МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

 **«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**КУРСОВАЯ РАБОТА**

**АНАЛИЗ СЕТЕВЫХ ДАННЫХ**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.Е.Лесь

 (подпись)

Направление 01.03.02 «Прикладная математика и информатика»

Направленность (профиль) «Системное программирование и компьютерные технологии» (Математическое и программное обеспечение вычислительных машин)

Научный руководитель

канд. физ.-мат.наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.П.Лукащик

 (подпись)

Нормоконтролер

ст.преп.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В.Харченко

(подпись)

Краснодар

2018

**РЕФЕРАТ**

Курсовая работа 24ст., 12 рис., 5 источников.

АНАЛИЗ СЕТЕВЫХ ДАННЫХ

Объектом изучения является программа-парсер, основанная на анализе DOM дерева страницы-источника информации.

Цель работы: изучить задачу анализа HTML-страниц в интернете, рассмотреть принцип парсинга страниц на основе метода анализа DOM дерева, рассмотреть пример реализации парсинг-программы для интернет-магазинов.

**СОДЕРЖАНИЕ**

Введение……………………………………………………………………..……4

1 Введение в область данной проблемы………………………………………...6

2 Структура программы-парсера ……………………………..…………………7

2.1 Структура веб-документа ………...………………………...…...…...11

2.2 Атрибуты. Теги. Селекторы ………………………………………....12

2.3 Инструменты PHPQuery ………………………..………..…...……...14

3 Реализация…...………………………………………………...….…………....16

4 Недостатки парсеров………………….…………………………………..…...19

Заключение……………………………………………………………………….20

Список использованных источников…………………………………………...21

**ВВЕДЕНИЕ**

Часто приходится работать с большим количеством информации, искать по ключевым словам в безмерном пространстве интернета, который с каждым днём только увеличивает свой объём, что делает поиск вручную очень длительным и ресурсозатратным ‒ необходимо прибегать к целой команде специалистов для поиска и сортировки нужной информации. В подобных ситуациях разумнее всего будет обраться к средствам синтаксического анализа текста, иначе говоря, парсерам. Сперва разберёмся, что это и в каких целях используется.

Представим ситуацию, когда нам необходимо проанализировать некоторый текст и преобразовать его в структурированный набор данных. Это может быть задача поиска формул и учёных, которые их вывели, или же топ-лист клиентов, которые приобрели товары на определенную стоимость в базе покупок. Эту задачу можно решить, полностью просмотрев текст. Но если количество данных, которые необходимо проанализировать, велико, то для решения этой задачи данным способом может уйти и не один день. Второй способ предлагает возложить анализ текста на плечи особой программы, которая будет «просматривать» текст и, встречая определённые наборы символов, заносить данные в новый документ. Такие программы построены на синтаксическом анализе данных, этот термин означает ‒ преобразование исходных данных в структурированный текст.

В данной работе я буду рассматривать анализ HTML-страниц. Это полезный инструмент в Web-разработке и статистике.

Каждая страница в Интернете имеет своё дерево документа DOM (Document Object Model ‒ «объектная модель документа»), который определяет расположение объектов страницы в браузере. Иначе говоря, DOM – это представление документа в виде дерева объектов. Именно DOM будет служить для нас исходными данными. Таким образом, мы можем работать с информацией любой страницы в интернете. Анализ DOM документов и их последующая обработка нашли применение во многих областях:

* cстатистика;
* поддержка актуальности информации;
* объединение потоков информации, для содержания их в одном месте;
* проверка на уникальность информации.

Существует несколько подходов к извлечению информации с сетевых ресурсов:

* анализ DOM дерева;
* парсинг строк;
* использование регулярных выражений;
* XML-парсинг;
* визуальный подход.

Подробно рассмотрим метод парсинга, основанный на анализе DOM дерева. Данный способ извлечения информации с интернет-ресурсов предполагает использование атрибутов элементов дерева для обращения к ним или путь к элементу в DOM дереве. Подробнее о DOM будет говориться в главе 2 пункте 1 «Структура веб-документа».

Из преимуществ данного метода можно отметить возможность получения данных любой сложности, а из недостатков ‒ DOM дерево может быть достаточно сложным (например, в случае, если HTML-код страницы формируется с помощью JavaScript), а также HTML-код интернет-страницы, которую мы используем для извлечения информации и на которой построено DOM дерево, может измениться. В этом случае программа-парсер, реализованная для ранее существующего DOM дерева страницы, перестанет работать корректно, поэтому такой способ парсинга используется для краткосрочных задач.

**1 Введение в область данной проблемы**

Список задач, разумеется, можно расширить или изменить для некоторых проектов. С помощью этой программы мы сможем узнать наименьшие стоимости на определённые товары. Этой идеей уже пользуются популярные сайты, сообщающие о том, где можно купить определённый товар дешевле или в каких магазинах сейчас проводятся акции или делаются скидки: «Яндекс. Маркет», «Триваго», «Биглион».

Программа-парсер на языке PHP с интеграцией разметки HTML для более удобного вывода информации будет опираться на исходный HTML-код страниц интернет-магазинов, которые будут рассмотрены в качестве примера. Необходимую информацию будем получать с помощью представления HTML-кода ‒ модели дерева документа, в котором удобнее ориентироваться в исходном HTML-коде. (Подробнее устройство DOM будет рассмотрено в главе 2 пункте 1 «Структура веб-документа») Для написания программы-парсер будем использовать библиотеку PHPQuery. Данная библиотека позволяет извлекать из HTML-кода информацию, работая с его логическими элементами и не прибегая к использованию сложных регулярных выражений. Используемые методы библиотеки будут рассмотрены в главе 2 пункте 3 «Инструменты PHPQuery».

**2 Принцип работы программы-парсера**

Уже реализованные программы-парсеры, основанные на анализе DOM, подчиняются одному и тому же принципу обработки данных и выполняет работу в несколько этапов:

Шаг 1*.*Выбор сайта-донора и анализ его исходного кода.

Сайт-донор ‒ это страница в Интернете, с которой мы будем осуществлять сбор информации. Она выбирается в зависимости от поставленной задачи. Ею может быть новостной сайт, интернет-магазин, банк и другие сайты из Интернета. Затем необходимо получить доступ к коду страницы, как это показано на рисунке 1.



Рисунок 1 ‒ Код страницы

Мы увидим HTML-страницу, которая формирует DOM выбранной нами страницы. Мы можем осуществлять поиск элементов напрямую по HTML-странице, или использовать его DOM-представление посредством инструментов разработчика (пункт «посмотреть код»): так можно видеть, что каждый фрагмент кода страницы соответствует некоторому элементу на странице в Интернете. Таким образом, мы можем установить связь между нужными нам элементами на странице-доноре и соответствующими им элементами в исходном HTML-коде. Например, на рисунке 2 показан заголовок со страницы интернет-магазина техники, а на русенке 3 изображён фрагмент HTML-кода, связанного с этим заголовком:



Рисунок 2 ‒ Заголовок



Рисунок 3 ‒ Фрагмент HTML

Селектор‒ это последовательность нескольких простых селекторов, которые разделяются символами «>» и «+», или им может быть также один простой селектор. Простой селектор, в свою очередь, является тегом либо атрибутом. Селекторы используются для связи элементов на странице и его DOM дерева. В селекторе символ «>» обозначает дочерний элемент, «+» обозначает «братские» отношения между элементами. В селекторе, содержащем несколько простых селекторов, расположение простых селекторов обозначает порядок вложения элементов, а сам селектор указывает на крайний справа элемент.

Например, селектор для найденного нами заголовка выглядит так, как это показано на рисунке 4:



Рисунок 4. ‒ Селектор заголовка

Шаг 2*.* Извлечение значений из кода веб-страницы, отделяя при этом требуемый материал от программного кода страницы.

На данном шаге программа-парсер, выполняющая сбор информации со страницы-донора, должна будет дополнена нашей информацией о расположении интересующих нас элементов, т. е. найденными нами селекторами, указывающими на расположение элемента в исходном коде страницы.

Этот путь можно получить, взглянув на HTML-страницу дерева документа рассматриваемого сайта, или автоматически скопировав его, воспользовавшись функцией «Скопировать селектор» в инструментах разработчика. При запуске программы-парсера, она начнёт анализ всего HTML-документа, сравнивая каждую синтаксическую лексему с селекторами, и сохранит фрагменты исходного кода, подходящие для данного селектора.

Шаг 3*.* Сохранение/вывод данных.

В зависимости от поставленных целей полученные данные можно сохранить в папке в текстовом файле или же внести данные на определённую интерент-страницу, таким образом информация на ней будет изменятся одновременно с изменениями на странице сайта-донора. Различные уже реализованные программы-парсеры могут сохранять полученную в результате поиска по заданным «ключевым словам» информацию в базу данных, подготовить её для печати для последующей работы с полученными данными или сохранить в формате отчёта. Рассмотрим более подробно случай, когда информацию необходимо отображать на некоторой интернет-странице.

Для этого можно воспользоваться функцией *echo* в PHP, но для большинства нужд этот вывод слишком простой. Для форматированного вывода можно воспользоваться вставками фрагментов кода PHP в HTML-код страницы, на которую необходимо отобразить результат работы программы-парсера. Реализация данного вывода показана на рисунке 5:



Рисунок 5 ‒ Вывод данных

Из преимущества программы-парсера данных можно отметить:

* высокую скорость обработки,
* анализ огромных объемов данных,
* автоматизацию процесса отбора.

Однако, если использовать программу-парсер в целях наполнения содержанием интернет-страницу (например для новостного сайта), то отсутствие уникального контента отрицательно отразится на рейтинге данной страницы в поисковых системах.

Программа-парсер может быть использована для широкого круга задач:

* извлечение контактных данных для создания базы клиентов;
* автоматический анализ по синтаксическим лексемам большого объёма текста;
* анализ популярных социальных сетей для составления статистики (популярные предпочтения, запросы, отклики и т. д.);
* сбор популярных ссылок для последующего составления рейтинга популярности сайтов;
* поиск нужной информации в базе данных (наличие определённого элемента в таблице);
* сбор ID популярных участников групп (например, организации базы постоянных клиентов).

В общем случае программа-парсер призвана упросить анализ информации, а также имеет возможность обработки обновляющейся информации непрерывно в реальном времени, что заменяет регулярный мониторинг вручную.

**2. 1 Структура веб-документа**

Объектная модель документа (или модель DOM) – представляет собой структуру веб-документа и определяет её организацию. Можно сказать, что DOM ‒ это модель HTML-кода, которую мы видим, заходя на тот или иной сайт. Таким образом, если на рисунке 6 показан простейший HTML-код страницы:



Рисунок 6 ‒ HTML-код

То на рисунке 7 изображена его модель дерева документа:



Рисунок 7 ‒ DOM

В этом дереве выделено два типа узлов:

* теги образуют узлы-элементы. Естественным образом одни узлы вложены в другие. Структура дерева образована исключительно за счет них;
* текст внутри элементов образует текстовые узлы, обозначенные как *#text*. Текстовый узел содержит исключительно строку текста и не может иметь потомков, то есть он всегда на самом нижнем уровне.

Узлы обозначают HTML-элементы – это основные составляющие пространства имён в HTML. HTML описывает множество элементов, каждый из которых принадлежит к определённой семантической группе. Они определяют структуру документа и его вид в Интернете.

**2. 2 Атрибуты. Теги. Селекторы**

Теги – элемент разметки HTML, который может также содержать атрибуты, и, как правило, теги открываются и закрываются, а внутри у тегов расположен некоторый контент. Пример тега можно увидеть на рисунке 8.



Рисунок 8 ‒ Тег

Различные теги указывают на различные элементы на HTML-странице: текст, заголовки, картинки, ссылки и прочее. Чаще всего в роли атрибутов у тегов выступают id и class. Различия между ними в том, что первый обычно используется для уникального элемента на странице, а второй – для схожих элементов. Однако любой атрибут используется для того, чтобы придать элементу определённый вид на странице (изменить размер, цвет, расположение и прочее). Наша программа-парсер будет строиться именно на использовании атрибутов – по ним мы будем идентифицировать нужную нам информацию на сайте-доноре. Но сначала поговорим о вложенности DOM, которую мы также будем использовать в своих целях.

Вложенные элементы являются ключевым аспектом применения HTML и CSS. При включении элемента в DOM возникают новые отношения между элементами, которые разделяются на родительские, дочерние и братские.

Родительский элемент – элемент, непосредственно содержащий внутри себя рассматриваемый элемент.

Предок – элемент, находящийся на несколько уровней выше в дереве документов и содержащий в себе рассматриваемый элемент.

Дочерний элемент – элемент, содержащийся непосредственно внутри рассматриваемого элемента.

Потомок – элемент, находящийся внутри рассматриваемого элемента на несколько уровней ниже.

Братский элемент – элемент, имеющий общий родительский элемент с рассматриваемым.

Таким образом для обработки нужных нам элементов с одинаковыми атрибутами, мы будем использовать уникальный путь вложенности элемента.

На рисунке 9 изображен последовательный путь элемента тега pи атрибута description.



Рисунок 9 ‒ Селектор тега p

Такие указатели на элемент, как мы уже говорили, называют селектором. Стоит отметить, что селектором может выступать, как и просто атрибут элемента, так и вложенная конструкция. Таким образом, конструкция, изображённая, на рисунке 9, показывает фрагмент дерева документа, который можно увидеть на рисунке 10:



Рисунок 10 ‒ Фрагмент дерева

С помощью рассматриваемого селектора мы получим контекст тега р – «текст».

**2. 3 Инструменты PHPQuery**

PHPQuery представляет собой библиотеку для PHP, позволяющую работать с селекторами элементов, а не просто работать с DOM как с простым текстом, используя сложные структуры регулярных выражений.

В первую очередь нам необходимо включить эту библиотеку в наш PHP-файл, чтобы ей в дальнейшем пользоваться. В PHP это делается данным образом:

<?php require\_once 'phpQuery.php';

Библиотека в этом случае должна лежать в той же папке, что и файл.

Итак, начало работы с библиотекой начинается с phpQuery::newDocument($str), где переменная $str - это HTML-код для разбора, пусть в ней будет лежать уже знакомый нам фрагмент дерева документа из рисунке 10. В результате вернется объект, к которому мы сможем применять определенные методы:

$str **=** '<body><div class="container"><p class=”description”>текст</p></div></body>';

$pq **=** phpQuery::newDocument($str);

Теперь в переменной $pq лежит объект. У этого объекта есть метод find, который параметром принимает селектор, а делает следующее: ищет внутри HTML кода из $pq элементы подпадающие под этот селектор.

$elem **=** $pq**->**find('body > div.container > p.description')->text();

echo $elem;

Сразу же преобразуем полученные данные в текст с помощью функции text(), выведем и получим содержимое тега p – «текст».

**3 Реализация**

Реализуем программу, которая будет анализировать цены на определённые товары в различных интернет-магазинах и находить тот, в котором данный товар будет иметь наименьшую стоимость. Для примера рассмотрим три интернет-магазина техники: «reStore», «М.видео» и «Цифрус», в которых мы будем сравнивать цену на смартфон Apple iPhone XS объёмом памяти 64GB и расцветки «Space Gray».



Рисунок 11 ‒ Страница товара

На рисунке 11 изображена страница данного товара, на которой мы можем видеть, что его цена находится в теге «div» и классом «product\_\_line». Получим его селектор:

body > div.global.active.c-active.global-fixed-header > div.product > div:nth-child(1) > div.product-info > div.product-info-right > div.product-buy-info > div.product\_\_line > div > span

Проделав описанные в главе 2 пункте 3 «Инструменты PHPQuery» шаги подготовки для работы с библиотекой, воспользуемся методом find() PHPQuery, чтобы отыскать элемент на странице по полученному селектору. Преобразуем найденные данные в текст и сохраним в элемент массива $number\_price[1]. Но, если мы попытаемся сравнивать полученные таким образом цены, то компилятор выдаст ошибку, т. к. в полученных данных с ценой на товар могли остаться пробелы и символы, обозначающие валюту. Чтобы этого не произошло, преобразуем строку функцией:

$number\_price[1] = preg\_replace("/[^0-9]/", '', $number\_price[1]);

Данная функция удаляет из строки $number\_price[1] все символы, кроме цифр. Теперь строку можно сравнивать, а также сортировать.

Аналогично обработаем другие интернет-магазины, а после отсортируем полученный массив по убыванию. Таким образом, мы будем знать, что наименьшая цена хранится в последнем элементе массива, в нашем случае это будет 3-й элемент массива.

Выведем полученные данные и выделим зелёным цветом оптимальный вариант для покупки смартфона. Результат на рисунке 12:



Рисунок 12 ‒ Результат работы программы-парсера

Мы видим, что программа-парсер сначала проанализировала страницы интернет-магазинов, ссылки на которые были ей даны, нашла нужную нам информацию ‒ цены на интересующий смартфон, отсортировала данные и вывела магазины в порядке убывания и выделила самый оптимальный интернет-магазин для покупки мобильного телефона.

**4 Недостатки парсеров**

Главным недостатком программ-парсеров, основанных на анализе DOM страниц, является то, что программа начинает вести себя непредсказуемо, если сайт-донор изменяет свою HTML-страницу. В таком случае программу придётся изменять.

Также программа-парсер не может быть универсальная ‒ вся её структура опирается на исходный код страницы, с которой осуществляется сбор информации. Таким образом, если вам необходимо осуществлять анализ данных с различных источников, то будет необходимо написать программу для каждого интернет-ресурса.

**ЗАКЛЮЧЕНИЕ**

В рамках проведенной работы был изучен принцип работы программ-парсеров, основанных на анализе дерева документа, была разработана программа-парсер с использованием языка PHP и разметки HTML. В процессе разработки программы были изучена библиотека PHPQuery.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1 Introduction to the DOM // (Engl). – URL: https://www.w3.org/TR/WD-DOM/introduction.html [28 Ноября 2018]

2 jQuery Library // (Engl) ‒ URL: [https://packagist.org/packages/bscheshir/phpquery [2](http://php.net/manual/mysqli.query.php%20%5B2) декабря 2018]

3 Бен Хеник. HTML и CSSS Путь к совершенству ‒ 2011.

**ПРИЛОЖЕНИЕ**

<!DOCTYPE html>

<html lang="ru">

<head>

 <meta charset="utf-8" >

 <title>parser</title>

 <link href="css.css" rel="stylesheet">

</head>

<body>

 <div>

<?php

set\_time\_limit(0);

include\_once 'phpQuery.php';

$string = 'https://www.re-store.ru/catalog/MT9E2RU-A/';

$html = file\_get\_contents($string);

$obj = phpQuery::newDocument($html);

$price = pq('.product-price')->text();

$number\_price[1] = $price;

$number\_price[1] = preg\_replace("/[^0-9]/", '', $number\_price[1]);

$string = 'https://www.mvideo.ru/products/smartfon-apple-iphone-xs-64gb-space-grey-mt9e2ru-a-30040014';

$html = file\_get\_contents($string);

$obj = phpQuery::newDocument($html);

$price = pq('body > div.wrapper > div > div.o-container.o-container\_\_section\_no-scroll.u-bg-white.sel-main-holder > div.o-container\_\_section.u-mb-40.sel-pdp-container > div.o-container\_\_row > div.o-container\_\_price-column > div > div.o-pay > div.c-pdp-price > div > div.c-pdp-price\_\_offers > div.c-pdp-price\_\_current.sel-product-tile-price');

$number\_price[2] = $price;

$number\_price[2] = preg\_replace("/[^0-9]/", '', $number\_price[2]);

$string = 'https://www.cifrus.ru/description/1/iphone\_xs\_2097\_64gb\_grey?gclid=EAIaIQobChMI1oivs8Cp3wIVx8qyCh1S9A25EAQYASABEgLnwvD\_BwE';

$html = file\_get\_contents($string);

$obj = phpQuery::newDocument($html);

$price = pq('#priceUpdate')->text();

$number\_price[3] = $price;

$number\_price[3] = preg\_replace("/[^0-9]/", '', $number\_price[3]);

$name[1] = 'reStore';

$name[2] = 'М.видео';

$name[3] = 'Цифрус';

asort($number\_price);

phpQuery::unloadDocuments();

?>

<h5 class="title">Выведем цены от наибошей до самой низкой стоимости</h5>

<div class="prices">

 <div class="price">

 <p><?php echo $name[1] ?></p>

 <div class="line"></div>

 <p><?php echo $number\_price[1] ?></p>

 </div>

 <div class="price">

 <p><?php echo $name[2] ?></p>

 <div class="line"></div>

 <p><?php echo $number\_price[2] ?></p>

 </div>

 <div class="lowest\_price">

 <p><?php echo $name[3] ?></p>

 <div class="lowest\_line"></div>

 <p><?php echo $number\_price[3] ?></p>

 </div>

 <p class="text">Наименьшая цена в магазине - <?php echo $name[3] ?></p>

</div>

 </div>

</body>

</html>