

Разработка графического интерфейса для парсинга новостей с сайта ПГУ с использованием Python и библиотеки BeautifulSoup

Малик Александр Игоревич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается процесс разработки приложения с графическим интерфейсом, предназначенного для парсинга новостей с сайта ПГУ с использованием Python и библиотеки BeautifulSoup. Приложение позволяет извлекать актуальные новости с веб-страницы, отображать их в графическом интерфейсе, а также обновлять список новостей по запросу пользователя. Описание работы включает в себя как технические аспекты создания приложения, так и особенности использования графического интерфейса с библиотекой Tkinter. Также рассматривается реализация функциональности загрузки изображений и перехода к полным статьям. Разработанное приложение может быть использовано для создания информационных панелей новостей и в качестве образовательного примера для студентов и начинающих разработчиков.

Ключевые слова: Python, BeautifulSoup, Tkinter, парсинг новостей, веб-разработка, графический интерфейс, библиотека, программирование, информационные технологии.

Development of a graphical interface for parsing news from the PSU website using Python and the BeautifulSoup library

Malik Aleksandr Igorevich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article discusses the process of developing an application with a graphical interface designed for parsing news from the PSU website using Python and the BeautifulSoup library. The application allows you to extract relevant news from a web page, display it in a graphical interface, and update the news list at the user's request. The job description includes both the technical aspects of creating the application and the specifics of using the graphical interface with the Tkinter library. The implementation of the functionality of uploading images and navigating to full articles is also being considered. The developed application can be used to create news dashboards and as an educational example for students and novice developers.

Keywords: Python, BeautifulSoup, Tkinter, news parsing, web development, GUI, library, programming, information technology.

1 Введение

1.1 Актуальность

В современном мире информация быстро устаревает, и для пользователя важно иметь возможность своевременно получать актуальные новости. Автоматизация процесса сбора новостей с популярных сайтов с использованием программных средств позволяет значительно упростить задачу получения свежей информации. Использование парсинга с помощью Python и библиотеки BeautifulSoup для автоматического извлечения данных из веб-страниц является эффективным и доступным решением. Графический интерфейс, реализованный с помощью Tkinter, улучшает взаимодействие пользователя с приложением и делает его удобным для широкой аудитории.

1.2 Обзор исследований

Обзор исследований по использованию библиотеки BeautifulSoup и методов парсинга веб-страниц с помощью Python показывает разнообразие подходов к решению задач автоматизации сбора данных. В исследовании, проведённом А.Н. Чирковым, рассматривается сравнительный анализ двух популярных библиотек для парсинга — BeautifulSoup4 и Selenium. Исследование было выполнено на примере торговой площадки скинов для игры Counter-Strike. Результаты показывают, что BeautifulSoup4 идеально подходит для работы с простыми статичными страницами, тогда как Selenium проявляет наибольшую эффективность при обработке динамических страниц, загружаемых с помощью JavaScript. Делается вывод, что выбор инструмента зависит от сложности и динамичности веб-страницы, с которой предстоит работать [1].

В работе Г.А. Болтовского описывается создание парсера для сбора текстов песен с сайта Genius с использованием библиотеки BeautifulSoup. Процесс разработки парсера включает анализ структуры страницы, настройку запросов и обработку исключений. Особое внимание уделяется извлечению текстов песен и их сохранению в удобном для пользователя виде. Результатом исследования является создание рабочего инструмента для автоматического сбора данных с сайта, что демонстрирует высокую эффективность использования BeautifulSoup при парсинге текстового контента [2].

Исследование Э.Э. Мальцевой и В.Ю. Кондратьева посвящено возможностям использования Python для парсинга веб-сайтов с фокусом на автоматизацию сбора данных в области интернет-маркетинга. Рассматриваются различные способы парсинга для анализа конкурентов, мониторинга цен и сбора статистики с веб-ресурсов. Особое внимание уделено библиотекам BeautifulSoup и Selenium, а также методам обработки и представления полученных данных. Подчёркивается важность этих технологий для развития маркетинговых стратегий и повышения эффективности бизнеса [3].

В статье Д.А. Турышева рассматривается процесс парсинга данных с сайта Авито в категории недвижимости с помощью Python. Описывается, как с помощью автоматизации сбора информации можно создавать сводные

таблицы и отчеты. Подробно рассматриваются способы обработки полученных данных в Excel, что позволяет эффективно анализировать информацию о ценах на недвижимость. Работа иллюстрирует возможности Python и парсинга для решения задач анализа рынка недвижимости [4].

Исследование В.А. Кононова направлено на описание процесса создания парсера для сбора информации с различных сайтов с использованием Python. Рассматриваются этапы настройки парсера, а также способы оптимизации работы с большими объемами данных. Особое внимание уделяется удобству и функциональности работы с HTML-форматом. Исследование помогает понять, как с помощью Python можно эффективно решать задачи извлечения данных с веб-страниц и их представления пользователю [5].

1.3 Цель исследования

Целью данной работы является разработка приложения для парсинга новостей с сайта ПГУ с использованием языка программирования Python и библиотеки BeautifulSoup, а также создание графического интерфейса с применением Tkinter для отображения полученной информации.

2 Материалы и методы

Для разработки приложения использовались следующие инструменты и технологии:

- Язык программирования **Python** — основной инструмент для написания логики приложения.
- Библиотека **BeautifulSoup** — используется для парсинга HTML-кода веб-страниц и извлечения информации о новостях.
- Библиотека **Tkinter** — применена для создания графического интерфейса пользователя (GUI).
- Библиотека **requests** — используется для отправки HTTP-запросов и получения данных с сайта.
- Библиотека **Pillow** — применяется для обработки изображений, скачанных с веб-страницы.

Метод парсинга заключается в загрузке HTML-кода страницы с помощью библиотеки requests, после чего данные о новостях извлекаются с помощью BeautifulSoup. Информация о заголовках, датах публикации, ссылках на статьи и изображениях отображается в окне приложения с использованием Tkinter.

3 Результаты и обсуждения

1. Импорт библиотек (рис.1).

```
import requests
from bs4 import BeautifulSoup
import tkinter as tk
from tkinter import messagebox
import webbrowser
from PIL import Image, ImageTk
from io import BytesIO
```

Рисунок 1. Импорт библиотек

Здесь импортируются необходимые библиотеки:

- requests для HTTP-запросов, чтобы получить содержимое страницы.
- BeautifulSoup для парсинга HTML.
- tkinter и messagebox для создания графического интерфейса и отображения сообщений.
- webbrowser для открытия полной статьи в браузере.
- PIL и ImageTk для работы с изображениями, загружаемыми с сайта.
- BytesIO для работы с изображениями в памяти.

2. Функция для получения новостей с сайта (рис.2).

```
def get_news(): 2 usages
    url = "https://pqusa.ru/news/"

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }

    # Отправляем запрос на страницу
    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print(f"Ошибка загрузки страницы: {response.status_code}")
        return []

    # Разбираем HTML с помощью BeautifulSoup
    soup = BeautifulSoup(response.text, features='html.parser')

    # Находим все карточки с новостями
    news_items = soup.find_all(name='article', class_='event-card_secondary')
    news_data = []

    # Извлекаем данные для каждой новости
    for item in news_items:
        title = item.find('h4', class_='event-card_title')
        date = item.find('span', class_='event-card_date')
        link_tag = item.find('a', class_='event-card_link')
        link = link_tag['href'] if link_tag else None # Ссылка на полную новость

        # Изображение новости
        image_tag = item.find('img')
        image_url = image_tag['data-src'] if image_tag else None # Получаем URL изображения

        # Добавляем базовый URL для изображения, если оно относительное
        if image_url and not image_url.startswith('http'):
            image_url = f"https://pqusa.ru{image_url}"

        title_text = title.text.strip() if title else 'Без заголовка'
        date_text = date.text.strip() if date else 'Без даты'
        link_url = f"https://pqusa.ru{link}" if link else None # Полный URL

        news_data.append({
            'title': title_text,
            'date': date_text,
            'link': link_url,
            'image_url': image_url
        })

    return news_data
```

Рисунок 2. Код

- URL: Отправляем запрос на страницу новостей.
- HTTP-запрос: С помощью requests.get отправляется запрос с необходимыми заголовками.
- Парсинг HTML: С помощью BeautifulSoup извлекаем карточки новостей.
- Извлечение данных: Для каждой новости извлекаем заголовок, дату, ссылку на статью и изображение.
- Возврат данных: Возвращаем список новостей в формате словаря.

3. Функция для загрузки изображений (рис.3).

```
def load_image(image_url): 1 usage
    try:
        img_response = requests.get(image_url)
        img_data = img_response.content
        img = Image.open(BytesIO(img_data))
        img = img.resize((150, 150)) # Размер изображения
        img_tk = ImageTk.PhotoImage(img)

        # Сохраняем изображение в глобальной переменной
        loaded_images.append(img_tk)
        return img_tk
    except Exception as e:
        print(f"Ошибка загрузки изображения: {e}")
        return None
```

Рисунок 3. Код

- Загрузка изображения: С помощью `requests.get` загружаем изображение по URL.
- Преобразование: Загружаем изображение в объект PIL, изменяем его размер и преобразуем в формат, подходящий для Tkinter.
- Глобальная переменная: Изображение сохраняется в глобальном списке `loaded_images`, чтобы избежать его исчезновения при обновлении интерфейса.

4. Функция для отображения полной новости (рис.4).

```
def show_full_news(news_url): 1 usage
    if not news_url:
        messagebox.showerror(title="Ошибка", message="Полная новость недоступна.")
        return
    # Открываем новость в браузере
    webbrowser.open(news_url)
```

Рисунок 4. Код

- Открытие ссылки: При нажатии на новость, открывается ссылка в браузере с помощью `webbrowser.open`.

5. Функция для отображения новостей в интерфейсе (рис.5).

```
def display_news(news_data, news_frame): 2 usages
# Очищаем область с новостями перед отображением новых данных
for widget in news_frame.winfo_children():
    widget.destroy()

# Список новостей с изображениями
for i, news in enumerate(news_data):
    # Строка новостей
    news_frame_row = tk.Frame(news_frame)
    news_frame_row.grid(row=i, column=0, sticky='w', padx=10, pady=5)

    # Изображение
    if news['image_url']:
        img = load_image(news['image_url'])
        if img:
            img_label = tk.Label(news_frame_row, image=img)
            img_label.image = img # Сохраняем ссылку на изображение в атрибуте метки
            img_label.grid(row=0, column=0, rowspan=2)

    # Заголовок новости
    title_label = tk.Label(news_frame_row, text=news['title'], font=('Arial', 12, 'bold'), anchor="w", width=50)
    title_label.grid(row=0, column=1, sticky="w")

    # Дата новости
    date_label = tk.Label(news_frame_row, text=news['date'], font=('Arial', 10), anchor="w")
    date_label.grid(row=1, column=1, sticky="w")

    # Обработчик клика по новости
    def on_news_click(news_url=news['link']):
        show_full_news(news_url)

    news_frame_row.bind("<Button-1>", lambda event, url=news['link']: on_news_click(url))
```

Рисунок 5. Код

- Очистка интерфейса: Прежде чем отображать новые новости, очищаем предыдущие виджеты.
- Отображение новости: Для каждой новости создается отдельный контейнер (фрейм) с изображением, заголовком, датой и кнопкой для перехода по ссылке.
- Обработчик клика: При клике на новость, она открывается в браузере.

6. Функция для обновления новостей (рис.6).

```
def refresh_news(news_frame): 1 usage
news_data = get_news() # Получаем новости
if news_data:
    display_news(news_data, news_frame) # Отображаем новости
else:
    messagebox.showerror(title="Ошибка", message="Не удалось загрузить новости.")
```

Рисунок 6. Код

- Обновление данных: Получаем новые данные и отображаем их в интерфейсе.

7. Функция для отображения информационного окна "О приложении" (рис.7).

```
def show_about(): 1 usage
about_message = "Приложение для отображения новостей с сайта ПГУ.\n\nВерсия 1.0\n\nАвтор: Малик Александр"
messagebox.showinfo(title="О приложении", about_message)
```

Рисунок 7. Код

- Окно с информацией: Появляется окно с информацией о приложении.

8. Основная функция (рис.8).

```
def main(): 1 usage
    window = tk.Tk()
    window.title("Новости ПГУ")

    # Отображаем начальные новости
    news_frame = tk.Frame(window)
    news_frame.pack(pady=10)

    news_data = get_news()
    if news_data:
        display_news(news_data, news_frame)
    else:
        messagebox.showerror(title="Ошибка", message="Не удалось загрузить новости.")

    # Кнопка "Обновить"
    refresh_button = tk.Button(window, text="Обновить", command=lambda: refresh_news(news_frame))
    refresh_button.pack(pady=5)

    # Кнопка "О приложении"
    about_button = tk.Button(window, text="О приложении", command=show_about)
    about_button.pack(pady=5)

    window.mainloop()
```

Рисунок 8. Код

- Основное окно: Создаем окно с помощью Tkinter.
- Кнопки: Добавляем кнопки для обновления новостей и открытия информации о приложении.
- Запуск: Запускаем главный цикл интерфейса с помощью window.mainloop().

9. Запуск приложения (рис.9).

```
if __name__ == "__main__":
    main()
```

Рисунок 9. Код

- Запуск программы: Если файл запускается как основной, вызывается функция main, которая запускает приложение.

10. Внешний вид приложения (рис.10).

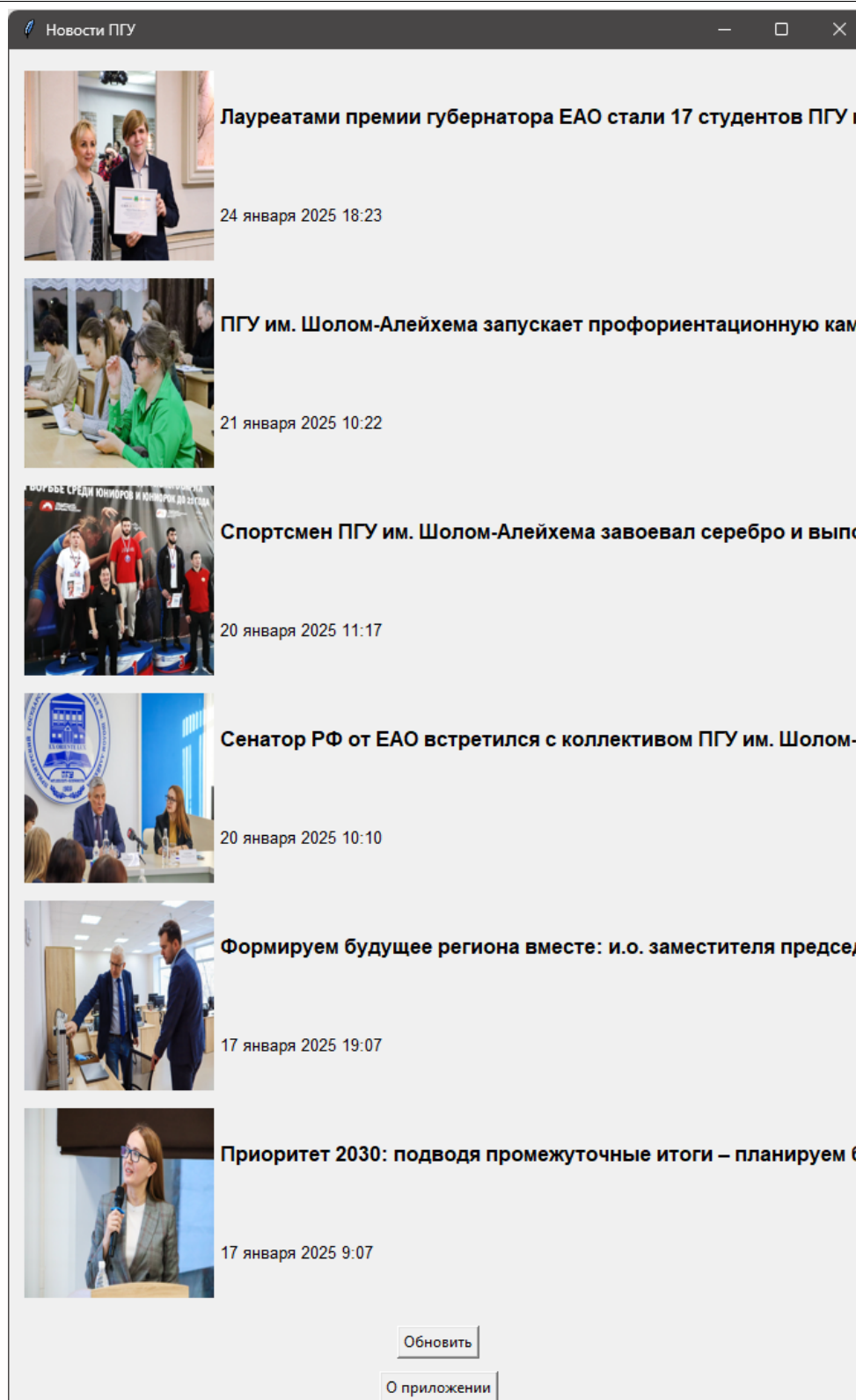


Рисунок 10. Внешний вид приложения

4 Выводы

В ходе выполнения работы был разработан парсер для автоматического сбора новостей с сайта ПГУ с использованием языка программирования Python и библиотеки BeautifulSoup. Разработанное приложение эффективно извлекает данные, такие как заголовки новостей, даты публикации, ссылки на полные статьи и изображения. Также реализована возможность обновления

новостей и их отображения в графическом интерфейсе с использованием библиотеки Tkinter.

Анализ возможностей использования Python для парсинга веб-страниц показал, что BeautifulSoup является мощным инструментом для обработки статичных веб-страниц. В ходе работы был также реализован функционал для работы с изображениями, что позволяет наглядно отображать новости.

Программное решение продемонстрировало свою эффективность в задаче автоматизированного сбора данных с веб-страниц и их отображения в удобном для пользователя формате. Важно отметить, что использование таких библиотек как requests, BeautifulSoup и Tkinter позволяет значительно ускорить процесс разработки и улучшить качество конечного продукта.

В рамках будущих исследований можно рассмотреть расширение функционала, включающее улучшение обработки динамических веб-страниц, оптимизацию работы с большими объемами данных и добавление дополнительных функций, таких как фильтрация и поиск новостей.

Библиографический список

1. Чирков А.Н. Сравнительное исследование библиотеки beautifulsoup4 и selenium для парсинга веб-страниц при помощи python // Научно-технические инновации и веб-технологии. 2022. №. 1. С. 74-78.
2. Болтовский Г.А. Создание парсера на языке Python с использованием библиотеки BeautifulSoup // Постулат. 2022. № 6.
3. Мальцева Э.Э., Кондратьев В.Ю. Возможности парсинга веб-сайтов на языке программирования python // Цифровизация экономики: направления, методы, инструменты. 2023. С. 444-446.
4. Турышев Д.А. Парсинг и обработка данных с помощью языка программирования python и табличного процессора Excel // Молодёжь XXI века: шаг в будущее. 2021. С. 788-789.
5. Кононов В.А. Парсинг сайтов. парсер на языке python// Интернаука. 2020. № 29 (158). С. 9-10.