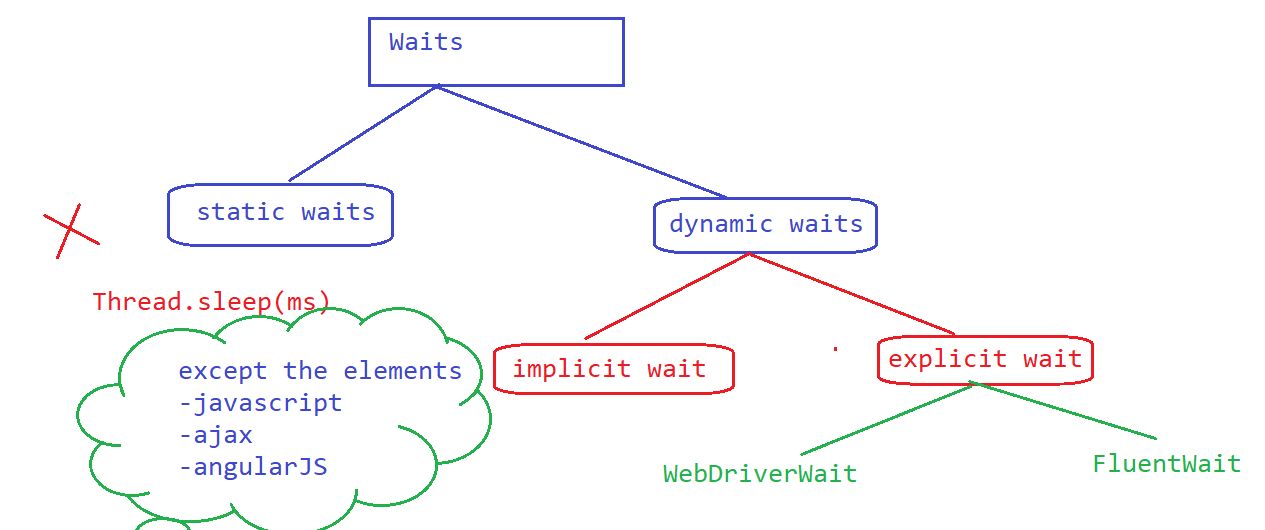


Automate Tool-Tip :

Its present inside alt or title property of a html tag

Sync issue or Synchronization issue:

Handling Sync issues :



WebDriverWait :

* Create an object of WebDriverWait and specify the max timeout
* Call until method present inside wait object
* Use methods present inside ExpectedConditions class

Drawback:

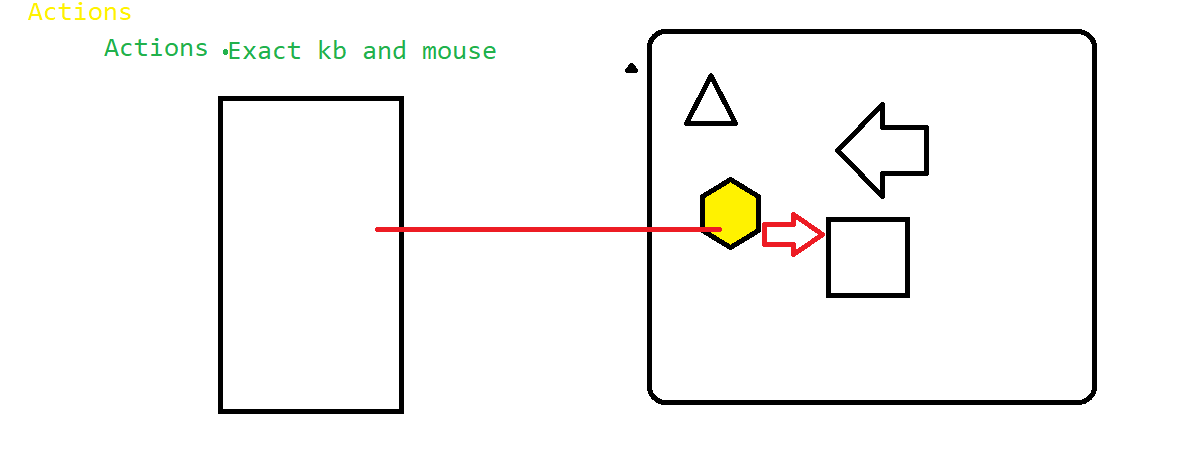
* We are forced to use the methods present in ExpectedConditions class
* Polling time is static (250 ms)
* If Any Exceptions occurred while waiting
* Keep logs while waiting...
* Customized messages while waiting

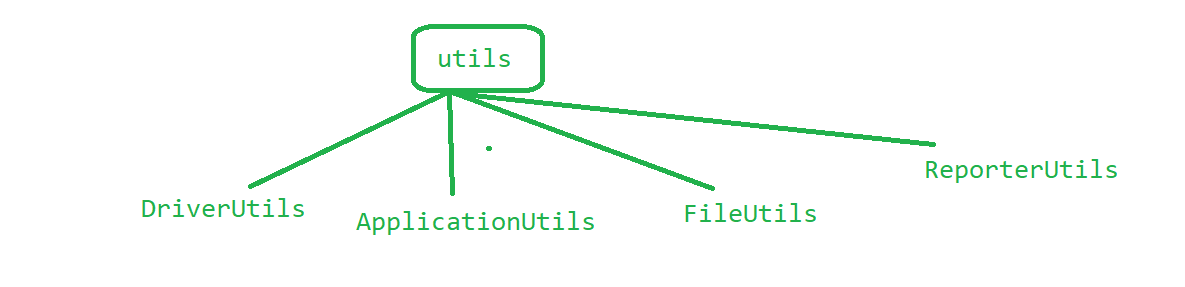
Fluent Wait :

1. Create Wait Object (Fluent Wait) –
   1. What is the webelement
   2. What is the max wait time
   3. What is the polling time
   4. What to do with the exception.
2. Create your own wait logic
   1. Function <I> 🡪 apply(arg1,arg2) -🡪 keep the logic
3. Use wait object and pass function object to until method

Actions:

To perform exact KB and mouse movement operations in selenium we can use actions.





ASSIGNMENT:

Write a Automation code to print all the items displayed on moving mouse on “women,men,Baby&Kids” in flipkart application.

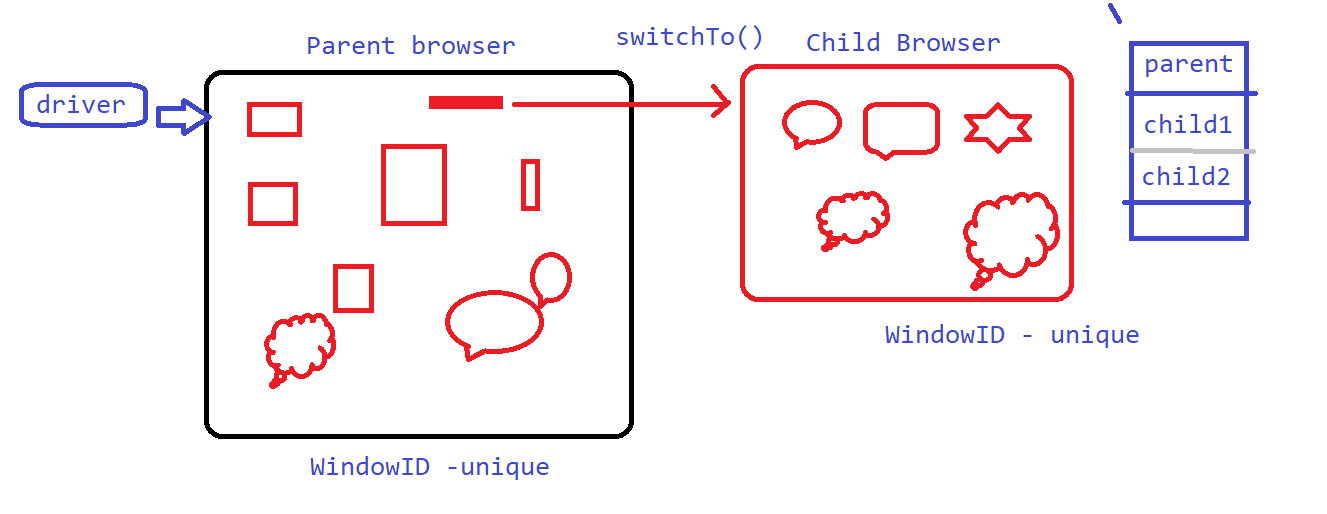
switchTo

switchTo()

frame

Alerts / popup Javascript

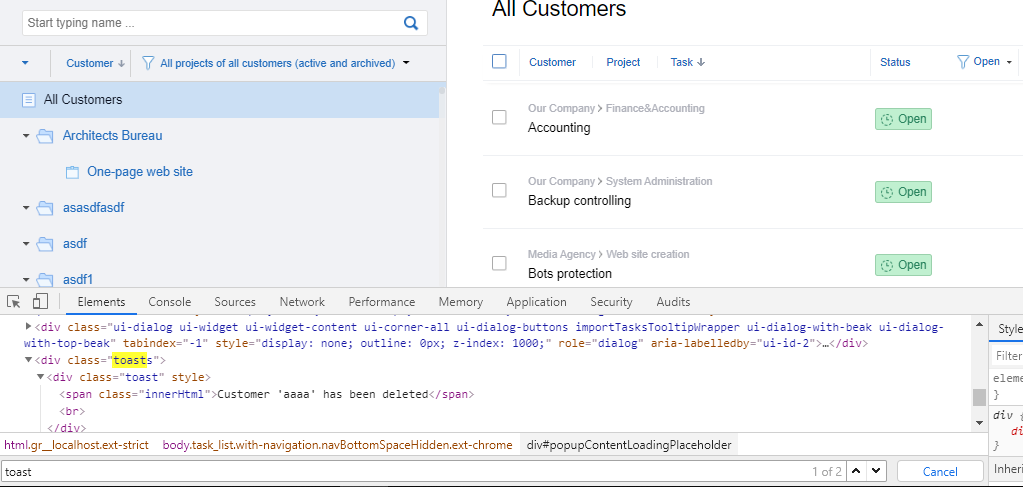
Multiple Browsers



End to End Automation:

1. Execute manual TestCase once or twice or thrice
   1. Identify the data required
   2. Identify the reusable functions that you can write
   3. Start writing reusable function
2. Start automating the tests by calling reusable functions.
3. Start Unit Testing of your test
4. Perform dry run (execute the test for couple of times)

Automating Toast Messages:

* You have to talk to developers regarding the properties
* Based on the String which is present on toast message you have to search in dom for the HTML CODE
* 

Data Driven testing:

DATA

Global Data

* url
* un
* pwd
* browser execution
* timeout

Local Data

* Data specific to test

Global data - *properties file or xml file*

* *Extension .properites*
* *<key>=<Value>*

Local data – excel, db, text, csv ....

Apache poi – library

**Apache poi->**

**Workbook –** represent excel file

**Sheet -**  One sheet in excel file

**Row –** one Row in excel file

**Cell –** represent one cell / column in a row

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0,0 | 0,1 | 0,2 |  |  |  |  |  |  |
| 1,0 | 1,1 | 1,2 |  |  |  |  |  |  |
| 2,0 | 2,1 | 2,2 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Sheet1 | Sheet2 | Sheet3 |  |  |  |  |  |  |

Handling PopUp:

alert() – JS -> use switchTo

confirmationPopup() – JS -> use switchTo

hiddenDivisionPopUP() - normal selenium code

cant handle using Selenium : (AutoIT, Sikuli, Robot)

--------------------------

pageLoadpopUp() - not from ur app

AuthenticationPopUp() - not from ur app

download popup ->browser

FrameWork :

* Function Driven Automation Framework
* Data Driven Automation Framework
* Keyword Driven Automation Framework
* Hybrid Driven Automation Framework
* **TestNG**
* **POM**

**Drawbacks of Selenium Without Framework :**

* **Execution Control**
* **Assert Statements in Selenium**
* **Mechanism to store the data**
* **Object Repository**
* **Reporting is not available**

TestNG

Annotations :

* BeforeSuite 🡪
* BeforeTest
* BeforeClass
* BeforeMethod
* Test
* AfterMethod
* AfterClass
* AfterTest
* AfterSuite

|  |  |  |  |
| --- | --- | --- | --- |
| Annotation | Execution | Purpose |  |
| beforeSuite | Only once before any annotation is executed | Creating Report |  |
| beforeTest | Only once before executing test tag in xml file | Data required for executing all tests under a module |  |
| beforeClass | Only once before executing class tag in xml | Pre condition to execute a test  Create driver/ browser obj only once |  |
| beforeMethod | Multiple times – before executing Test annotations | Launch and login functionality |  |
| Test | Actual Test case |  |  |
| Aftermethod | Multiple times– after executing Test annotations | Logout from app |  |
| afterClass | Executed only once after executing all classes | Close the driver |  |
| afterTest | Executed only once after executing all classes in test tag | Close the excel Object |  |
| afterSuite | Executed only once after executing all tests | Save the reports |  |

TAGS of xml :

