

Keyword-Driven Framework for Selenium and Protractor

Serguei Kouzmine
kouzmine_serguei@yahoo.com



Keyword-Driven Frameworks

The original Keyword Driven Framework few columns

- * **Summary:** brief description of the step
- * **Target:** name of the Web Page object/element, like "Link" or "Input"
- * **Action:** name of the action, which will be performed on Target Element such as click, open browser, input text etc.
- * **Data:** any value which is needed by the Object to perform any action, like text value for input field

<http://toolsqa.com/selenium-webdriver/keyword-driven-framework/introduction/>



Keyword-Driven Frameworks

Advantages from taking Keyword Driven Framework approach:

Less Technical Expertise: manual testers or non technical testers can easily write test scripts: no coding is exposed, test flow is easy to read and understand, keywords & actions are descriptive (implementation detail of the script is hidden from the test engineer). Keyword steps are easy to identify from requirements documentation or manual test steps.

Easy To Understand: One can start building Keyword Driven test cases immediately deferring technically challenging tasks to a later stage.

Re-usability: Stable and powerful Execution Engine in Keyword Driven Framework encourage code re-usability.

Automation: Excel file is (a lot) easier to produce by a recording tool than a full blown program.



Programming quiz, part I

```
import org.openqa.selenium.WebDriver;

WebDriver driver = new ChromeDriver();
WebDriverWait wait = new WebDriverWait(driver, 10);

driver.get("http://www.store.demoqa.com"); // open the start page

driver.findElement(By.xpath("//*[@id='account']/a")).click(); // go to login page
wait.until( driver ->
driver.getCurrentUrl().matches("http://store.demoqa.com/products-page/your-account/"));

driver.findElement(By.id("log")).sendKeys("testuser_3"); // log in
driver.findElement(By.id("pwd")).sendKeys("Test@123");
driver.findElement(By.id("login")).click();

// confirm the error message is displayed
assertThat(wait.until(ExpectedConditions.visibilityOf(driver.findElement(
    By.cssSelector("#ajax_loginform > p.response")))).getText(),
    containsString("The password is incorrect"));
driver.quit();
```



Programming quiz, part II

```
import io.webfolder.cdp.*;

SessionFactory factory = new Launcher().launch();
Session session = factory.create();

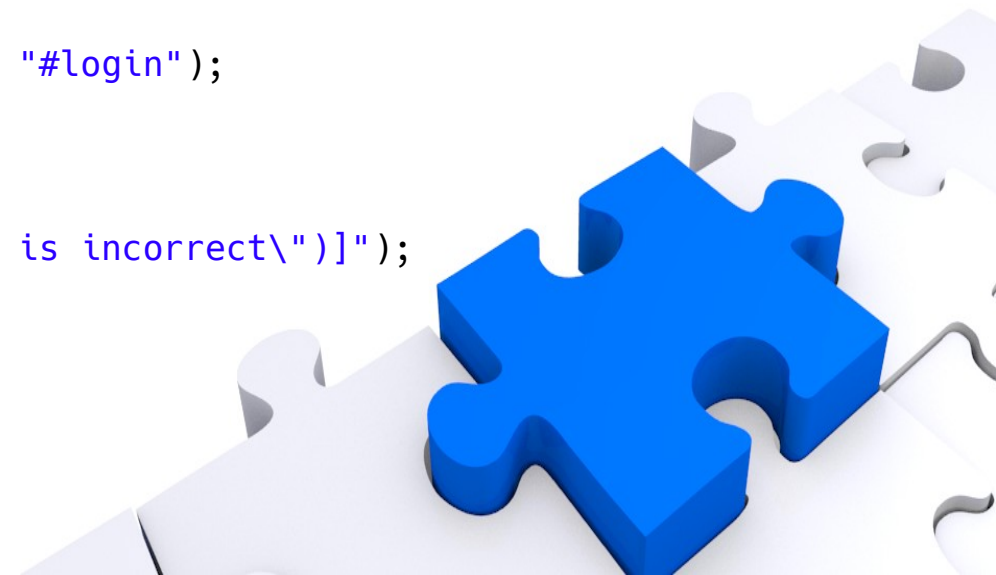
session.navigate("http://www.store.demoqa.com"); // navigate to start screen

session.click("#account > a"); // go to login page
session.waitFor(s -> s.getLocation().matches(
    "http://store.demoqa.com/products-page/your-account/"));

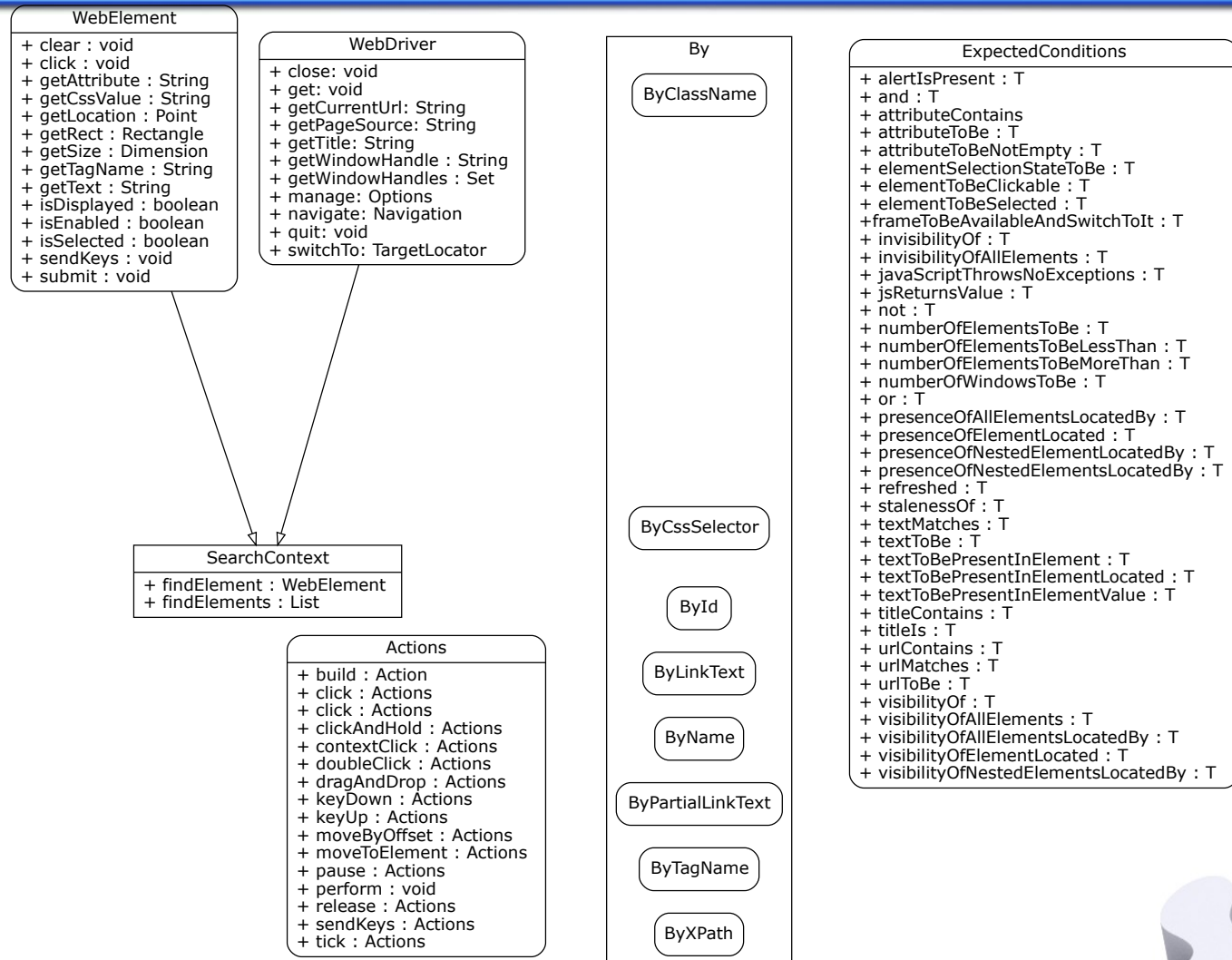
session.focus("#log"); session.sendKeys("testuser_3"); // log in
session.focus("#pwd").sendKeys("Test@123");
executeScript(session, "function(){this.click();}", "#login");

// confirm the error message is displayed
List<String> errMsgs = session.getObjectIds(
    "/*[contains (text(), \"The password you entered is incorrect\")]");
assertTrue(errMsgs.size() > 0);

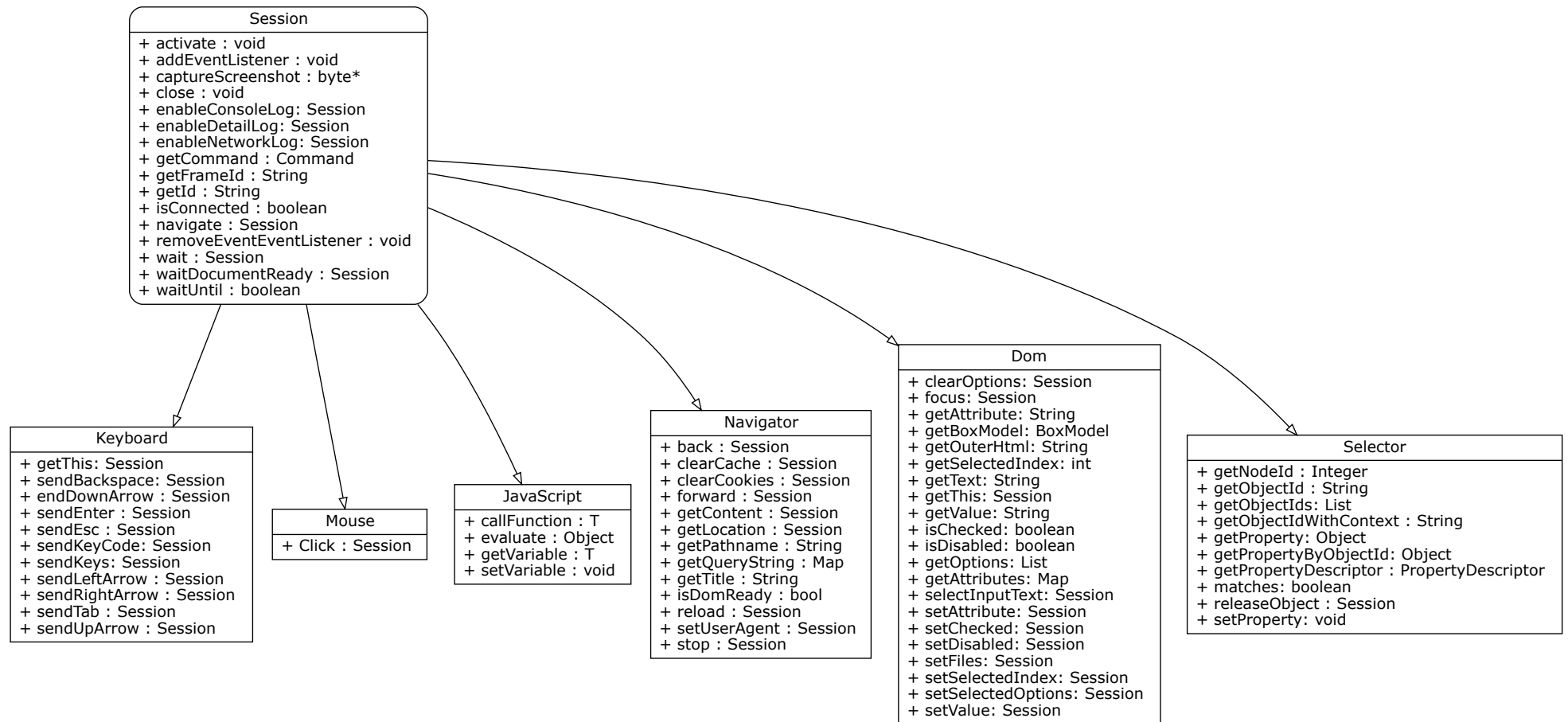
session.stop(); session.close();
```



Core Selenium Class Hierarchy



CDP Class Hierarchy



<https://webfolder.io/cdp4j/javadoc/index.html>

Java Protractor Wrapper Methods

Java Class Viewer

File Help

Expand All Collapse All

C:\developer\sergueik\keyword_driven_framework\ngwebdriver\target\lib\ngwebdriver-1.0.jar

META-INF

com

paulhammant

ngwebdriver

- ByAngular\$BaseBy.class
- ByAngular\$Factory.class
- ByAngular.class
- ByAngularBinding.class
- ByAngularButtonText.class
- ByAngularCssContainingText.class
- ByAngularExactBinding.class
- ByAngularModel.class
- ByAngularOptions.class
- ByAngularPartialButtonText.class
- ByAngularRepeater.class
- ByAngularRepeaterCell.class
- ByAngularRepeaterColumn.class
- ByAngularRepeaterRow.class
- NgWebDriver.class
- VariableNotInScopeException.class

js

C:\developer\sergueik\keyword_driven_framework\ngwebdriver\target\lib\ngwebdriver-1.0.jar - co

Expand All Collapse All

Class File

- magic
- minor_version: 0
- major_version: 49
- constant_pool_count: 256
- constant_pool
 - access_flags: public super
 - this_class: 52 - [Class: com.paulhammant.ngwebdriver.ByAngular]
 - super_class: 72 - [Class: java.lang.Object]
 - interfaces_count: 0
 - fields_count: 1
- fields
- methods_count: 15
- methods
 - method 1: private static void iterateOverJsFunctionsInSource (java.lang.String)
 - method 2: private static void inlineUtilityFunctionsIfNeeded ()
 - method 3: private static void storeJavaScriptFunction (java.lang.String)
 - method 4: private void <init> ()
 - method 5: public static com.paulhammant.ngwebdriver.ByAngular\$Factory withRootSelector (java.lang.String)
 - method 6: public static com.paulhammant.ngwebdriver.ByAngularRepeater repeater (java.lang.String)
 - method 7: public static com.paulhammant.ngwebdriver.ByAngularRepeater exactRepeater (java.lang.String)
 - method 8: public static com.paulhammant.ngwebdriver.ByAngularBinding binding (java.lang.String)
 - method 9: public static com.paulhammant.ngwebdriver.ByAngularExactBinding exactBinding (java.lang.String)
 - method 10: public static com.paulhammant.ngwebdriver.ByAngularModel model (java.lang.String)
 - method 11: public static com.paulhammant.ngwebdriver.ByAngularOptions options (java.lang.String)
 - method 12: public static com.paulhammant.ngwebdriver.ByAngularButtonText buttonText (java.lang.String)
 - method 13: public static com.paulhammant.ngwebdriver.ByAngularPartialButtonText partialButtonText (java.lang.String)
 - method 14: public static com.paulhammant.ngwebdriver.ByAngularCssContainingText cssContainingText (java.lang.String, java.lang.String)
 - method 15: static void <clinit> ()

Alternative Java Protractor Wrapper Methods



TestCase, core Selenium

	A	B	C	D	E	F	G	H	I	J
1	Keyword	param1	param2	param3	param4	param5	param6	param7	param8	status
2	CREATE_BROWSER									
3	GOTO_URL	https://www.google.com/gmail/about/#								
4	CLICK_LINK	xpath	//*[@data-g-label='Sign in']							
5	WAIT_URL_CHANGE	https://accounts.google.com/signin						120		
6	WAIT_ELEMENT_CLICKABLE	xpath	//*[@id='identifierNext']/content/span[contains(text(),'Next')]					120		
7	SET_TEXT	id	identifierId			automationnewuser24@gmail.com				
8	CLICK_BUTTON	xpath	//*[@id='identifierNext']/content/span[contains(text(),'Next')]							
9	WAIT							1000		
10	SET_TEXT	css	#password input			automationnewuser24@gmail.com				
11	CLICK_BUTTON	xpath	//*[@id='passwordNext']/content/span[contains(text(),'Next')]							
12	WAIT_URL_CHANGE	https://mail.google.com/mail						120		
13	CLOSE_BROWSER									

Each core Selenium locator accepts a *single* input argument of a specific *type*. Inspired by Selenium IDE Command, Target, Value columns, but with narrow choice of commands.

Table	Source	
Command	Target	Value
open	/en-US/firefox/addon/seleniu...	
clickAndWait	css=span.lcon.lcon-plus	
Command	clickAndWait	
Target	css=span.lcon.lcon-plus	Select Find
Value		



Test Case, Protractor

A	B	C	D	E	F	G	H	I
Keyword	param1	param2	param3	param4	param5	param6	param7	status
CREATE_BROWSER								Passed
GOTO_URL	http://juliemr.github.io/protractor-demo/							Passed
SET_TEXT	model	first			40			Passed
SET_TEXT	model	second			2			Passed
GET_ATTR	options	value for (key, value) in operators			innerHTML			Passed
CLICK_BUTTON	buttontext	Go!						Passed
GET_TEXT	binding	latest						Passed
CLOSE_BROWSER								

Protractor allows test operate the very same language (attributes) as Angular Web developer used when designing the page by finding elements by their *binding*, *model*, *repeater* etc.

It also provides Angular agnostic methods like *buttonText*, *cssContainingText*



Test Case, Protractor

	A	B	C	D	E	F	G	H	I
1	Keyword	param1	param2	param3	param4		param6	param7	status
2	CREATE_BROWSER								
3	GOTO_URL	http://amitava82.github.io/angular-multiselect/							
4	VERIFY_TAG	model	selectedCar			am-multiselect			
5	CLICK_BUTTON	buttontext	Select Some Cars						
6	WAIT							1000	
7	VERIFY_TEXT	repeaterElement	i in items	2	i.label	Honda			
8	CLICK_LINK	repeaterElement	i in items	1	i.label				
9	CLICK_LINK	repeaterElement	i in items	0	i.label				
10	WAIT							1000	
11	VERIFY_TEXT	css	am-multiselect > div > button			There are 2 car(s) selected			
12	CLOSE_BROWSER								

Protractor locates repeated elements relying on Angular attributes the individual elements and groups of elements were created on the web page by MVC. When the target element belongs to a set, additional coordinates are required to identify the set and the position of the member

The Keyword Table becomes *sparse*



Implementation



Keyword proceccing through Java Reflection

```
private static Map<String, String> methodTable = new HashMap<>();
static {
    methodTable.put("CLICK_BUTTON", "clickButton");
    ...
}
private static final Class<?> _class = Class.forName("Keyword Library class");

public static void callMethod(String keyword,
                               Map<String, String> _params;

    // for static methods
    _class.getMethod(methodTable.get(keyword), Map.class).invoke(null, _params);
    // for instance methods
    _class.getMethod(methodTable.get(keyword),
Map.class).invoke( _class.newInstance(), _params);
```



Loading Steps from Excel or OpenOffice spreadsheet

```
List<Object[]> steps = utils.createDataFromExcel2003(testCase);  
    // utils.createDataFromExcel2007(testCase);  
    // utils.createDataFromOpenOffice(testCase);  
  
for (int step = 0; step < steps.size(); step++) {  
    Object[] row = steps.get(step);  
    Map<String, String> data = new HashMap<>();  
    String keyword = (String) row[0];  
    for (int col = 1; col < row.length; col++) {  
        data.put(String.format("param%d", col), row[col].toString());  
    }  
    KeywordLibrary.callMethod(keyword, data);  
}
```



Keyword method implementanion

```
provate static WebDriver driver;  
private static NgWebDriver ngDriver;  
private static WebElement element;  
private static String status;  
private static String result;
```

```
public static void clickButton(Map<String, String> params) {  
    element = _findElement(params);  
    if (element != null) {  
        highlight(element);  
        element.click();  
        status = "Passed";  
    } else {  
        status = "Failed";  
    }  
}
```



Extracting arguments from params map

```
public static void verifyText(Map<String, String> params) {  
    boolean flag = false;  
    expectedText = params.get("param5");  
    element = _findElement(params);  
    if (element != null) {  
        highlight(element);  
        flag = element.getText().equals(expectedText);  
    }  
    if (flag)  
        status = "Passed";  
    else  
        status = "Failed";  
}
```



Find element using Selenium and Protractor

```
private static WebElement _findElement(Map<String, String> params) {  
    strategy = params.get("param1");  
    if (!strategies.containsKey(strategy)) {  
        throw new RuntimeException("Unknown strategy: " + strategy);  
    }  
    value = params.get("param2");  
    if (params.containsKey("param3")) {  
        row = params.get("param3");  
    }  
    if (params.containsKey("param4")) {  
        column = params.get("param4");  
    }  
  
    if (params.containsKey("param5")) {  
        containedText = params.get("param5");  
    }  
    if (params.containsKey("param6")) {  
        tagName = params.get("param6");  
    }  
}
```



Supported Locator Strategies

```
private static Map<String, Method> strategies = new HashMap<>();
strategies.put("css",
    By.class.getMethod("cssSelector", String.class));
strategies.put("id",
    By.class.getMethod("id", String.class));

strategies.put("binding",
    NgBy.class.getMethod("binding", String.class));
strategies.put("repeaterColumn",
    NgBy.class.getMethod("repeaterColumn", String.class,
        String.class));
strategies.put("repeaterElement",
    NgBy.class.getMethod("repeaterElement", String.class,
        Integer.class, String.class));

strategies.put("repeaterRow",
    methodMissing);
```



Phony method

```
// phony method
Method methodMissing = null;
try {
    @SuppressWarnings("rawtypes")
    Constructor<Method> methodConstructor = Method.class
        .getDeclaredConstructor(new Class[] { String.class,
            String.class });
    methodConstructor.setAccessible(true);
    methodMissing = (Method) methodConstructor.newInstance();
} catch (NoSuchMethodException |
        IllegalAccessException |
        InstantiationException |
        IllegalArgumentException |
        InvocationTargetException e) {
    System.out.println(e.toString()); // slurp
}
```



Find element using Selenium and Protractor, contd.

```
WebElement _element = null;

switch (strategy) {
  case "binding":
    _element = ngDriver.findElement(NgBy.binding(value));
    break;

  case "css":
    _element = driver.findElement(By.cssSelector(value));
    break;

  case "repeaterRow":
  case "repeaterRows":
    _element = ngDriver.findElement(NgBy.repeaterRows(
      value, row));
    break;
}
return _element;
}
```



Find element extending Selenium and Protractor, contd.

case "text":

// Option 1: construct XPath selector

```
Map<String, String> newParams = new HashMap<>();
newParams.put("param1", "xpath");
newParams.put("param2", String.format(
    "//%s[contains(normalize-space(text()), '%s')]",
    (tagName != null) ? tagName : "*", value));
```

```
_element = _findElement(newParams);
```

// Option 2: use Java streams/filter

```
if (tagName != null) {
    _element = driver.findElements(
        By.tagName(tagName)).stream().filter(
            o -> o.getText().contains(value))
        .findFirst().get();
}
break;
```

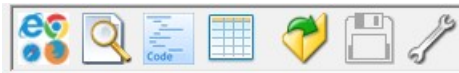


Finding elements with explicit wait

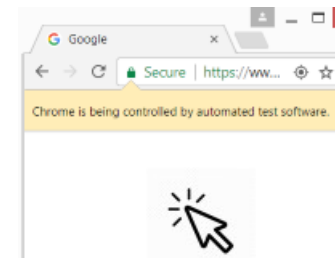
```
switch (strategy) {
    case "css":
        locator = By.cssSelector(selectorValue);
        break;
}
timeout = (long) (Float.parseFloat(params.get("param7")));
_wait = new WebDriverWait(driver, timeout);
_element = _wait.until(new ExpectedCondition<WebElement>() {
    @Override
    public WebElement apply(WebDriver d) {
        Optional<WebElement> e = d.findElements(locator)
            .stream()
            .findFirst();
        if (e.isPresent()) // log some debugging info
            logger.debug("find using strategy: " + strategy +
                " => " + e.get().getAttribute("outerHTML"));
        return (e.isPresent()) ? e.get() :
            (WebElement) null;
    }
});
```



Element Recorder Flow



```
// https://chromium.googlesource.com/chromium/src.git
getCssSelectorOf = function(element) {
  hello[getCssSelectorOf] = element.tagName;
  var specialAttributesArray = ['href', 'src', 'title', 'alt'];
  if (!element instanceof Element)
    return;
  var path = [];
  while (element.nodeType !== ELEMENT_NODE) {
    var selector = element.nodeName.toLowerCase();
    if (element.id && path.length != 0) {
      if (element.id.indexOf("-") > -1) {
        selector += "[id = '" + element.id + "']";
      } else {
        selector += "#" + element.id;
      }
      path.unshift(selector);
      break;
    } else if (element.className) {
      var attr = element.className;
      // ignore className attributes with special characters
    }
  }
}
```



Step 1: Step 2: Step 3: Step 4:



Element	Action Keyword	Selector Choice	Selector Value
step 1	CLICK_LINK	cssSelector	div#gs_lcl > input;
step 2	VERIFY_TEXT	xpath	div#rs0 > div_Nld >
step 3	VERIFY_ATTR	id	a[id = "js-link-box-n
step 4	SEND_KEYS	text	div[id = "mw-conten



```
import org.openqa.selenium.*
import org.openqa.selenium.chrome.*

// Selenium Code Selenium Java
// driver
// driver.get("http://www.seleniumhq.org")
// driver.findElement(By.id("id")).click()
// driver.findElement(By.id("id")).sendKeys("text")
// driver.findElement(By.id("id")).sendKeys(Keys.ENTER)
```



```
@Test
public void invalid() {
    // go to login
    driver.findElement(By.id("id")).click()
    wait.until(driver.findElement(By.id("id")).isDisplayed())
}
```



Flow explained

- Toolbar launches the Browser and lets it navigate to the target page
- Toolbar injects into the browser the Javascript Snippet, which attaches itself to Mouse Events (like Selenium IDE)
- When a Mouse Event is fired, Snippet helper functions are run and generate the CSS, Xpath, etc. Locators, extracts various attributes of the Target Element - in a similar fashion Firebug works.
- Toolbar continuously checks if the Target Element information is ready for pickup and when so, loads it, similar to how Selenium Wait works.
- The process repeats for the next Target Element in question.
- After the information about all elements was collected, Toolbar generates the Excel Flow and/or the QA Test Program utilizing a template(s)



Resources

- <https://github.com/angular/protractor>
- <https://github.com/bbaia/protractor-net>
- <https://github.com/paul-hammant/ngWebDriver>
- <https://github.com/caarlos0/jProtractor>
- <https://github.com/henrrich/jpagefactory>
- <https://github.com/sergueik/SKDF>
- <https://github.com/dzharii/swd-recorder>
- <https://github.com/sergueik/SWET>
- <https://github.com/ealves/protractor-recorder>
- <https://github.com/hanthomas/protractor-recorder>
- <https://github.com/dzharii/swd-recorder>
- https://github.com/sergueik/powershell_selenium_plugin

