Инженерный вестник Дона, №12 (2023)
ivdon.ru/ru/magazine/archive/n12y2023/8904

Проектирование приложения для сбора данных из сторонних интернет-источников

С.Э. Шмаков

Мордовский государственный университет им. Н.П. Огарёва, Саранск

Аннотация: В данной статье рассматриваются основные принципы и шаблоны проектирования приложения для сбора данных из сторонних источников. Проведено исследование различных способов получения данных, включая веб-скрапинг, использование API и парсинг файлов. Также описываются различные подходы к извлечению информации из структурированных и неструктурированных источников.
Ключевые слова: интернет-источники, API, парсинг, web, веб, безголовый браузер, скрапинг, etag, сбор данных.

В современном мире веб-технологии не стоят на месте. С каждым днем появляется все больше и больше различных систем, упрощающих жизнь людей. Например, интернет-магазины, приложения с доставкой еды и продуктов, развлекательные порталы, форумы и много всего, что позволит интернет-пользователю найти, купить, продать или заказать что-либо. Однако становится сложно ориентироваться во всем этом многообразии сервисов. Один из способов решения этой проблемы – это сервисы-агрегаторы.

Сервис-агрегатор — это приложение клиент-сервер, объединяющее в себе информацию из нескольких источников. Клиент-сервер (англ. Client-server) — сетевая архитектура, в которой устройства являются либо клиентами, либо серверами. Клиентом (front end) является запрашивающая машина (обычно компьютер пользователя), сервером (back end) — машина, которая отвечает на запрос. Оба термина (клиент и сервер) могут применяться как к физическим устройствам, так и к программному обеспечению [1]. В качестве примера такого приложения можно привести агрегатор ресторанов и кафе, который не только имеет контракт с перечисленными заведениями, но и свою доставку. Но данный агрегатор

сильно зависит от ручного труда, так как позиции ресторанов и информация о них заполняется вручную менеджерами этого сервиса. Данные, взятые из основных сайтов-источников, являются статичными и изменяются очень редко. В статье же речь пойдет о разработке автоматизированных сервисов-агрегаторов, имеющих дело с источниками, данные которых являются динамическими и новые записи появляются чаще, чем их сможет вручную заполнить команда менеджеров.

Способы получения данных из сторонних источников

Самый надежный способ получения данных из сторонних источников – это использование Application Programming Interface (далее API). При таком подходе данные будут поставляться регулярно, с наименьшим кол-вом ошибок и процесс настройки взаимодействия займет немного времени. Однако не все источники имеют свой API интерфейс, а если и имеют, могут содержать ограниченных объем информации, в отличие от основного сайта.

Второй способ — это непосредственная коммуникация с владельцем источника. Это наиболее сложный и длительный процесс и наименее успешный, так как владелец приложения может быть закрыт для предложений сотрудничества или он не обладает необходимыми знаниями, временем для предоставления необходимой информации.

Третий способ — парсинг стороннего источника.

Парсинг сторонних источников

Этот способ получения данных менее приоритетный по сравнению с предыдущими и имеет свои плюсы и минусы:

Преимущества:

– Не требует коммуникации с владельцем источника;

– Является легитимным с юридической точки зрения при указании ссылки на источник;

– Автоматизированный процесс;

– Возможность сбора данных почти из любого веб-приложения.

Недостатки:

– Долгая разработка;

– Схема парсинга источника описывается разработчиком, обычный пользователь не сможет добавить новый объект сбора данных (существует алгоритм парсинга, при котором обычный пользователь сможет добавлять новые источники, однако эта тема для отдельной статьи);

– Источник при развертывании может использовать алгоритм минимизации Cascading Style Sheets (далее css) стилей, и если при разработке схемы использовать имена классов css, то после очередного развертывания приложения-источника (например, после выхода очередного патча или новой версии) схему придется изменять;

– Описанная ранее схема после изменения дизайна источника становится неактуальной.

При наличии всех этих недостатков, парсинг сайта тем не менее является единственным способом получения данных из стороннего источника при отсутствии API-интерфейса и возможности непосредственного взаимодействия с владельцем источника.

Подходы к извлечению данных

Парсер – это, как правило, сервис, который на входе получает данные об источнике, а на выходе отдает структурированные данные, например, в формате JavaScript Object Notation (далее JSON). JSON — текстовый формат

обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования [2].

Данные, предоставляемые разработчиком приложению, могут отличаться в зависимости от архитектуры парсера – это может быть «сырой» HyperText Markup Language (далее html). HTML – код или ссылка на источник. Это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице разные элементы: текст, картинки, таблицы и видео [3].

Самый удобный подход при создании парсера – это использование так называемого «безголового браузера» (англ. headless browser). «Без головы» просто означает, что нет графического пользовательского интерфейса (GUI). Вместо взаимодействия с визуальными элементами, как обычно (например, с помощью мыши или сенсорного устройства), вы автоматизируете варианты использования с помощью интерфейса командной строки (CLI). [4] В качестве примера можно привести chromium, который, в отличии от самого браузера, отправляющего запросы на сервер, осуществляет парсинг полученного от сервера HTML и строит Document Object Model (далее DOM) дерево, при этом имеет несколько особенностей [5]:

– Нет рендеринга реального контента. То есть все, что он делает, отображается в его памяти.

– Минимальные запросы внутренней памяти. Поскольку нет необходимости рендерить тяжелые portable network graphics (далее PNG) и изображения, потребности будут намного меньше.

– Более высокая скорость. Опять же, это связано с тем, что не придется тратить время на отображение контента на реальном дисплее.

– Наличие программного интерфейса для оперативного и удобного управления. Нет классических кнопок, окон, иконок.

С помощью безголового браузера можно описать схему парсинга, в которой будет проиллюстрирован процесс взятия данных, например:

1. Перейти на страницу сайта;
2. Найти элемент с классом «title»;
3. Извлечь текст из найденного элемента;
4. Зафиксировать извлеченные данные.

Процесс, описанный выше, лишь пример. Для парсинга больших объемов данных понадобится гораздо больше шагов, которые будут более подробно описаны на выбранном языке программирования.

Другие подходы парсинга, к примеру, отправка get-запроса на сайт-источник, взятие html и последующее извлечение информации, являются менее эффективными, так как современные сайты имеют защиту от запросов сторонних доменных имен.

Также подход, использующий безголовый браузер, при определенной настройке, направленной на имитацию действий реального человека, может обойти даже серьезные средства защиты. Например, можно обойти даже Web Application Firewall, который может использовать под капотом нейронные сети для отслеживания и пресечения вредоносных действий.

Шаблон разработки парсера

При использовании безголового браузера в качестве средства извлечения данных, парсинг даже одной страницы является ресурсоемкой операцией. Процесс извлечения данных – это не мгновенная операция. Среди мгновенных можно выделить запросы в базу данных (далее БД). Чтобы осуществлять тысячи таких процессов, необходимо использовать такую структуру данных, как «очередь».

Очередь необходима для того, чтобы распараллелить процессы, при этом не задействуя все ресурсы сервера.

Принцип такого подхода прост. Процесс парсинга одной страницы сайта – это одна задача в очереди. Чтобы не превысить ресурсы сервера и распараллелить парсинг нескольких страниц – нужно запускать несколько задач одновременно. Количество запущенных процессов зависит от мощности машины, на которой запущена система. Чем меньше ресурсов выделено для сервера, тем меньше одновременно запущенных задач.

Парсинг файлов, таких, как изображения, видео и т. д., тоже отдельная для каждого объекта задача в очереди.

Парсинг файлов

Стоит отдельно выделить процесс извлечения файлов. Есть два подхода:

1. Извлечение ссылок на медиафайлы и дальнейшее их использование. Такой подход имеет большое преимущество, так как при нем не нужно хранить сами файлы, а только их ссылки. Такой подход не требует большого хранилища данных, но является неактуальным при парсинге современных сервисов, которые используют различные средства защиты, например «подписка файлов». При такой мере защиты все медиафайлы предоставляются пользователю на время и ссылка на файл может быть недоступна после его истечения.

При парсинге источника, на котором данные меняются динамически, например, интернет-магазина, необходимо периодически запускать парсер, чтобы актуализировать уже собранную информацию. В таком случае появляется проблема дублирования файлов, так как неизвестно какой именно файл мы получили по ссылке, новый или же тот же самый.

Чтобы решить данную проблему, нужно использовать etag файла.

2. Использование entity tag (далее etag).

ETag или entity tag — один из регламентируемых спецификацией Request for Comments (далее RFC) 7232, служебных заголовков протокола HTTP/1.1, который может быть установлен веб-сервером в фазе формирования ответа, на полученный от клиента запрос [6].

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — сетевой протокол прикладного уровня [7].

Преимущества протокола HTTP: простота, расширяемость, распространённость, документация на различных языках. Недостатки протокола HTTP: отсутствие "навигации", отсутствие поддержки распределенных действий [8].

Содержимое заголовка etag является идентификатором, значение которого прямо зависит от состояния загружаемого клиентом ресурса.

В качестве etag’a, можно использовать хэш самого бинарного файла, полученные при использовании какого-либо алгоритма хэширования, например, md5, который использует в качестве входных данных сообщение произвольной длины и выдает на выходе 128-битовое значение [9].

Если заголовки ответа на запрос файла не содержит etag, его нужно получить самостоятельно и сохранить в базу данных, чтобы в дальнейшем идентифицировать файл.

Итоги исследования

При разработке системы сбора данных из сторонних источников были выявлены основные принципы и шаблоны проектирования, которые позволяют эффективно и надежно получать и обрабатывать информацию.

Исследование различных способов получения данных показало, что веб-скрапинг и использование API являются наиболее распространенными методами. Выбор конкретного метода зависит от требований проекта и особенностей источника данных.

При извлечении данных из структурированных и неструктурированных источников применяются различные подходы. Использование регулярных выражений, CSS-селекторов и других технологий позволяет эффективно обрабатывать разные форматы данных. CSS-селекторы — это выражения, которые говорят браузеру, к какому элементу HTML нужно применить те или иные свойства CSS, определённые внутри блока объявления стиля [10].

При разработке парсера необходимо учитывать разные сценарии и возможные ошибки при парсинге данных. Важно обеспечить гибкость и масштабируемость парсера для работы с различными источниками данных.

Особое внимание было уделено разработке парсера для парсинга файлов с использованием механизма etag для идентификации обновлений. Этот механизм позволяет минимизировать трафик и время обработки при повторном сборе данных.

Итоги исследования и разработки подтверждают, что правильный выбор способов получения данных, подходов к извлечению информации, принципов разработки парсера и использование механизма etag являются ключевыми факторами для создания эффективной и надежной системы сбора данных из сторонних источников. Эти принципы и шаблоны проектирования могут быть применены в различных проектах, где требуется сбор данных для анализа и обработки.

Литература

1. Натальченко И.А. Анализ механизмов передачи крупных массивов данных через сеть интернет с помощью технологии веб-сервиса // Инженерный вестник Дона, 2008, №4. URL: ivdon.ru/ru/magazine/archive/n4y2008/98

2. Flanagan D. JavaScript: The Definitive Guide. O'REILLY Media Inc., 2008. 982 p.

3. MacDonald M. HTML5: The Missing Manual. O'REILLY, 2014. 433 p.

4. Chikovalni N. Web Scraping with a Headless Browser: A Puppeteer Tutorial, URL: toptal.com/puppeteer/headless-browser-puppeteer-tutorial (Дата обращения: 14.10.2023)

5. Flanagan D. JavaScript: The Definitive Guide. O'REILLY Media Inc., 2008. 982 p.

6. Grigorik I. High-Performance Browser Networking. O'REILLY Media Inc., 2013. 363 p.

7. Gourley D., Totty B. HTTP: The Definitive Guide. O'REILLY Media Inc., 2002. 617 p.

8. Бондаренко Т.В., Федотов Е.Н. Разработка HTTP сервера // Инженерный вестник Дона, 2018, №2. URL: ivdon.ru/ru/magazine/archive/N2y2018/5038

9. Васильева И.Н. Криптографические методы защиты информации // учебник и практикум для вузов / И. Н. Васильева. М.: Издательство Юрайт, 2023. 349 с.

10. Мейер Э. CSS: каскадные таблицы стилей. 3 изд. СПб.: Символ-Плюс, 2008. 576 с.

References

1. Natalchenko I.A. Inzhenernyj vestnik Dona, 2008, №4. URL: ivdon.ru/ru/magazine/archive/n4y2008/98

2. Flanagan D. JavaScript: The Definitive Guide. O'REILLY Media Inc., 2008. 982 p.

3. MacDonald M. HTML5: The Missing Manual. O'REILLY, 2014. 433 p.

4. Chikovalni N. Web Scraping with a Headless Browser: A Puppeteer Tutorial, URL: toptal.com/puppeteer/headless-browser-puppeteer-tutorial (Date of the application: 14.10.2023)

5. Flanagan D. JavaScript: The Definitive Guide. O'REILLY Media Inc., 2008. 982 p.

6. Grigorik I. High-Performance Browser Networking. O'REILLY Media Inc., 2013. 363 p.

7. Gourley D., Totty B. HTTP: The Definitive Guide. O'REILLY Media Inc., 2002. 617 p.

8. Korolenko I.A. Inzhenernyj vestnik Dona, 2023, №11. URL: ivdon.ru/ru/magazine/archive/n11y2023/8777

9. Vasil'eva I.N. Kriptograficheskie metody zashhity informacii [Cryptographic methods of information protection]. Uchebnik i praktikum dlja vuzov I. N. Vasil'eva. M.: Jurajt, 2023. 349 p.

10. Meyer E. CSS: kaskadnye tablicy stilej [CSS: Cascading Style Sheets]. 3rd ed. SPb.: Simvol-Pljus, 2008. 576 p.

Дата поступления: 18.11.2023

Дата публикации: 28.12.2023