УДК 004.04 https://doi.org/10.26160/2541-8637-2025-15-243-246

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ СБОРА И АНАЛИЗА ДАННЫХ В НАУЧНЫХ ИССЛЕДОВАНИЯХ

Михайлов А.М., Щербакова И.В.
Воронежский государственный лесотехнический университет имени Г.Ф. Морозова, Воронеж

Ключевые слова: веб-скрапинг, python, selenium, python-docx, автоматизация сбора данных, извлечение данных с веб-сайтов.
Аннотация. В быстро развивающейся сфере научных исследований способность эффективно собирать и анализировать данные приобретает существенное значение. В этой связи возникает потребность в разработке программы (бота), выполняющей рутинные операции по извлечению данных со страниц определенного сайта и формированию из них текстового документа, например, в формате MS Word. Автоматизация процесса сбора данных позволяет исследователю сосредоточить усилия на интеллектуальной работе. В статье рассматривается, методика применения Selenium и Python для автоматизации сбора данных и оптимизации исследовательских процессов.

AUTOMATISATION OF DATA COLLECTION AND ANALYSIS PROCESSES IN SCIENTIFIC RESEARCH

Mikhailov A.M., Shcherbakova I.V.
Voronezh State Forest Engineering University named after G.F. Morozov, Voronezh

Keywords: web scraping, python, selenium, python-docx, data collection automation, data extraction from websites.
Abstract. In the rapidly developing field of scientific research, the ability to efficiently collect and analyze data is becoming essential. In this regard, there is a need to develop a program (bot) that performs routine operations to extract data from the pages of a particular site and form a text document from them, for example, in MS Word format. Automating the data collection process allows the researcher to focus on intellectual work. The article discusses the methodology of using Selenium and Python to automate data collection and optimize research processes.

При первоначальных исследованиях библиотек для автоматизации сбора данных, таких как Scrapy, lxml и Beautiful Soup, было обнаружено, что многие современные веб-сайты имеют динамическую структуру, что значительно усложняет задачу парсинга. Для решения задачи автоматизации процесса сбора и анализа данных была выбрана библиотека Selenium, которая позволяет имитировать действия пользователя и собирать данные динамически [1]. Для примера была разработана программа, собирающая информацию о зарегистрированных патентах: название, изображение и описание патента. Исследование и тестирование программы проводились на динамическом сайте https://yandex.ru/patents/. Следует отметить, что программа может быть легко перестроена для сбора другой информации, не связанной с патентами.

Selenium – это фреймворк с открытым исходным кодом, который используется для автоматизации веб-приложений в целях тестирования. Однако его возможности выходят далеко за рамки тестирования. При разработке программы использовался Python, поскольку он является одним из самых

популярных языков программирования на сегодняшний день, особенно в сфере обработки данных и автоматизации [2]. Для удобства чтения, написания и дальнейшего масштабирования кода были использованы принципы объектно-ориентированного программирования: были созданы несколько классов и их методов, которые отвечают за определенные задачи [3].

На рисунке 1 перечислены модули и библиотеки, которые использовались для написания бота.

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import time
import zipfile
from docx import Document
from docx.shared import Pt
from docx.shared import Inches
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT
import requests
import bs4
import urllib.request
```

Рис. 1. Перечень библиотек, применяемых для создания программы сбора и систематизации информации

Основным классом в разработанной программе является BotWindow, который решает следующие основные задачи:

– Инициализация браузера с опциональной поддержкой прокси и пользовательского агента: позволяет настроить параметры браузера, такие как использование прокси-сервера для обхода географических ограничений или изменение пользовательского агента для имитации различных устройств или браузеров.

– Сбор ссылок: Метод get_driver осуществляет переход по страницам результатов поиска по определенному запросу и собирает ссылки на отдельные страницы.

– Извлечение данных из собранных ссылок: Метод get_data_links открывает каждую ссылку, извлекает заголовок и текст патента, а затем сохраняет эту информацию в документ Word с помощью библиотеки python-docx [4].

Описание класса BotWindow представлено на рисунке 2.

В конструкторе класса инициализируются настройки браузера. Если переданы параметры use_proxy или user_agent, они добавляются в опции браузера. Также создаётся объект документа Word для последующего сохранения данных. Метод get_driver открывает браузер и переходит по страницам результатов поиска. Он собирает ссылки на страницы и сохраняет их в файл links.txt. Описание метода представлено на рисунке 3.

Код программы для извлечения данных из ссылок и сохранения в документ Word представлен на рисунке 4.

В результате выполнения кода мы получим готовый документ MS Word с уже собранной информацией, отформатированной в соответствии с установленными настройками. Настройки задавались через библиотеку docx.

```
class BotWindow() :
    def __init__(self, use_proxy = False, user_agent = False) :
    self.chrome_options = webdriver.ChromeOptions()
    if user_agent :
        self.chrome_options.add_argument(f'user-agent={user_agent}')
        if use_proxy :
            pluginfile = 'proxy_auth_plugin.zip'
            with zipfile.ZipFile(pluginfile, 'w') as zp :
zp.writestr("manifest.json", manifest_json)
zp.writestr("background.js", background_js)
self.chrome_options.add_extension(pluginfile)
self.headers = {
    'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7',
    'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 YaBrowser/24.4.0.0 Safari/537.36'}
    self.driver = webdriver.Chrome(options = self.chrome_options)
    self.doc = Document()
```

Рис. 2. Описание класса BotWindow

```
def get_driver(self) :
    try :
    with self.driver as driver :
self.links = []
for page_number in range(1, 21) :
    url =
    f'https://yandex.ru/patents?dco=RU&dco=SU&dl=ru&dt=0&dty=1&dty=2&s=0&sp
    ={page_number}&spp=10&st=0&text=авиация'
    driver.get(url)
    time.sleep(1)
    elems =
    WebDriverWait(driver, 20).
    until(EC.presence_of_all_elements_located((By.CLASS_NAME, 'snippet-title')))
    for elem in elems :
link = elem.get_attribute('href')
if link is None :
continue
else :
    self.links.append(link)
    with open('links.txt', 'w') as file :
for link in self.links :
    file.write(link + '\n')
    finally :
    driver.quit()
```

Рис. 3. Описание метода get_driver

Автоматизация сбора данных позволяет обеспечить их целостность и значительно сократить время, затрачиваемое на рутинную работу. С помощью Selenium можно запрограммировать проверку данных, полученных с веб-сайтов, на наличие несоответствий или пропущенных значений. Для программирования процесса проверки можно использовать Python. Это позволит исследователям автоматически очищать данные, что обеспечит попадание в их аналитическую систему только необходимой информации.

Selenium позволяет решить основную проблему сбора данных, связанную с динамическим характером сайтов. Selenium эффективно работает с динамическими веб-страницами, обеспечивая доступ к необходимым данным, даже если они загружаются с помощью JavaScript.

```
def with_selenium(self) :
    with open('links.txt', 'r') as file :
links = file.readlines()
with self.driver as driver :
for url in links :
driver.get(url)
try :
    title = WebDriverWait(driver, 10).
    until(EC.presence_of_element_located((By.CLASS_NAME, 'document-title')))
    doc_text = WebDriverWait(driver, 10).
    until(EC.presence_of_element_located((By.CLASS_NAME, 'doc-text')))
    images = WebDriverWait(driver, 10).
    until(EC.presence_of_all_elements_located((By.CLASS_NAME, 'carousel-image')))
    title_paragraph = self.doc.add_paragraph()
    title_run = title_paragraph.add_run(title.text)
    title_run.bold = True
    title_run.font.size = Pt(14)
    title_paragraph.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
    self.doc.add_paragraph(doc_text.text)
    for image in images :
src = image.get_attribute('src')
urllib.request.urlretrieve(src, 'temp.jpg')
self.doc.add_picture('temp.jpg', width = Inches(2.5))
self.doc.save('test.docx')
except :
    print('No such element')
    driver.quit()
```

Рис. 4. Код программы для извлечения данных из ссылок и сохранения в документ Word

Данный подход может быть расширен и адаптирован для различных задач, связанных с автоматизацией сбора и обработки данных, что открывает широкие возможности для анализа и принятия обоснованных решений на основе полученной информации.

Важно отметить, что на веб-сайтах часто действуют условия предоставления услуг, которые ограничивают автоматизированный сбор данных. Нарушение этих условий недопустимо. Исследователи должны обдуманно подходить к решению задачи автоматизации сбора данных, соблюдая этические стандарты и нормативные требования.

Список литературы

1. Крапивин Р.Р., Хамидулин М.Р. Улучшенное логирование ошибок при работе с модулем selenium на языке Python // International Journal of Advanced Studies. – 2023. – Т. 13, №3. – С. 26-35.
2. Muthukadan, B. Selenium with Python [Электронный ресурс]. – Режим доступа: https://selenium-python.readthedocs.io/.
3. Чирков А.Н. Сравнительное исследование библиотеки beautifulsoup4 и selenium для парсинга веб-страниц при помощи python // Научнотехнические инновации и веб-технологии. – 2022. – № 1. – С. 74-78.
4. Canny S. python-docx 1.1.2 documentation [Электронный ресурс]. – Режим доступа: https://python-docx.readthedocs.io/en/latest/index.html.

Сведения об авторах:

Михайлов Андрей Михайлович – студент;

Щербакова Ирина Владимировна – к.т.н., доцент, заведующий кафедрой автоматизации производственных процессов.