Черкашин Алексей Васильевич
Магистрант
Уральский федеральный университет
Екатеринбург, Россия

МЕТОДЫ АВТОМАТИЗОРОВАННОЙ РАБОТЫ С СИСТЕМОЙ SAP CRM

Аннотация: *В данной статье описаны методы для автоматизированной работы с системой SAP CRM имитирующие работу реального пользователя, позволяющие уменьшить дублируемость кода и увеличить скорость выполнения автоматизированных тестов, реализованные с помощью стандартных средств Python и библиотеки для работы с браузером.*

Ключевые слова: *Тестирование, python, pytest, POM, Selenium, SAP.*

Для автоматизированной работы с веб приложениями, в том числе автоматизированное тестирование позволяет сократить трудозатраты на выполнение рутинных действий, что значит сэкономить финансовые ресурсы организации, а также во многих случаях увеличить скорость выполнения тех или иных операций. Стандартом для взаимодействия с браузером является инструмент Selenium WebDriver, по сути это — универсальный интерфейс, который позволяет манипулировать разными браузерами напрямую из кода на языке программирования как показано на рисунке 1. Для работы Selenium WebDriver непосредственно с браузером используются драйвера для конкретного браузера, которые предоставляют разработчики этих браузеров.

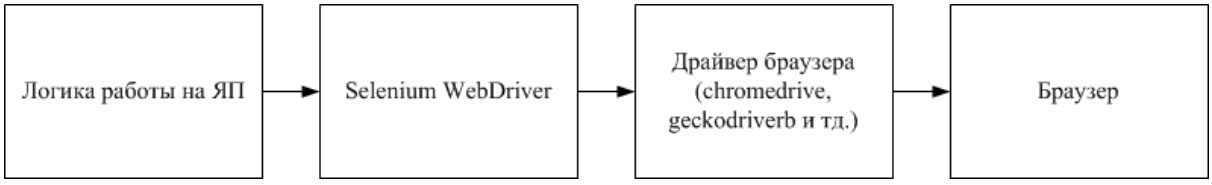

Рисунок 1. Взаимосвязь сценариев и браузера.

Selenium WebDriver предоставляет основные методы для работы с браузером, такие как поиск элемента, нажатие, ввод символов и прочее, однако

очень часто этих методов недостаточно, особенно это свойственно для сложных веб приложений, как например CRM система.

Целью данной статьи является определение оптимальных методов и алгоритмов для автоматизации основных действий в данной системе используя инструментарий языка программирования Python и некоторых его библиотек.

Реализовано разработать четыре основных алгоритма, которые соответствуют наиболее часто используемым действиям при тестировании CRM системы. Для реализации этих методов использованы возможности библиотеки selenium и встроенные возможности языка.

- **Переход по фреймам** - необходимость в данном методе обусловлена тем, что одна из особенностей указанной системы разделение приложения на части используя фреймы, таким образом для доступа к какому-либо элементу необходимо в нужный фрейм и лишь замет необходимый элемент будет в области видимости. Для автоматизации тестирования система фреймов плоха тем, что поиск элементов в Selenium осуществляется, опираясь на DOM-модель. По умолчанию это всегда DOM страницы. Для того, чтобы начать работать с контентом фрейма нужно указать драйверу с какой DOM-моделью необходимо работать (у каждого фрейма своя DOM модель). Для того, чтобы начать работать с контентом фрейма нужно указать драйверу с какой DOM-моделью необходимо работать. В качестве параметров он принимает индекс, имя или сам iframe элемент. Для перехода к нужному фрейму необходимо использовать метод browser.switch_to.frame из библиотеки selenium, для перехода в родительский фрейм используется метод browser.switch_to.default_content.

- **Поиск элемента** - как было сказано выше, при переходе на другую страницу или другой функциональный модуль значит лишь переход на другую страницу текущего фрейма, в то время оставаясь в родительском фрейме, то есть основная страница не обновляется, что лишает возможности привязаться к обновлению страницы как к точке отсчета и усложняет поиск элемента. Для решения данной проблемы решено использовать инструмент явного ожидания из библиотеки selenium, позволяющий задать время в течении которого будет

ожидаться появление элемента на странице. Явное ожидание используется в связке с правилами, используя правило element_to_be_clickable будет производится поиск элемента на который можно нажать, так как не всегда появление какого либо элемента равно активности этого элемента. Так же, для правильной обработки исключения TimeoutException, которое возникает в случае если в течении заданного времени элемент все же не найден, использована конструкция try...except.

- **Поиск элемента в модульном окне** - так как для реализации некоторых функций используется вызов модульных окон необходимо правильно обрабатывать информацию в них. Для перехода на новую вкладку браузера, в том числе и на новое окно, чем по сути и является модульное окно, используется метод browser.switch_to.window(window_name), а для получения списка доступных вкладок и окон используется метод browser.window_handles. Поиск элемента в модульном окне аналогичен обычному поиску элемента.
- **Проверка наличия заданного текста** - для проверки наличия текста в элементе так же использовано явное ожидание expected_conditions совместно с правилом text_to_be_present_in_element, которое проверяет присутствует ли заданный текст в заданном элементе.

Для минимизации дублируемости кода использованы принципы объектно-ориентированного программирования, такие как инкапсуляция и наследования. Описанные методы определены в базовом классе и наследуются в дальнейшем во все классы, в которых необходимо их применение. В случае автоматизированных тестов, когда код организован согласно паттерну POM(Page Object Model) эти методы определены в base page.

Результатом являются методы позволяющие уменьшить дублируемость кода, а также, в некоторых случаях увеличить скорость выполнения тестов за счет того, что необходимо выполнять меньшее количество переходов по фреймам, а делать это лишь по необходимости. Схема фреймов представлена на рисунке 2.

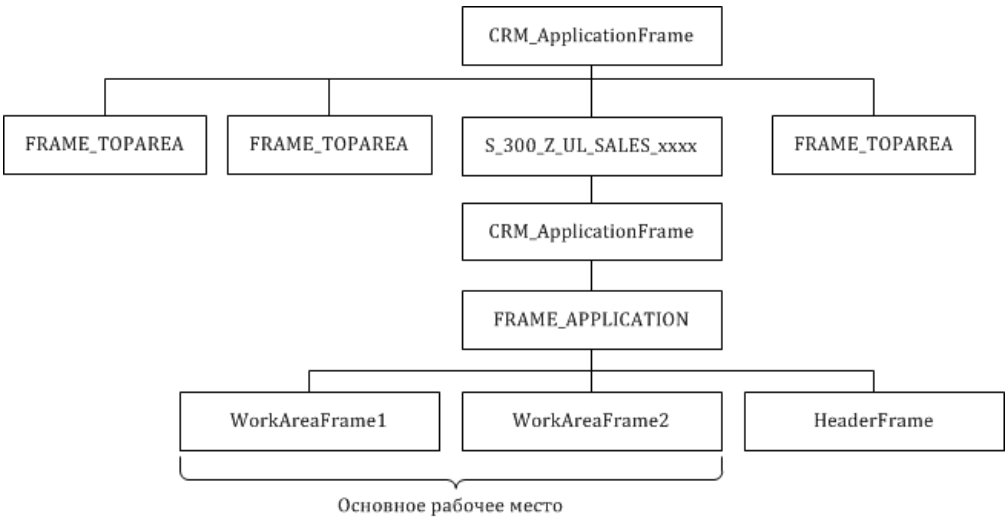

Рисунок 2. Схема фреймов

Для последовательного перехода по ним реализован метод sap_to_frame, представленный в листинге 1.

Листинг 1. Метод перехода по фреймам.

```
def sap_to_frame (self, area=1):
  self.browser.switch_to.frame («CRMApplicationFrame»)
  self.browser.switch_to.frame (3)
  self.browser.switch_to.frame («CRMApplicationFrame»)
  self.browser.switch_to.frame («FRAME_APPLICATION»)
  self.browser.switch_to.frame («WorkAreaFrame» + str (area))
```

В качестве аргумента метод принимает номер конечного фрейма, для перехода в WorkAreaFrame1 или WorkAreaFrame2. По умолчанию выполняется переход в WorkAreaFrame1, так как чаще всего рабочая область находится в этом фрейме.

Для поиска элемента необходимо, во-первых, выполнить поиск в нескольких фреймах, а во-вторых, в случае отсутствия его на странице, подождать заданное время его появления. Для этого разработан алгоритм, представленный на рисунке 3 и метод, представленный в листинге 2.

Листинг 2. Метод поиска элемента.

```
def sap_find (self, by, value):
```

```
  try:
      element = WebDriverWait(self.browser, self.timeout).until
(EC.element_to_be_clickable ((by, value)))
      return element
  except TimeoutException:
      try:
          self.browser.switch_to.default_content()
          self.sap_to_frame(1)
          element = WebDriverWait(self.browser,
self.timeout).until(EC.element_to_be_clickable((by, value)))
          return element
      except TimeoutException:
          try:
              self.browser.switch_to.default_content ()
              self.sap_to_frame(2)
              element = WebDriverWait(self.browser,
self.timeout).until(EC.element_to_be_clickable ((by, value)))
              return element
          except TimeoutException:
              print(«Элемент не найден!»)
```

В качестве аргументов метод принимает способ поиска элемента и значение, по которому необходимо производить поиск. В случае если элемент не найден не в одном из фреймов в консоль выводится сообщение «Элемент не найден!».

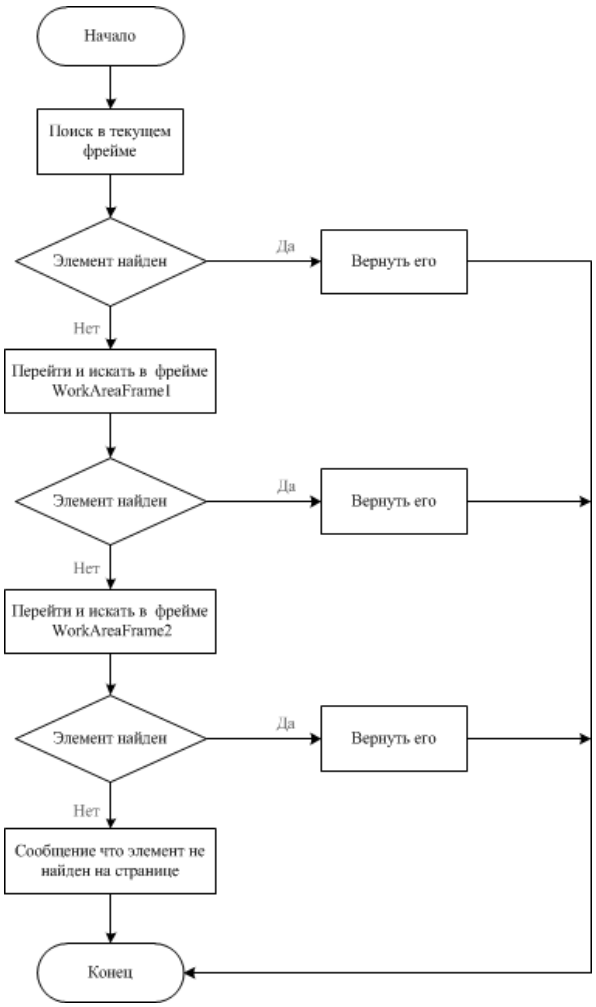

Рисунок 3. Алгоритм поиска элемента

Для поиска элемента в модульном окне необходимо перейти в это окно, так как модульное окно считается отдельной страницей со совим DOM-деревом. Модульное окно воспринимается вебдрайвером как отдельное окно, для перехода в него неоюходимо получить список доступных окон браузера и выполнить переход в нужны, а затем перейти в нужное. Для этого разработан метод, представленный в листинге 3.

Листинг 3. Метод поиска в модульном окне.

```
def sap_find_module (self, by, value, n=1):
  try:
    new_window = self.browser.window_handles[n]
    browser.switch_to.window(new_window)
    self.browser.switch_to.frame(«WorkAreaFrame1»)
```

```
    element = WebDriverWait(self.browser,
self.timeout).until(EC.presence_of_element_located((by, value)))
    return element
  except NoSuchElementException:
    print («Нет элемента»)
  finally:
    browser.switch_to.window(self.browser.window_handles [0])
```

В качестве аргументов метод принимает способ поиска элемента и значение, по которому необходимо производить поиск и номер окна, в котором необходимо производить поиск. После выполнения поиска и возвращения элемента или даже при неудачном поиске элемента управление возвращается к родительскому окну, с индексом 0. В модульных окнах существует только два фрейма - родительский и WorkAreaFrame1.

Метод для поиска подобен методу поиска элемента sap_find, но очень необходим для построения проверок (assertion). Алгоритм работы этого метода также совпадает с рисунком 3, за исключением того, что проверяет не только наличие элемента, а еще и наличие указанного теста в этом элементе. Реализация этого метода показана на листинге 4.

Листинг 4. Метод проверки наличия текста.

```
def sap_find_text(self, by, value, text):
  try:
    result = WebDriverWait(self.browser,
self.timeout).until(EC.text_to_be_present_in_element((by, value), text))
    return result
  except TimeoutException:
    try:
      self.browser.switch_to.default_content()
      self.sap_to_frame(1)
      result = WebDriverWait(self.browser,
```

```
self.timeout).until(EC.text_to_be_present_in_element((by, value), text))
            return result
        except TimeoutException:
            try:
                self.browser.switch_to.default_content()
                self.sap_to_frame(2)
                result = WebDriverWait(self.browser,
self.timeout).until(EC.text_to_be_present_in_element((by, value), text))
                return result
            except TimeoutException:
                print('Указанного текста в элементе не найдено')
                return False
```

В качестве аргументов метод принимает способ поиска элемента и значение, по которому необходимо производить поиск и текст, наличие которого необходимо проверять. В случае если текст в элементе не найден ни в одном из фреймов метод возвращает значение False и выводи в консоль сообщение «Указанного текста в элементе не найдено».

Результатом работы являются методы, позволяющие эффективно автоматизировать действия в системе SAP CRM и сократить время на реализацию автоматизированных тестов за счет переиспользования методов, определенных в базовом классе.

Помимо этого, использование этих методов позволяет непосредственно ускорить выполнение этих тестов, что для автоматизированных тестов очень важно. Например, описанный метод по поиску элементов по сравнению алгоритмом, когда мы переходим во фрейм и там ищем элемент, затем ищем в другом, сильно выигрывает в тестах, где большое количество шагов. По собранным данным построен график, который построен на рисунке 4.

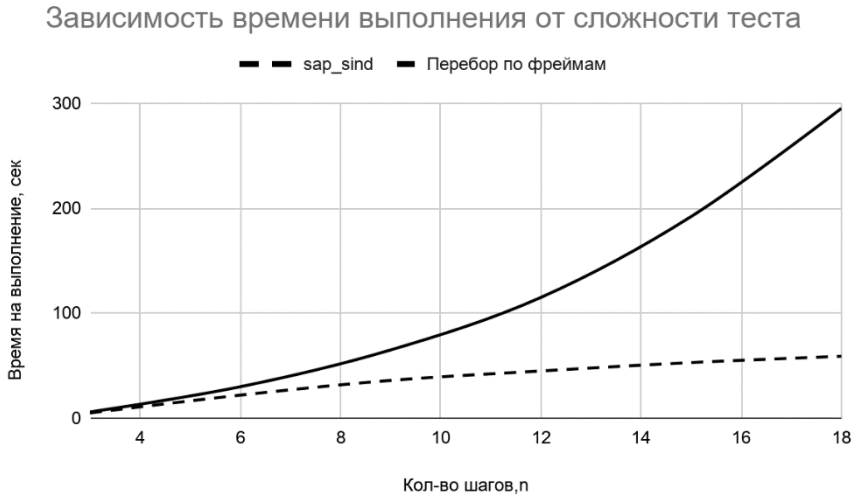

Рисунок 4. Зависимость времени выполнения от сложности теста.

Из графика на рисунке 4 видно, что зависимость времени выполнения теста от сложности теста, то есть от количества шагов в нем, при использовании описанного метода sap_find имеет довольно линейный характер, в то время как при использовании перебора по фреймам зависимость имеет очень нелинейный характер и время на выполнение теста с увеличением шагов нелинейно увеличивается.

Таким образом описанные методы позволяют облегчить написание тестов и ускорить их выполнение.

БИБЛИОГРАФИЧЕКИЙ СПИСОК

1. Full pytest documentation[Электронный ресурс]. – URL: https://docs.pytest.org/en/latest/contents.html (Дата обращения: 22.01.2020)

2. Page Objects for Python[Электронный ресурс]. – URL: https://page-objects.readthedocs.io (Дата обращения: 24.01.2020)

3. Python 3.8.2rc1 documentation[Электронный ресурс]. – URL: https://docs.python.org/3/ (Дата обращения: 24.01.2020)

4. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

5. Selenium with Python[Электронный ресурс]. – URL: https://selenium-python.readthedocs.io (Дата обращения: 23.01.2020)

6. How Browsers Work: Behind the scenes of modern web browsers [Электронный ресурс]. – URL: https://www.html5rocks.com/ут/tutorials/internals/howbrowserswork. (Дата обращения: 20.01.2020)