TESTLEAF

Advanced User Interactions

TESTLEAF

Introduction

The Advanced User Interactions API is a new, more comprehensive API for describing actions a user can perform on a web page through Keyboard and Mouse Actions.

This includes actions such as drag and drop or clicking multiple elements while holding down the Control key

TESTLEAF

Keyboard Interactions

The new Interactions API takes a user-oriented approach, it is more logical to explicitly interact with the element before sending text to it, like a user would.

This means clicking on an element or sending a Keys.TAB when focused on an adjacent element.

Mouse Interactions

Mouse actions have a context - the current location of the mouse.

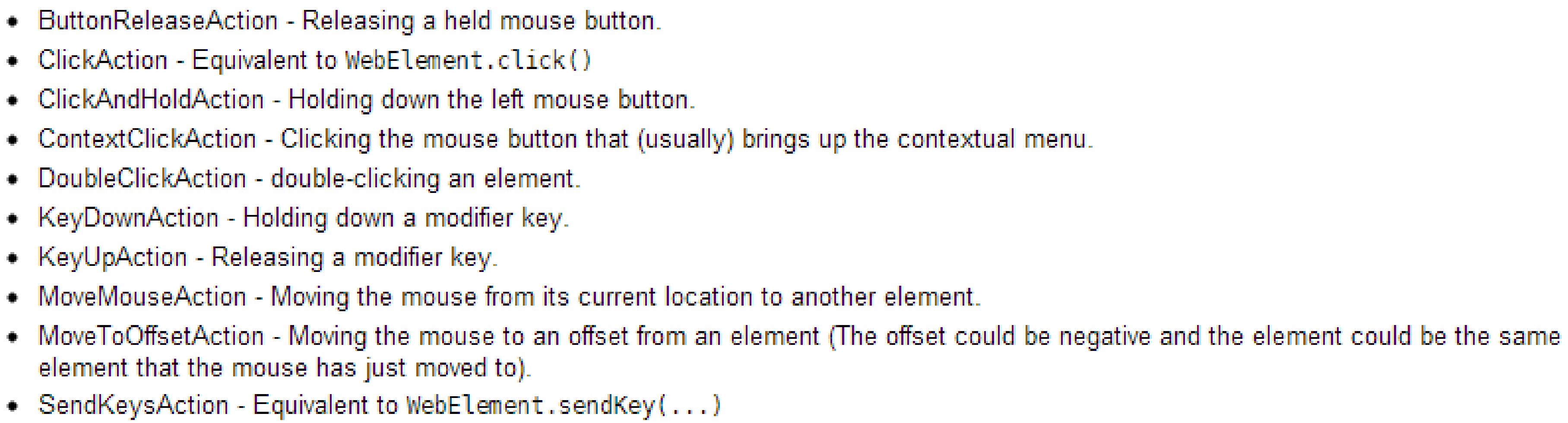
So when setting a context for several mouse actions (using onElement), the first action will be relative to the location of the element used as context, the next action will be relative to the location of the mouse at the end of the last action, etc.

Single Action

All actions implement the Action interface. This action only has one method: perform().

When invoked, the action then figures out how it should interact with the page and calls the underlying implementation to actually carry out the interaction.

Single Action - Methods



- ButtonReleaseAction - Releasing a held mouse button.
- ClickAction - Equivalent to `WebElement.click()`
- ClickAndHoldAction - Holding down the left mouse button.
- ContextClickAction - Clicking the mouse button that (usually) brings up the contextual menu.
- DoubleClickAction - double-clicking an element.
- KeyDownAction - Holding down a modifier key.
- KeyUpAction - Releasing a modifier key.
- MoveMouseAction - Moving the mouse from its current location to another element.
- MoveToOffsetAction - Moving the mouse to an offset from an element (The offset could be negative and the element could be the same element that the mouse has just moved to).
- SendKeysAction - Equivalent to `WebElement.sendKey(...)`

Multiple Actions

The Actions chain generator implements the Builder pattern to create a CompositeAction containing a group of other actions.

This should ease building actions by configuring an Actions chains generator instance and invoking it's build() method to get the complex action.

A planned extension to the Actions class is adding a method that will append any Action to the current list of actions it holds. This will allow adding extended actions without manually creating the CompositeAction

Keyboard Interface

The Keyboard interface has three methods:

void sendKeys(CharSequence... keysToSend) - Similar to the existing sendKeys(...)

void pressKey(Keys keyToPress) - Sends a key press only, without releasing it. Should only be implemented for modifier keys (Control, Alt and Shift).

void releaseKey(Keys keyToRelease) - Releases a modifier key.

The element which will receive those events is the active element.

Mouse Interface

The Mouse interface includes the following methods

void click(WebElement onElement) - Similar to the existing click() method.

void doubleClick(WebElement onElement) - Double-clicks an element.

void mouseDown(WebElement onElement) - Holds down the left mouse button on an element.

void mouseUp(WebElement onElement) - Releases the mouse button on an element.

void mouseMove(WebElement toElement) - Move (from the current location) to another element.

void mouseMove(WebElement toElement, long xOffset, long yOffset) - Move (from the current location) to new coordinates: (X coordinates of toElement + xOffset, Y coordinates of toElement + yOffset).

void contextClick(WebElement onElement) - Performs a context-click (right click) on an element.

Sample Code to perform drag & drop

```
Actions builder = new Actions(driver);


Action dragAndDrop = builder.clickAndHold(someElement)
       .moveToElement(otherElement)
       .release(otherElement)
       .build();

 dragAndDrop.perform();
```

Sample Code to select multi value combo

```
Actions builder = new Actions(driver);
List<WebElement> listItems = driver.findElements(By .cssSelector("ol#selectable *"));
Actions builder = new Actions(driver);
builder.clickAndHold(listItems.get(1)).clickAndHold(listItems.get(2)).click();
Action selectMultiple = builder.build();
selectMultiple.perform();
```