УДК 004

Создание скрипта для синтаксического анализа HTML

Радионов Сергей Владимирович
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
Целью данного исследования является создание программы синтаксического анализа гипертекстового языка HTML. В статье показан процесс разработки программы на языке программирования python. Результатом исследования является продукт, который может преобразовать HTML текст в набор классов, имеющих соответствующие параметры.
Ключевые слова: python, html, синтаксический анализ, программа.

Create script for HTML Parsing

Radionov Sergey Vladimirovich
Sholem-Aleichem Priamursky State University
Student

Abstract
The purpose of this study is to create a program for parsing the hypertext language HTML. The article shows the process of developing a program in the python programming language. The result of the study is a product that can convert HTML text to a set of classes that have the appropriate parameters.
Keywords: python, html, parsing, program.

В мире широко распространен интернет. Основным средством просмотра информации в интернете являются web страницы. Эти страницы написаны на гипертекстовом языке HTML. Чтобы получить данные из HTML текста, такие как цена продукта в интернет-магазине или данные о погоде метеорологического сервиса, необходимо провести синтаксический анализ (парсинг) кода страницы.

В статье И.Кутепова рассматривается пример построения вычислителя на основе нечеткой логики и применение языка Python для написания интегрированной среды разработки [2]. И.Е.Бронштейн в своей статье описал вывод типов для программного кода на языке Python. Сначала производится обзор описанных в научной литературе алгоритмов вывода типов для языков с параметрическим полиморфизмом. Затем даётся описание нового алгоритма, являющегося модификацией одного из предыдущих: алгоритма декартова произведения. Показывается, как модуль вывода типов, использующий новый алгоритм, анализирует различные конструкции языка Python. Представляются результаты работы над прототипом [3]. В работе

В.В.Найденова протестирована производительность пары аналогичных приложений, реализующих CRUD логику с помощью прослойки ORM. Сравнивается SQLAlchemy – де-факто стандартный ORM для Python с динамическим ORM для C++ собственной разработки – YB.ORM. Сравнивается производительность при использовании CPython и PyPy. Проверяется влияние отключения логов на производительность [4]. В статье И.Е.Бронштейн рассматриваются виды дефектов, которые обычно встречаются в программном коде на языке Python. Показывается, что возможные дефекты для Python не похожи на те, что часто встречаются в коде на Си/Си++ и, следовательно, необходимо исследование дефектов в крупных проектах с открытым исходным кодом. Даётся классификация найденных дефектов на основе того, нужен ли для нахождения ошибки вывод типов. Показывается, что существует небольшая доля "простых" дефектов, но для обнаружения большинства дефектов вывод типов необходим. Рассматривается вопрос, какие конструкции языка Python должны поддерживаться при выводе типов для нахождения реальных дефектов [5]. В работе Д.А.Кузнецов рассматривается структура интерфейса программы «Фармацевтическая экономическая безопасность» для фармацевтических организаций. Описывается функциональное предназначение и возможности пунктов основного меню прикладной программы [6]. Ю.А.Котов, А.В.Шаповалов в своей статье рассмотрели интерфейс программной реализации экспертной системы для восстановления простой замены букв текста. Описаны базовые элементы интерфейса, включающие выбор функциональных методов и базовых операций (замена, сдвиг, перебор). Не менее значимы иностранные исследования в данной сфере [8-9].

Разработку программы следует начать с загрузки библиотеки requests. Эта библиотека понадобится для получения html кода по веб-запросу. Установить ее можно командой «pip install requests». Создаем скрипт «html_parser.py». В нем подключаем все нужные библиотеки (Рис.1).

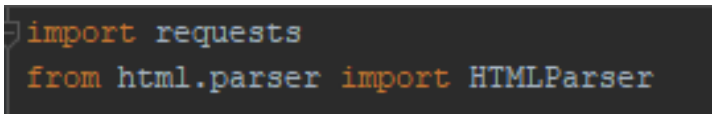

Рис. 1. Подключаем библиотеки

До синтаксического анализа нужно подготовить класс, в котором будет храниться информация о тэге (div, p, img и т.д.). Создадим класс и опишем его конструктор. На вход конструктора подаем название тэга, атрибуты тэга и его родителя. В этой функции сохраняем данные, а также создаем пустые списки для потомков и комментариев этого тэга (Рис.2).

```
class HtmlTag:

    def __init__(self, tag, attrs, parent=None):
        self.tag = tag
        self.attrs = attrs
        self.parent = parent
        self.childrens = []
        self.comments = []
        self.text = ''
```

Рис.2. Конструктор класса HtmlTag

Добавим в этот класс две функции для удобства(Рис.3). Первая будет добавлять потомка тэгу. Вторая будет добавлять комментарий.

```
    def add_child(self, c):
        self.childrens.append(c)

    def add_comment(self, c):
        self.comments.append(c)
```

Рис.3. Дополнительные функции

Создадим класс HtmlParser, который будет отвечать за вывод результата синтаксического анализа. В этом классе создаем вложенный класс CustomHTMLParser, наследуемый от HTMLParser, который производит непосредственно синтаксический анализ HTML кода (Рис.4). CustomHTMLParser будет содержать обработчики событий, а именно: начало тэга, конец тэга, данные, комментарий. В конструкторе создаем основной тэгс названием root, от которого будут наследоваться все остальные. Переменная cur_tag нужна для хранения текущего тэга.

```
class HtmlParser:
    class CustomHTMLParser(HTMLParser):

        def __init__(self):
            super().__init__()
            self.root = HtmlTag('root', {})
            self.cur_tag = self.root
```

Рис.4. Создаем классы

Далее необходимо определить какие действия будут при обнаружении нового тэга. Для этого напишем функцию handle_starttag, в которую подается название тэга и его атрибуты (Рис.5). В обработчике меняем формат атрибутов на более удобный. Создаем новый объект тэга, которому родителем приписываем предыдущий тэг. Добавляем новый тэг предыдущему в список наследников. Сохраняем новый тэг вместо предыдущего.

```
def handle_starttag(self, tag, attrs):
    attrs = dict(attrs)
    if 'class' in attrs:
        attrs['class'] = attrs['class'].split(' ')

    t = HtmlTag(tag, attrs, parent=self.cur_tag)
    self.cur_tag.add_child(t)
    self.cur_tag = t
```

Рис.5. Обработчик начала тэга

Для обработки конца тэга создадим функцию handle_endtag. В этой функции перебираем иерархию тэгов до первого совпадающего по названию, запоминая все перебранные тэги в список. Каждому элементу списка очищаем наследников и присваиваем нового родителя. Родителю добавляем в список новых детей. И закрываем тэг, меняя запомненный тэг на более старший.

```
def handle_endtag(self, tag):
    tags = []
    while self.cur_tag.tag != tag:
        tags.append(self.cur_tag)
        self.cur_tag = self.cur_tag.parent

    for t in tags:
        t.parent = self.cur_tag
        t.childrens = []

    self.cur_tag.childrens += tags
    self.cur_tag = self.cur_tag.parent
```

Рис.6. Обработчик конца тэга

Добавим обработчики для текста и комментариев внутри тэга (Рис.7). В обработчике текста прибавляем полученный текст к имеющемуся тексту тэга. В обработчике комментариев добавляем входящее значение в список комментариев.

```
def handle_data(self, data):
    self.cur_tag.text += data

def handle_comment(self, data):
    self.cur_tag.add_comment(data)
```

Рис.7. Обработчики текста и комментариев

Осталось переопределить функцию feed. В ней добавим возвращение корневого тэга (Рис.8).

```
def feed(self, data):
    super().feed(data)
    return self.root
```

Рис.8. Функция feed

Теперь класс CustomHTMLParser при вызове функции feed вернет корневой тэг (объект класса HtmlTag). Нужно добавить конструктор для класса HtmlParser (Рис.9). В этом конструкторе на вход получаем url или html строку (html_s). Если получили url, то с помощью библиотеки requests делаем запрос и получаем код страницы. Далее запускаем синтаксический анализ, результат которого записываем в переменную root. Теперь из корневого тэга можно получить доступ ко всем тэгам и их параметрам, а также текстовому содержимому.

```
def __init__(self, url=None, html_s=None):
    if url:
        html_s = requests.get(url).content.decode('utf-8')
    self.content = html_s
    p = self.CustomHTMLParser()
    self.root = p.feed(self.content)
```

Рис.9. Конструктор класса HtmlParser

Таким образом класс HtmlParser хранит дерево тэгов, корнем которого является объект root. Пример использования скрипта показан на рисунке 10. Обратиться к дочерним объектам можно через переменную childrens, к родителю через переменную parent. Таким образом можно из любого объекта дерева получить доступ к другому объекту.

```
from html_parser import HtmlParser

p = HtmlParser(url='https://easypassword.ru')
print(p.root)

with open('index.html', 'r', encoding='utf-8') as f:
    s = str(f.read())
    p = HtmlParser(html_s=s)
    print(p.root)
```

Рис.10. Пример использования скрипта

В итоге был написан скрипт для синтаксического анализа гипертекстового языка HTML. Данный скрипт легок в использовании, достаточно всего нескольких строк кода чтобы получить данные синтаксического анализа.

Библиографический список

1. Лутц М. Изучаем python. М., 2009.
2. Кутепов И. Применение языка python при проектировании нечеткого контроллера // Компоненты и технологии. 2013. № 8 (145). С. 148-154.
3. Бронштейн И.Е. Вывод типов для языка python // Труды Института системного программирования РАН. 2013. Т. 24. С. 161-190.
4. Найденов В.В. Тестирование производительности orm в языках python и c++ // RSDN Magazine. 2014. № 1. С. 05-08.
5. Бронштейн И.Е. Исследование дефектов в коде программ на языке python // Программирование. 2013. Т. 39. № 6. С. 25-32.
6. Кузнецов Д.А. Интерфейс программы "фармацевтическая экономическая безопасность" // Российский медико-биологический вестник им. академика И.П. Павлова. 2009. № 2. С. 162-165.
7. Котов Ю.А., Шаповалов А.В. Интерфейс программы восстановления простой замены букв текста // Современные тенденции развития науки и технологий. 2016. № 4-4. С. 57-59.
8. Smith A. W. et al. Application program interface that enables communication for a network software platform : пат. 7117504 США. – 2006.
9. Parikh V., Moore R., Cheng H. Application program interface for a graphics system : пат. 6456290 США. – 2002.