Selenium 2.0 + Python

Обзор команд Selenium WebDriver

Занятие 2

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

Общий обзор команд

http://selenium.googlecode.com/git/docs/api/py/index.html

- браузер – запуск, остановка
- окно браузера – открыть, закрыть, переключиться, размер
- диалоговые окна, ~~меню, тулбар, статусбар, плагины~~
- страница – открыть, предыдущая, следующая, выполнить JS-код
- элементы – найти, мышь, клавиатура, текст, атрибуты, стили, размер
- фреймы – найти, переключиться

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

Запуск браузера

```
from selenium import webdriver
driver = webdriver.Firefox()
driver = webdriver.Chrome()
driver = webdriver.Ie()
driver = webdriver.Opera()
driver = webdriver.PhantomJS()
driver = webdriver.Remote()
```

Запуск браузера

- Google Chrome

  https://code.google.com/p/chromedriver/downloads/list
- Internet Explorer

  https://code.google.com/p/selenium/downloads/list
- PhantomJS

  https://code.google.com/p/phantomjs/downloads/list

Запуск браузера

```
from selenium import webdriver
driver = webdriver.Firefox(
    capabilities={'native_events': False})
```

https://code.google.com/p/selenium/wiki/DesiredCapabilities

Остановка браузера

driver.quit()

закрыть все окна и завершить работу

driver.close()

закрыть текущее окно,
если оно последнее – завершить работу

Открытие страниц и навигация

driver.get("http://selenium2.ru/")

открыть страницу

и подождать, пока она загрузится

driver.back() driver.refresh() driver.forward()

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов и ожидания
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

Базовые команды поиска

element = driver.find_element(by, locator)

найти первый элемент по заданному условию

elements = driver.findElements(by, locator)

найти все элементы по заданному условию

Типы локаторов

- By.ID
- By.NAME
- By.CSS_SELECTOR
- By.XPATH
- By.TAG_NAME
- By.CLASS
- By.LINK_TEXT
- By.PARTIAL_LINK_TEXT

Типизированные команды поиска

```
driver.find_element_by_id('myid')
driver.find_element_by_name('myname')
driver.find_element_by_xpath('//a')
driver.find_element_by_css_selector('a')
driver.find_element_by_tag_name('a')
```

Поиск внутри элемента

element2 = element1.find_element(by, locator)

найти первый элемент по заданному условию, находящийся внутри элемента element1

elements = element1.find_elements(by, locator)

найти все элементы по заданному условию, находящиеся внутри элемента element1

Пример

```
cell41 =
    driver.find_element_by_css_selector("table.t1")
    .find_elements_by_tag_name("tr")[3]
    .find_elements_by_tag_name("td")[0]
```

это первая ячейка в четвёртой строке таблицы, имеющий класс t1

Если ничего не нашлось…

find_element

выбрасывает исключение NoSuchElementException

find_elements

возвращает пустой список

Как проверить наличие?

```
def is_element_present(self, by, locator) {
    try:
        self.driver.find_element(by, locator)
    except NoSuchElementException, e:
        return False
    return True
```

Как проверить наличие?

```
def is_element_present(self, by, locator) {
    return len(
        self.driver.find_elements(by, locator)) > 0
```

то же самое, даже по скорости одинаково!

Неявное ожидание

```
driver.implicitly_wait(10)
```

- find_element ждёт, пока элемент появится
- find_elements ждёт, пока **хотя бы один** элемент появится

Явное ожидание

```
for i in range(60):
    try:
        if self.is_element_present(by, locator): break
    except: pass
    time.sleep(1)
else:
    self.fail("time out")
```

Явное ожидание

```
from selenium.webdriver.support.wait import
    WebDriverWait

wait = WebDriverWait(driver, 30)

element = wait.until(
    lambda x: x.find_element(by, locator))
```

Явное ожидание

```
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support.expected_conditions import *

wait = WebDriverWait(driver, 30)

element = wait.until(
    presence_of_element_located((by, locator))
```

http://selenium.googlecode.com/git/docs/api/py/_modules/selenium/webdriver/support/expected_conditions.html

Явные и неявные ожидания

Явные

- На стороне клиента
- Ждать можно чего угодно
- Надо писать явно
- TimeoutException
- Много сетевых запросов

Неявные

- На стороне браузера
- Ожидание появления в DOM
- Работают автоматически
- NoSuchElementException
- Один сетевой запрос

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Окнами и фреймами

«Простые» действия

- кликнуть
  - ссылки и кнопки
  - радио и чекбоксы
  - списки
- ввести текст
- прицепить файл

«Сложные» действия

- клавиатурные сочетания
- наведение мыши
- перетаскивание
- правая кнопка мыши
- двойной клик

Две главные операции

click

- любой видимый элемент
- клик в центр элемента
- срабатывают все события

send_keys

- любой видимый элемент
- добавление в конец текста
- срабатывают все события
- прицепляет файлы
- сочетания клавиш
- работают стрелки и т.д.

А другие операции?

- select ?
  это просто клик по элементу списка
- toggle ?
  это просто клик по чекбоксу
- check / uncheck ?
  клик с предварительной проверкой

Тем не менее, Select

```
from selenium.webdriver.support.select import Select

dropdown = Select(element)

dropdown.select_by_index(1)
dropdown.select_by_value("mon")
dropdown.select_by_visible_text("Monday")
```

Пример

```
from selenium.webdriver.common.keys import Keys

element.click()
element.send_keys(Keys.HOME)
element.send_keys("some text")
```

добавить текст в начало, а не в конец

Пример

```
element.send_keys(Keys.CONTROL, "a")
time.sleep(1)
element.send_keys(Keys.DELETE)
```

удалить содержимое поля ввода

Пример

```
element.send_keys("text to search")
element.send_keys(Keys.RETURN)
```

засабмитить форму, если нет кнопки

Пример

```
body =
    driver.find_element_by_tag_name("body")
body.send_keys(
    Keys.CONTROL, Keys.SHIFT, "1")
```

«горячие клавиши», клавиатурные сочетания

Пример

```
set_clipboard_contents(longtext)
textarea.send_keys(Keys.CONTROL , "v")
```

вставить длинный текст из буфера обмена

Нативные и синтезированные события

Нативные

- На уровне ОС
- Реализация на C/C++
- Точнее эмулируют
- Иногда требуют фокус
- Не все версии браузеров

Синтезированные

- Внутри браузера
- Реализация на JavaScript
- Не всегда точно эмулируют
- Работают в бэкграунде
- Все версии всех браузеров

Запуск браузера

```
from selenium import webdriver
driver = webdriver.Firefox(
    capabilities={'native_events': True})
```

https://code.google.com/p/selenium/wiki/DesiredCapabilities

Actions

- click
- send_keys
- move_to_element
- click_and_hold
- release
- key_down
- key_up

```
webdriver.ActionChains(driver)
    .move_to_element(drag)
    .key_down(Keys.CONTROL)
    .click_and_hold()
    .move_to_element(drop)
    .release()
    .key_up(Keys.CONTROL)
    .perform()
```

Actions

```
webdriver.ActionChains(driver)
    .move_to_element(el, 1, 1)
    .click()
    .perform()
```

```
webdriver.ActionChains(driver)
    .move_to_element(el)
    .move_by_offset(5,5)
    .click()
    .perform()
```

```
webdriver.ActionChains(driver)
    .move_to_element(menu)
    .move_to_element(submenu)
     .move_to_element(item)
    .click()
    .perform()
```

```
webdriver.ActionChains(driver)
    .drag_and_drop(el1, el2)
    .perform()
```

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

text

- **Видимый** текст
  невидимые элементы имеют пустой текст

- Нормализация – удаление пробелов
  Preformatted – сохранение пробелов

get_attribute

- input.get_attribute("value");
- input.get_attribute("href");
  всегда абсолютные ссылки
- button.get_attribute("disabled")
  либо null, либо true
  disabled, selected, checked, readonly, …
- div.get_attribute("innerText");
  attribute или property?

is_displayed

- вроде бы всё очевидно, но…
- находится за левым или верхним краем *частично* находится за краем
- скрыт под другим элементом *частично* скрыт под другим элементом
- прозрачный, либо цвет сливается с фоном

Что ещё?

- element.value_of_css_property("color")
- element.size()
- element.location()
- element.get_tag_name()
- element.is_enabled()
- element.is_selected()

План занятия

- Общий обзор команд
- Запуск и остановка, открытие страниц
- Поиск элементов (+ожидания)
- Действия с элементами, простые и сложные
- Получение свойств элементов
- Переключение между окнами и фреймами

driver.switch_to_...

- driver.switch_to_alert()

- driver.switch_to_frame()
- driver.switch_to_default_content()

- driver.switch_to_window()

Диалоговые окна

```
alert = driver.switch_to_alert()
alert_text = alert.text()
alert.accept()    # либо alert.dismiss()
```

а если нет алёрта? NoAlertPresentException

а если не сделать? UnhandledAlertException

Фреймы

```
driver.switch_to_frame(
    driver.find_element_by_tag_name("iframe"))
# что-то сделали внутри фрейма
driver.switch_to_default_content()
```

фреймы могут быть вложены как матрёшка

Окна

```
all_windows = driver.window_handles()
this_window = driver.current_window_handle()

driver.switch_to_window(handle)
# что-то делаем в этом окне
driver.close()
driver.switch_to_window(original_window)
```

Туда и обратно

```
# запоминаем идентификатор текущего окна
original_window = driver.current_window_handle()
# запоминаем идентификаторы уже открытых окон
existing_windows = driver.window_handles()
# кликаем кнопку, которая открывает новое окно
driver.find_element_by_id("button").click()
# ждем появления нового окна, с новым идентификатором
new_window = wait.until(any_window_other_than(existing_windows))
# переключаемся в новое окно
driver.switch_to_window(new_window)
# закрываем его
driver.close()
# и возвращаемся в исходное окно
driver.switch_to_window(original_window)
```

Ожидание появления нового окна

```
class any_window_other_than(object):
    def __init__(self, existing_windows):
        self.existing_windows = existing_windows

    def __call__(self, driver):
        handles = driver.window_handles()
        diff = set(handles)-set(existing_windows)
        return iter(diff).next() if len(diff) > 0 else False
```

Размеры и положение окна

```
driver.get_window_size()
driver.set_window_size(800, 600)
driver.maximize_window()

driver.get_window_position()
driver.set_window_position(0, 0)
```

- На этом пока всё

- «Домашка»
- Форум
- Скайп-чат