**Автоматизация работы с браузером с помощью Selenium**

*Воробьева Марья*

*09.12.2019*

Доклад был подготовлен в рамках практических семинаров НУГ, посвящённых изучению различных методов вэб-скрапинга данных. В данном докладе рассказывалось о возможностях автоматической работы с браузером, разбирались примеры взаимодействия с сайтом и парсинга текстовой информации.

Взаимодействовать с сайтом в автоматическом режиме позволяет веб-драйвер, он имитирует действия пользователя. На семинаре был рассмотрен веб-драйвер для работы с Google Chrome. Установить веб-драйвер необходимо на свой компьютер, это возможно сделать с помощью скачивания необходимых установщиков с официального сайта. Файлы располагаются в открытом доступе для каждого из популярных веб-браузеров.

Одной из возможностей библиотеки selenium, реализуемой на python, выступает локализация и извлечение необходимой информации с сайта. Использование selenium для этих целей имеет особую пользу, если информация на страницах сайта представлена в виде прокрутки и загрузки новых объектов. Информация на таких сайтах загружается только при взаимодействии пользователя с сайтом, selenium же позволяет автоматизировать это взаимодействия, и, таким образом, полностью собрать данные – все загружаемые объекты. Локализация и извлечение информации реализуется с помощью класса By. Существует несколько возможностей нахождения нужного элемента структуры кода страницы: css selector, class name, id name, xpath и др. Необходимая команда для поиска – find\_element\_by\_(css\_selector/class\_name/id/xpath). На семинаре был рассмотрен метод поиска через css selector. Можно отметить, что существует также функция find\_element**s**\_by, которая позволяет находить **все** подобные элементы на странице и возвращать их в виде списка.

Взаимодействие с объектами на сайте, нажатие на определённый структурный элемент реализуется с помощью функции click(). Например: driver.find\_element\_by\_css\_selector("div.top-bar > div > a").click(). Помещение объекта в другой объект, например, строку поиска реализуется с помощью команды send\_keys(“text”). Например, driver.find\_element\_by\_css\_selector("input[type='search']").send\_keys(“London”). Извлечение содержимого атрибутов, например, ссылок на другие страницы сайта может быть выполнено с помощью команды get\_attribute. Например, driver.find\_elements\_by\_css\_selector("a.story-block\_\_clickable").get\_attribute('href'). Извлечение текстового содержимого элемента возможно с помощью команды text. Например, driver.find\_elements\_by\_css\_selector("h3.story-block\_\_title").text.

Во время семинара был также рассмотрен пример взаимодействия с динамическими элементами сайта, в частности, закрытие всплывающих окон. Ожидание появления всплывающих элементов или просто замедление работы с сайтом возможно в двух вариантах: explicit and implicit. В первом случае мы ждём выполнения определённого условия. Во втором случае просто задаём, какое количество времени в секундах надо ждать. На семинаре был рассмотрен первый пример подобного взаимодействия. Для реализации данной возможности необходимо импортировать два модуля: WebDriverWait и expected\_conditions. Код: WebDriverWait(driver, 30).until( expected\_conditions.presence\_of\_element\_located((By.CSS\_SELECTOR/CLASS\_NAME/ID…,"название элемента"))) позволяет дождаться возникновения элемента, соответствующего способу его локализации и имени объекта.

На семинаре был рассмотрен конкретный пример излечения названий и ссылок на статьи, которые выдаёт сайт mixmag.net при введении поискового запроса “London”.