Олешко И.А.
Научный руководитель: к.т.н., доцент А.А. Фомин
Муромский институт (филиал) федерального государственного образовательного учреждения высшего образования «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»
602264, г. Муром, Владимирская обл., ул. Орловская, 23
E-mail: oleshko.ilya@gmail.com

Проблемы агрегирования web-ресурсов в формате XML

Большое количество онлайн торговых площадок создает проблему выбора для конечного пользователя. Поэтому актуальной являются задачи агрегирования информации с разных ресурсов и представления ее в виде обобщенного списка. Одним из подходов к решению этой задачи является Web Mining.

Web Mining – это сбор данных (парсинг) с последующим сохранением в нужном формате. Фактически, задача сводится к написанию HTTP парсеров [2].

Существует несколько подходов к извлечению данных:

- Анализ DOM дерева, использование XPath;
- Парсинг строк;
- Использование регулярных выражений.

Анализ DOM дерева основывается на понятии иерархического дерева. Согласно DOM модели, каждый тег образует узел дерева с типом «элемент». Вложенные в него теги становятся дочерними узлами. Используя этот подход, данные можно получить напрямую по идентификатору, имени или других атрибутов элемента дерева (таким элементом может служить параграф, таблица, блок и т.д.). Кроме того, если элемент не обозначен каким-либо идентификатором, то к нему можно добраться по некоему уникальному пути, спускаясь вниз по DOM дереву.

Достоинства подхода:

- возможность получить данные любого типа и любого уровня сложности;
- возможность получить значение элемента, прописав путь к нему.

Недостатки подхода:

- различные HTML / JavaScript «движки» по-разному генерируют DOM дерево, поэтому нужно привязываться к конкретному «движку»;
- путь элемента может измениться, поэтому, как правило, такие парсеры рассчитаны на кратковременный период сбора данных;
- DOM путь может быть сложный и не всегда однозначный.

Следующим эволюционным этапом анализа DOM дерева является использования XPath – т.е. путей, которые широко используются при парсинге XML данных. Суть данного подхода в том, чтобы с помощью некоторого простого синтаксиса описывать путь к элементу без необходимости постепенного движения вниз по DOM дереву [1].

Парсинг строк базируется на построчном переборе кода страницы или документа с целью поиска заранее известного шаблона, и последующего извлечения интересующей информации. Для составления сложных парсеров данный метод не подходит, но в рамках метода анализа строк статичной структуры работает эффективней чем анализ DOM дерева или XPath.

Регулярные выражения – самый распространенный способ парсинга различных ресурсов [2]. Несмотря на кажущуюся простоту использования, этот подход является в корне неверным. Основной проблемой регулярных выражений является их размер, при парсинге небольшого объема информации это не сильно ощутимо. Но когда объем и сложность структуры данных возрастает использование регулярных выражений значительно осложняется. Регулярные выражения необходимо использоваться только для извлечения данных, которые имеют строгий формат – электронные адреса, телефоны и т.д., в редких случаях – адреса, шаблонные данные.

При XML парсинге объемных документов часто используются методы:

- Simple XML;
- DOM;

- Xml_parser;
- XMLReader.

Simple XML.

Достоинства: простота работы, работа «из коробки» (требует библиотеки libxml которая включена практически на всех серверах).

Недостатки: низкая скорость, вызванная необходимостью размещения всего файла в памяти и составления дерева документа в отдельном массиве.

DOM.

Достоинства: простота обработки результатов.

Недостатки: очень низкая скорость работы и высокие затраты памяти.

Из-за большого объема исходного файла эти методы использовать не представляется возможности, так как они могут очень сильно замедлить работу сервера.

Xml_parser и XMLReader.

Оба способа выполняют построчное чтение файла, что подходит идеально для поставленной задачи.

Основное различие между Xml_parser и XMLReader в том, что в первом случае необходимо писать собственные функции, которые будут реагировать на начало и конец тэга [1].

Метод Xml_parser работает через 2 триггера – тэг открыт, тэг закрыт. Дальнейшие данные не учитываются, что значительно ускоряет работу метода.

Методы были протестированы на XML документах с различным объемом, итоговые данные приведены в таблице 1.

Таблица 1. Результаты замера скорости чтения XML документа.

Метод	Время выполнения (19 Mb)	Время выполнения (190 Mb)
Simple XML	0.46 сек	4.56 сек
DOM	0.52 сек	4.09 сек
xml_parser	0.22 сек	2.25 сек
XML Reader	0.26 сек	2.18 сек

Проблемы при парсинге данных:

– использование JavaScript / AJAX / асинхронных загрузок очень усложняют написание парсеров;

– различные «движки» для рендеринга HTML могут выдавать разные DOM деревья;

– большие объемы данных требуют писать распределенные парсеры, что влечет за собой дополнительные затраты на синхронизацию.

Нельзя однозначно выделить подход, который будет 100% применим во всех случаях, поэтому необходимо комбинировать существующие методы, что позволит компенсировать слабые стороны отдельных элементов парсинга.

Литература

1. Ward Jacob Instant PHP Web Scraping. – Birmingham: Packt Publishing Ltd. – 2013. – 60 p.

2. Markov Z., Larose D.T. Data-mining the Web: Uncovering Patterns in Web Content, Structure, and Usage // Journal of Statistical Software. – Vol. 25. – Book Review 1. – 2008. – pp. 1–3.