ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 20 за 2019 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Китаев Е. Л., Скорнякова Р. Ю.

Скрейпинг «на лету»
внешних веб ресурсов,
управляемый разметкой
HTML страницы

Рекомендуемая форма библиографической ссылки: Китаев Е. Л., Скорнякова Р. Ю. Скрейпинг «на лету» внешних веб ресурсов, управляемый разметкой HTML страницы // Препринты ИПМ им. М.В.Келдыша. 2019. № 20. 31 с. <http://doi.org/10.20948/prepr-2019-20>
URL: <http://library.keldysh.ru/preprint.asp?id=2019-20>

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Е.Л. Китаев, Р.Ю. Скорнякова

**Скрейпинг «на лету»
внешних веб-ресурсов,
управляемый разметкой
HTML-страницы**

Москва — 2019

Китаев Е.Л., Скорнякова Р.Ю.

Скрейпинг «на лету» внешних веб-ресурсов, управляемый разметкой HTML-страницы

В работе изложен подход к отображению на веб-страницах данных из кросс-доменных ресурсов с использованием REST API и описан созданный на основе этого подхода инструмент, позволяющий извлекать и показывать на веб-странице метаданные размещенных в интернете html-документов, pdf-файлов и документов Word, а также микроданные и данные в формате JSON-LD. Инструмент включает в себя REST API на веб-сервере IIS и скрипты на языке JavaScript. Приведены примеры использования этого инструмента для создания списка организаций с меняющимися атрибутами, веб-страницы с текущими ценами на один и тот же товар в разных интернет-магазинах, списка научных статей. Созданный REST API допускает кросс-доменный доступ (CORS) и может быть использован при запросах из веб-страниц любых доменов.

Ключевые слова: веб-скрейпинг, микроразметка, семантическая разметка, микроданные, JSON-LD, REST API, CORS

Evgeny L'vovich Kitaev, Rimma Yuryevna Skornyakova

Scraping on the fly of external web resources, driven by HTML page markup

The paper presents an approach to displaying data from cross-origin resources on web pages using the REST API and describes a tool based on this approach that allows one to extract and display on the web page metadata of html-documents, pdf-files and documents Word posted on the Internet, as well as microdata and data in JSON-LD format. The tool includes the REST API on the IIS web server and JavaScript scripts. Examples of using this tool are given. The created REST API enables cross-origin resource sharing (CORS) and can be requested from web pages of any origins.

Key words: web scraping, semantic markup, microdata, JSON-LD, REST API, CORS

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 19-01-00069 А.

Оглавление

Введение	3
Структурированные данные в Сети. Микроразметки	4
О способах отображения на веб-странице данных из кросс-доменных ресурсов	10
Веб-скрейпинг. Инструменты	12
StructScraper — инструмент для отображения на веб-странице структурированных данных из различных веб-ресурсов.....	14
Примеры использования инструмента StructScraper	21
Заключение.....	29
Литература	30

Введение

Развитие интернет-технологий привело к тому, что компьютерные приложения все больше перемещаются во всемирную паутину. Современные программные и технические средства позволяют создавать полноценные клиент-серверные веб-приложения, когда сервер предоставляет данные, а их визуальное представление реализуется в клиентской части приложения. Такого рода одностраничные приложения (Single Page Applications – SPA) становятся все более популярными. Клиентская часть в таких приложениях, как правило, реализуется на языке JavaScript, а запросы к серверу через JavaScript API XMLHttpRequest или более современный его аналог fetch[1]. Для того чтобы упростить разработку такого рода приложений, созданы различные фреймворки, такие как Angular [38], React [39].

В Сети размещено большое количество данных, и, если эти данные открыты, естественно желание получать их динамически непосредственно из тех источников, где они были размещены изначально. На всемирную паутину можно смотреть как на распределенную базу данных, а в базах данных стараются соблюдать принцип нормализации: хранить данные в одном месте, избегая избыточности. Принцип этот может нарушаться для улучшения производительности, однако, если время ответа на запрос приемлемо, то лучше не хранить одни и те же данные в разных местах. Если данные могут меняться со временем, это может привести к их рассогласованию. Примерами задач, где извлечение данных из веб-ресурсов «на лету» может быть полезным, могут служить веб-страница со списком контактных данных организаций или веб-страница с ценами на один тот же товар в разных интернет-магазинах: контактные данные и цены со временем могут меняться. Но даже в тех случаях, когда исходные данные не должны меняться со временем, их автоматическое извлечение может быть полезным, позволяя избежать трудоемкого и чреватого ошибками ручного копирования. Примером может

служить список ссылок на размещенные в открытом доступе научные публикации. Преимущество извлечения «на лету» и моментального показа данных также в том, что такой подход не требует организации промежуточного хранения данных.

В общем случае задача извлечения информации из веб-ресурсов весьма сложна, поскольку данные, представленные в Сети, слабо структурированы. Они предназначены в первую очередь для прочтения человеком, а не для компьютерной обработки. О методах извлечения информации из веб-ресурсов можно прочитать в работах [2,3,4].

Дополнительную сложность для извлечения информации через XMLHttpRequest (fetch) создает правило одного источника [6,7]. В соответствии с этим правилом браузеры разрешают выполнять запросы к серверу, только если он находится в том же домене, что и веб-страница, из которой осуществляется запрос. Исключение составляют сервера, которые явно разрешают кросс-доменные запросы в соответствии с технологией CORS (Cross-Origin Resource Sharing) [9,10].

В настоящей работе излагается один из возможных подходов к отображению на «на лету» данных из кросс-доменных ресурсов – подход с использованием REST API. Этот подход позволяет обойти правило одного источника. В ней также описывается реализованный на основе этого подхода инструмент StructScraper. Инструмент позволяет отображать метаданные веб-ресурсов (для html – извлеченные из тегов <meta>, для документов формата pdf, doc и docx – из встроенных и пользовательских свойств), а также структурированные данные, содержащиеся в микроразметках «микроданными» [16] или JSON-LD[18].

Вначале более подробно расскажем о способах представления в Сети структурированных данных, возможных подходах к отображению на веб-страницах данных из кросс-доменных ресурсов, а также приведем краткий обзор имеющихся инструментов для извлечения данных из веб-ресурсов.

Структурированные данные в Сети. Микроразметки

Взгляд на всемирную паутину как на распределенную базу данных нашел свое отражение в концепции Linked Data [12]. В рамках этой концепции были разработаны модель представления данных RDF (Resource Description Framework) и язык запросов SPARQL. Примером может служить проект DBpedia[13], в котором модель RDF использована для представления данных Википедии. При размещении данных в интернете модель данных RDF реализуется, как правило, в формате XML или при помощи добавления в html-документ семантической разметки в соответствии с предложенным в 2004 году разработчиками из консорциума W3C стандартом RDFa (Resource Description Framework in attributes) [14]. Язык RDFa богат и гибок, с его помощью можно выразить практически все что угодно, но это имеет и свою обратную сторону: язык довольно сложен, и при его реализации трудно

избежать ошибок. Поэтому веб-разработчики часто используют другие, более простые, способы семантической разметки. Способ семантической разметки, или, как ее еще называют, микроразметки, определяется используемым синтаксисом и словарем. В синтаксисе определяется, какие элементы html используются. Словарь определяет содержательную часть, семантику. Он представляет собой набор сущностей/классов и свойств.

Самым простым и исторически первым появившимся синтаксисом микроразметки является синтаксис с использованием тегов `<meta>`. Название свойства записывается в атрибут «name» а его значение — в атрибут «content». В качестве примера словаря, используемого в тегах `<meta>`, можно привести Дублинское ядро (Dublin Core) [20]. Вот так, например, с использованием словаря Дублинское ядро могут быть представлены сведения о публикации:

```
<meta name="DC.title" content="Живая публикация"/>
<meta name="DC.author" content="Горбунов-Посадов, Михаил Михайлович"/>
<meta name="DC.issued" content="2011" />
<meta name="DC.modified" content="2018-10-02"/>
```

Листинг 1. Микроразметка с использованием тегов `<meta>` и словаря Дублинское ядро.

Такой синтаксис микроразметки довольно примитивен, с его помощью неудобно представлять иерархические структуры или одновременно несколько объектов.

В 2005 году был предложен новый стандарт для микроразметки — «микроформаты» [15]. Язык «микроформатов» проще, чем RDFa, но с большими выразительными возможностями, чем язык с использованием тегов `<meta>`. При помощи него, в частности, значительно удобнее представлять иерархические структуры. Стандарт «микроформаты» объединяет в себе и синтаксис и словарь. Для передачи семантической информации в нем применяются существующие в стандарте html атрибуты `class`, `rel`, `rev`, `title`. Никакие новые атрибуты для обозначения сущностей и их свойств не вводятся. Примерами сущностей из словаря микроформатов могут служить `hCard` — контактные данные, `hProduct` — товар. Пример для контактных данных организации приведен в листинге 2.

Стандарт этот одно время был довольно популярным, однако его недостатки, такие как объединение в одном стандарте синтаксиса и словаря, а также резервирование значений атрибута `class` для придания семантики, привели к тому, что после появления других стандартов микроразметки он практически перестал использоваться.

```

<div class="vcard">
  <div>
    <span class="fn org">ИПМ им. М.В.Келдыша РАН</span>
  </div>
  <div class="adr">
    <span class="locality">Москва</span>,
    <span class="street-address">Миусская пл., д.4</span>
  </div>
  <div>Телефон: <span class="tel">+7 499 978-13-14</span></div>
  <div>
    <span class="url">
      <span class="value-title" title="http://keldysh.ru"></span>
    </span>
  </div>
</div>

```

Листинг 2. Микроразметка с использованием микроформатов.

В 2008 году сотрудником Google и на тот момент участником W3C Яном Хиксоном был предложен стандарт синтаксиса микроразметки – «микроданные» [16]. Стандарт «микроданные» является частью стандарта HTML5. Цель его – добавление семантики к имеющимся html-элементам. Для этого в стандарте вводятся специальные глобальные атрибуты `itemscope`, `itemtype`, `itemprop`, `itemid`, `itemref`. Атрибут `itemscope` используется для того, чтобы сделать html-элемент узлом микроданных. Атрибут `itemtype` задает тип данных. Как правило, тип выбирается из какого-нибудь словаря. Атрибут `itemprop` задает имя свойства. В атрибуте `itemid` задается глобальный идентификатор. Атрибут `itemref` используется в тех случаях, когда свойство не содержится внутри узла микроданных.

```

<div itemscope itemtype="http://schema.org/Organization">
  <span itemprop="name"> ИПМ им. М.В.Келдыша РАН </span>
</div>
Контакты
  <div itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
    Адрес:
    <span itemprop="postalCode">125047</span>,
    <span itemprop="addressLocality">Москва</span>,
    <span itemprop="streetAddress"> Миусская пл., д.4</span>
  </div>
  Телефон:<span itemprop="telephone">+7 499 978-13-14</span>,
  Факс:<span itemprop="faxNumber">+7 499 972-07-37</span>,
  Электронная почта: <span itemprop="email">office@keldysh.ru</span>
</div>
</div>

```

Листинг 3. Микроразметка с использованием микроданных и словаря Schema.org.

Листинг 3 содержит пример разметки микроданными контактных данных организации. В этом примере используется словарь Schema.org[22],

объявленный в 2011 году стандартом семантической разметки данных в Сети поисковыми системами Google, Bing и Yahoo. Впоследствии к ним присоединился и российский поисковик Яндекс. В настоящее время это самый популярный словарь.

Возможности стандарта «микроданные» вместе со словарем Schema.org полностью перекрывают возможности микроформатов. Однако и этот стандарт не лишен недостатков. Основным предметом критики в нем стала несовместимость с RDF.

В 2009 году разработчиками RDFa была предложена альтернатива микроданным – стандарт RDFa Lite[17], являющийся сокращенной версией RDFa. Из множества атрибутов RDFa в этом стандарте используются только vocab, typeof, property, resource и prefix. Вот так, например, в RDFa Lite будет выглядеть микроразметка организации, эквивалентная разметке из Листинга 3.

```
<div vocab="http://schema.org/" typeof="Organization">
  <span property="name"> ИПМ им. М.В.Келдыша РАН </span>
  <div>
    Контакты
    <div property="address" typeof="PostalAddress">
      Адрес:
      <span property="postalCode">125047</span>,
      <span property="addressLocality">Москва</span>,
      <span property="streetAddress">Миусская пл., д.4</span>
    </div>
    Телефон:<span property="telephone">+7 499 978-13-14</span>,
    Факс:<span property="faxNumber">+7 499 972-07-37</span>,
    Электронная почта: <span property="email">office@keldysh.ru</span>
  </div>
</div>
```

Листинг 4. Микроразметка с использованием RDFa Lite и словаря Schema.org.

Преимущество RDFa Lite в сравнении с микроданными в том, что разметка RDFa Lite может быть расширена до RDFa. В 2015 году RDFa Lite стал официальным стандартом, рекомендуемым консорциумом W3C, однако стандарт «микроданные» продолжает использоваться веб-разработчиками. В настоящее время он поддерживается и развивается сообществом WHATWG – сообществом, которое занимается разработкой стандартов, реализуемых в веб-браузерах.

В 2010 году был предложен еще один стандарт синтаксиса микроразметки – JSON-LD (JSON for Linking Data) [18, 19]. Он разрабатывался как альтернатива формату RDFa программистами, которые в своем проекте использовали для внутреннего хранения данных широко распространенный в веб-приложениях формат JSON. При этом данные, которые они извлекали и обрабатывали, хранились в Сети в формате RDFa. Идея состояла в том, чтобы данные в веб-документах, которые предназначены для программной

обработки, тоже хранились в формате JSON. В этом случае отпадает необходимость преобразования одного формата в другой. В 2014 году JSON-LD стал официальным стандартом, поддерживаемым консорциумом W3C. JSON-LD совместим с RDF, он является еще одним способом сериализации RDF.

В html-документе данные формата JSON-LD помещаются в тег `<script>` с атрибутом `type="application/ld+json"`. Ключевые слова, входящие в синтаксис JSON-LD, начинаются с символа `"@"`. Ниже приведены примеры разметок для контактных данных организации (листинг 5) и сведений о публикации (листинг 6).

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Organization",
  "name": "ИПМ им. М.В.Келдыша РАН",
  "address":
  {
    "@type": "PostalAddress",
    "postalCode": "125047",
    "addressLocality": "Москва",
    "streetAddress": "Миусская пл., д.4"
  },
  "telephone": "+7 499 978-13-14",
  "faxNumber": "++7 499 972-07-37",
  "email": "office@keldysh.ru"
}
</script>
```

Листинг 5. Микроразметка контактных данных организации с использованием JSON-LD и словаря Schema.org.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Article",
  "name": "Живая публикация",
  "author":
  {
    "@type": "Person",
    "name": "Горбунов-Посадов, Михаил Михайлович"
  },
  "datePublished": "2011",
  "dateModified": "2018-10-02"
}
</script>
```

Листинг 6. Микроразметка сведений о публикации с использованием JSON-LD и словаря Schema.org.

Синтаксис JSON-LD имеет свои преимущества и недостатки по сравнению с микроданными и RDFa Lite. Микроданные и RDFa Lite накладывают разметку непосредственно на данные, которые отображаются в браузере, при помощи этих разметок реализуется принцип WYSIWYG (what you see is what you get). Однако такой способ внесения семантики усложняет html-код, он становится менее читабельным. В случае JSON-LD html-код проще читать и проще извлекать информацию программным путем, однако связь между данными, предназначенными для прочтения человеком и для компьютерной обработки, отсутствует.

Микроразметки используются поисковиками для представления данных о сайте в виде так называемых расширенных сниппетов. В отличие от обычных сниппетов, где информация о сайте представляет собой выдержку из неструктурированного текста, расширенные сниппеты представляют структурированную информацию, которая лучше отражает содержание сайта.

В рунете в настоящее время чаще встречается разметка микроданными. Объясняется это, по-видимому, тем, что популярный в России поисковик Яндекс использует для формирования расширенных сниппетов в поисковых выдачах главным образом этот тип разметки. Реализация стандарта JSON-LD, появившегося позднее, пока только в планах. На зарубежных сайтах формат JSON-LD по частоте использования обошел микроданные [24]. Использовать JSON-LD рекомендует самая популярная поисковая система Google [23].

Общая статистика использования различных форматов структурированных данных по данным сайта [24] на момент написания этой работы такова:

- 48.1% сайтов не содержит структурированных данных;
- формат RDFa использует 45.4% сайтов (RDFa Lite в этой статистике отдельно не выделяется);
- теги <meta> используют 26.6% сайтов;
- формат JSON-LD используют 23.5% сайтов;
- микроданные используют 16.1%;
- микроформаты используют лишь 0.1%.

При этом один и тот же сайт может использовать разные форматы структурированных данных.

Можно отметить также, что частота использования всех упомянутых видов микроразметки, кроме микроформатов, по данным того же сайта со временем увеличивается [25]. Число сайтов без разметки в течение 2018 года уменьшилось на 6% и стало меньше половины. Разметка RDFa и разметка, базирующаяся на тегах <meta>, стали использоваться на 5.6% чаще, JSON-LD – на 4.6%, а микроданные – на 2.7% чаще.

Отдельный случай представляют веб-ресурсы, где данные хранятся в форматах Word и pdf. Для добавления структурированных данных в эти форматы можно использовать встроенные и пользовательские свойства. Ясно, что сложную структуру таким образом добавить нельзя. Возможности

представления структурированных данных в этом случае приблизительно такие же, как и в случае использования тегов meta в html-документах.

О способах отображения на веб-странице данных из кросс-доменных ресурсов

Как уже отмечалось, возможность получения «на лету» данных из кросс-доменных источников с помощью XMLHttpRequest API ограничена правилом одного источника (Same Origin Policy) [6, 7]. Правило одного источника, на русском языке именуемое также «правилом одного домена», «принципом одинакового источника», «правилом ограничения домена», «политикой одного домена» и т.п., было введено из соображений безопасности. Его цель — дать возможность коду, содержащемуся в веб-странице одного сайта, свободно взаимодействовать с ресурсами этого же сайта и максимально ограничить такое взаимодействие с ресурсами других сайтов. Если бы взаимодействие с ресурсами других сайтов не было ограничено, написанный злоумышленником скрипт из страницы, загруженной с какого-нибудь сайта, мог обратиться, например, к почтовому серверу пользователя через сессию, открытую в другом окне браузера, получить его почту или отправить от его имени письмо.

Считается, что два ресурса имеют один и тот же источник, если в их веб-адресах совпадают протокол, домен и порт. Запросы из веб-страниц к сторонним источникам (cross-origin requests) в русскоязычной литературе называют кросс-доменными запросами, хотя это и не совсем корректно. В соответствии с правилом одного источника браузеры запрещают производить кросс-доменные запросы через XMLHttpRequest API. До появления в недавнем времени технологии CORS [9,10] такие запросы были запрещены полностью. Технология CORS частично сняла этот запрет. Запросы к стороннему серверу можно осуществлять, но только если сторонний сервер дает на это явное разрешение. При этом сервер также контролирует детали кросс-доменных запросов: разрешенные методы, возможности передачи куки, авторизующих заголовков и т.п. Реализована технология CORS посредством специального протокола, являющегося надстройкой над HTTP-протоколом. Подробное изложение этой технологии с поясняющими примерами имеется в книге [11].

В зависимости от того, что представляет собой кросс-доменный ресурс, и от того, допускает ли сервер, на котором он расположен, кросс-доменные запросы, для получения данных из него используются разные методы.

Идеальный вариант – когда сайт, данные которого мы хотим показывать на веб-странице, предоставляет эти данные через API с разрешенным кросс-доменным доступом. В этом случае получить данные можно непосредственно выполнив ajax-запрос к этому API.

Если API отсутствует, но ресурс, данные которого нужны, представляет собой html- или xml-документ, к которому разрешен кросс-доменный доступ, то

через ajax-запрос можно получить текст документа и выполнить его разбор средствами JavaScript.

Если ресурс представляет собой html- или xml-документ, но кросс-доменные запросы к нему не разрешены, то можно получить содержимое документа с помощью запроса к одному из имеющихся в Сети прокси-сервисов, позволяющих обойти правило одного источника. Примерами таких сервисов могут служить All Origins (<http://multiverso.me/AllOrigins/>) или crossorigin.me (<https://corsproxy.github.io/>). Эти сервисы разрешают кросс-доменный доступ.

Для получения данных из кросс-доменных ресурсов используют также технику JSONP (JSON with padding) [8]. Она была изобретена до появления технологии CORS. JSONP использует для запросов не XMLHttpRequest, а тег <script>. Из тега <script> обращения к кросс-доменным ресурсам не запрещены. Сервер, к которому осуществляется запрос, должен предоставлять данные в формате JSONP. Формат JSONP – это JSON, обернутый вызовом JavaScript-функции. Существенное ограничение техники JSONP по сравнению с технологией CORS в том, что она позволяет делать только GET-запросы.

Еще один метод, используемый для обхода правила одного источника, – метод промежуточного сервера. В нем запрос направляется не напрямую к стороннему серверу, а к промежуточному серверу, находящемуся в том же домене, что и веб-страница (рис.1). Промежуточный сервер перенаправляет запрос к стороннему серверу и, получив от него ответ, посылает его клиенту.

Метод основан на том, что правило одного источника относится только к клиентской части веб-приложения, оно реализуется браузерами. Запросы от сервера к серверу правило одного источника никак не ограничивает. Со стороны стороннего сервера могут быть ограничения другого характера. Некоторые сервера допускают запросы только через браузеры и имеют механизмы защиты от роботов. В некоторых случаях блокируется веб-адрес, если запросы с него приходят слишком часто. Однако в большинстве случаев такой подход срабатывает.

Рис. 1. Запрос к веб-ресурсу через прокси-сервер

Метод промежуточного сервера особенно удобен в тех случаях, когда данные, поступающие от стороннего сервера имеют сложную структуру, а в клиентской части нужна только часть этих данных. Примером могут служить данные формата Word и PDF. Извлечение данных из ресурсов таких форматов средствами JavaScript невозможно, нужны специальные библиотеки,

работающие с этими форматами. И, хотя с применением новой технологии Web Assembly [40] появилась возможность производить обработку подобных форматов в клиентской части веб-приложения, написанный с применением этой технологии код у многих пользователей может не работать, поскольку поддержка этой технологии была включена во все основные версии браузеров лишь в конце 2017 года.

Появление технологии CORS расширило возможности применения техники промежуточного сервера. Если промежуточный сервер в соответствии с протоколом CORS допускает кросс-доменные запросы, то веб-страница, из которой происходит запрос, и промежуточный сервер не обязаны находиться в одном домене. Именно такой подход для извлечения «на лету» структурированных данных из кросс-доменных ресурсов используется в предлагаемом нами инструменте StructScraper, описание которого будет дано ниже.

Веб-скрейпинг. Инструменты

Автоматическое извлечение данных из веб-ресурсов называют веб-скрейпингом (web scraping). Словосочетание «web scraping» можно перевести как «выковыривание данных из веб-ресурсов». По-русски это звучит не очень благозвучно, поэтому это словосочетание используют без перевода. Веб-скрейпинг используют как в коммерческих, так и в исследовательских целях. Например, он может использоваться для отслеживания цен конкурентов, для получения биржевых сводок, для анализа состояния рынка труда, для получения сведений о погоде.

В Сети широко представлены разнообразные средства и инструменты для веб-скрейпинга [26, 27]. Среди них имеются библиотеки, настольные приложения, online-сервисы, облачные сервисы, сервисы типа DaaS, плагины к браузерам. Скрейперы в этих инструментах часто сочетаются с краулерами (поисковыми роботами), что позволяет извлекать информацию не только из конкретной веб-страницы, но и из страниц, на которые выводят гиперссылки с этой страницы. В этих случаях процесс извлечения информации может занимать довольно много времени. Некоторые инструменты предоставляют возможность запускать скрейперы по расписанию с определенной периодичностью. Поскольку современные веб-страницы часто бывают динамическими, часть инструментов для веб-скрейпинга дают возможность извлекать информацию из html-кода, который получается после отработки скриптов. Среди инструментов для веб-скрейпинга есть те, которые требуют программирования и те, которые не требуют, есть инструменты с открытым кодом, свободно-распространяемые, условно-бесплатные и полностью коммерческие.

Изначально инструменты веб-скрейпинга создавались с целью получения информации в структурированном виде и сохранения ее для дальнейшего анализа. Однако в последнее время в них все чаще стали включать средства для

создания REST API – программных интерфейсов к извлекаемым данным, что позволяет использовать извлекаемые данные в приложениях. Некоторые инструменты предлагают готовые API для определенных предметных областей, встречаются также предложения создания API под заказ.

Приведем примеры средств для веб-скрейпинга, которые могут быть полезны для решения рассматриваемой в этой работе задачи динамического включения в контент на веб-страницы кросс-доменных данных.

Scrapy

Одним из наиболее популярных средств веб-скрейпинга является Scrapy [28]. Scrapy – это бесплатный фреймворк с открытым кодом. Он включает в себя библиотеку на языке Python, облачный сервис для запуска веб-скрейперов, средства отладки и средство для создания REST API – ScrapyRT [29]. Создание веб-скрейпера требует программирования на языке Python. Веб-скрейпер создается как класс, наследуемый от одного из классов, включенных в библиотеку Scrapy. При помощи ScrapyRT можно создать REST API, который может возвращать как результат работы скрейпера, запущенного ранее, так и запускать скрейпер и возвращать результат его работы в режиме реального времени.

Apache Any23

Для извлечения из веб-ресурсов заключенных в них структурированных данных может быть полезен свободно распространяемый инструмент Apache Any23 (Anything To Triples) [30]. Он включает в себя библиотеку на языке Java, веб-службу и интерфейс командной строки. Извлеченные структурированные данные представляются в виде RDF. Any23 позволяет извлекать структурированные данные следующих форматов:

- RDF с сериализацией в виде XML, Turtle, Notation 3
- Rdfa
- Микроформаты
- JSON-LD
- Микроданные
- CSV
- YAML

Поддерживаются словари Dublin Core Terms, Description of a Career, Description Of A Project, Friend Of A Friend, GEO Names, ICAL, Ikif-core, Open Graph Protocol, BBC Programmes Ontology, RDF Review Vocabulary, schema.org, VCard, BBC Wildlife Ontology, XHTML и др. Дополнительно с помощью системы Open IE (Open Information Extraction system), которая из предложений естественного языка извлекает тройки, Any23 позволяет также получать структурированные данные из текстов на английском языке.

Веб-служба представляет собой REST API, которая может быть загружена с сайта Any23 и установлена на сервере пользователя. Ей можно передать в качестве аргументов URL интересующего ресурса и желаемый формат

представления полученных данных. Веб-служба извлечет все имеющиеся структурированные данные и представит их в запрашиваемом формате.

Import.IO

Среди коммерческих платформ для веб-скрейпинга одна из наиболее популярных – платформа Import.IO [31]. Она включает визуальные инструменты для создания скрейперов, облачную платформу для запуска скрейперов, а также разные REST API, среди которых имеется API, позволяющий запускать созданный ранее скрейпер и получать его результаты в режиме реального времени. API поддерживает CORS, что позволяет вызывать его из клиентской части веб-приложения.

Библиотеки

Упомянем также некоторые библиотеки с открытым кодом, которые могут быть использованы при программировании веб-скрейперов и которые не являются частью интегрированных платформ:

- Html Agility Pack (<https://html-agility-pack.net/>) – C# - библиотека; позволяет извлекать данные из html-страниц с использованием XPath;
- ScrapySharp (<https://github.com/rfleckner/ScrapySharp>) – расширение Html Agility Pack; позволяет извлекать данные с использованием CSS-селекторов, включает поддержку динамических страниц;
- Extruct (<https://github.com/scrapinghub/extract>) – библиотека на Python для извлечения структурированных данных; поддерживаемые форматы – JSON-LD, «микроданные», RDFa.

API Валидатора микроразметки от компании Яндекс

Компания Яндекс предоставляет API к своему онлайн-валидатору микроразметки [32]. При помощи него можно извлекать из веб-страниц структурированные данные форматов JSON-LD, RDFa, микроданные (microdata) и микроформаты. Для передачи данных в API необходимо указать HTML-код или URL страницы. Результат содержит все структурированные данные, которые имеются в странице, а также сообщения об ошибках в разметках. Доступ к API предоставляется по ключу. Имеется ограничение: не более 10 запросов в секунду.

StructScraper – инструмент для отображения на веб-странице структурированных данных из различных веб-ресурсов

Как было отмечено в предыдущем разделе, существуют инструменты для веб-скрейпинга, в том числе бесплатные, которые можно использовать для извлечения данных из веб-ресурсов «на лету». Среди инструментов, которые извлекают структурированные данные, есть такие, которые не требуют программирования на серверной стороне. Однако для включения извлеченных данных в контент веб-страницы при использовании этих инструментов

необходимо писать программный код на клиентской стороне, в частности делать асинхронные запросы к REST API. Это требует хорошего знания JavaScript, поэтому имеющиеся инструменты могут использовать только профессиональные фронтенд-разработчики.

Предлагаемый нами инструмент не требует программирования ни на серверной, ни на клиентской стороне. Автору страницы достаточно соответствующим образом подготовить разметку HTML-страницы (вставив нужные атрибуты) и подключить стартовые скрипты (вставив в страницу фрагмент заранее подготовленного кода) – вся остальная работа выполняется автоматически в процессе загрузки страницы. StructScraper могут использовать как профессиональные разработчики, так и непрофессиональные авторы, создающие собственные страницы, поскольку для его использования необходимо знать только основы html и css. Он может быть полезен блогерам, авторам страниц с кулинарными рецептами, научным работникам для создания персональных страниц и списков публикаций, его можно использовать для сравнения цен на товары, рейтингов сайтов и т.п.

При помощи инструмента StructScraper можно извлекать и включать в контент веб-страницы метаданные веб-ресурсов, заключенные в микроразметках тегами <meta> и в текстах отдельных тегов, в свойствах документов Word и pdf-файлов, а также данные из микроразметок формата «микроданные» [16] и JSON-LD [18]. В разделе «Структурированные данные в Сети. Микроразметки» была приведена статистика, из которой видно, что, хотя на данный момент число сайтов, имеющих микроразметку, не очень велико, но оно растет довольно быстро, и, по мнению авторов, рост продолжится и дальше. По мере увеличения числа сайтов с микроразметкой увеличатся и возможности использования предлагаемого инструмента.

StructScraper включает в себя серверную и клиентскую части. Серверная часть представляет собой REST API для извлечения данных. Ее можно использовать как вместе с предлагаемой клиентской частью, так и самостоятельно. Клиентская часть включает jQuery-плагины, вызов которых при загрузке веб-страницы выполняет работу по добавлению данных в контент. Ниже, в разделе, посвященном клиентской части инструмента, будет описано, каким образом должна быть размечена веб-страница, чтобы запрашиваемый контент автоматически добавлялся при ее загрузке.

Как уже отмечалось выше, идеальный вариант для представления кросс-доменных данных на веб-странице – когда кросс-доменный ресурс предоставляет программный интерфейс к своим данным и поддерживает кросс-доменные запросы. Однако данные можно извлекать и показывать на веб-страницах даже в тех случаях, когда программный интерфейс к данным отсутствует и кросс-доменные запросы к ресурсу не разрешены, если использовать метод промежуточного сервера, описанный в разделе «О способах отображения на веб-странице данных из кросс-доменных ресурсов». Промежуточным сервером, как правило, является веб-служба, запросы к

которой осуществляются через REST API. Запрос из веб-страницы направляется к веб-службе, веб-служба извлекает данные из стороннего веб-ресурса и передает результат в веб-страницу, откуда исходил запрос. Если веб-служба поддерживает CORS, то такой подход делает доступными данные кросс-доменных ресурсов из веб-страницы любого домена, а не только того, в котором находится сама веб-служба (рис.2).

Рис. 2. Запрос к веб-ресурсу через прокси-API

В общем случае указанный подход с использованием промежуточного REST API требует программирования извлечения данных на серверной стороне. Библиотеки и инструменты для веб-скрейпинга, описанные в предыдущем разделе, могут помочь решить эту задачу. REST API для извлечения данных можно создать, пользуясь, например, инструментами Scrapy [28] и ScrapyRT [29]. При помощи Scrapy на языке Python программируется алгоритм извлечения данных, при помощи ScrapyRT создается REST API.

Для извлечения структурированных данных, содержащихся в микроразметках, можно воспользоваться REST API, входящей в состав Apache Any23 [30] или API Валидатора микроразметки от компании Яндекс [32]. В этом случае программирования на серверной стороне не требуется. Однако для рассматриваемой задачи представления на веб-странице данных из нескольких веб-ресурсов эти REST API, на наш взгляд, не очень удобны. Главное неудобство в том, что в запросе к этим сервисам можно указать только один адрес URL. Если бы была возможность передавать в запросе несколько адресов и получать результат в одном ответе, это позволило бы уменьшить общее время ответа за счет сокращения времени на установку отдельных соединений к REST API. На обращения к API Валидатора микроразметки от Яндекс установлено ограничение не более 10 запросов в секунду с одного ключа. В случае если адресов больше 10, придется ждать секунду, прежде чем делать запросы для остальных адресов. Еще один недостаток этих сервисов для рассматриваемой задачи в том, что возвращаемый ими результат содержит все структурированные данные, имеющиеся в микроразметках ресурса. Для включения в веб-страницу, как правило, все эти данные полностью не нужны. Если бы была возможность извлекать данные только определенного типа, это сократило бы трафик и упростило бы программирование на клиентской стороне.

Среди имеющихся в свободном доступе сервисов мы не нашли таких, которые извлекали бы структурированные данные из документов формата pdf и Word. На наш взгляд сервис с таким функционалом может иметь применение, поскольку многие документы, размещенные в Сети, имеют такие форматы. Такой формат имеет подавляющее большинство научных публикаций, в этих форматах часто размещают разного рода документацию. Автоматическое извлечение авторов, названий, дат создания и модификации, кратких описаний и т.п. из свойств документов, на наш взгляд, было бы полезным.

Отсутствие достаточно удобных серверных средств для решения поставленной задачи представления на веб-странице данных из различных кросс-доменных ресурсов побудило нас создать свой REST API, который можно было бы использовать в качестве бэкэнда.

Код инструмента StructScraper размещен на сайте веб-сервиса GitHub: <https://github.com/RimmaSkorn/struct-scraper>.

Серверная часть StructScraper: REST API на платформе .NET для извлечения структурированных данных из различных веб-ресурсов

REST API StructScraper реализован на языке C# при помощи технологии Microsoft ASP.NET Web API [33]. Одна из причин, по которой был выбран язык C#, заключается в том, что в C# имеется встроенная поддержка асинхронного программирования. Для извлечения данных из html-страниц использовалась библиотека Html Agility Pack (<https://html-agility-pack.net/>), для извлечения данных из pdf-файлов использовалась C#-версия библиотеки iText (<https://itextpdf.com/>), для извлечения данных из файлов формата docx использовалась разработанная Microsoft библиотека Open XML SDK (<https://www.microsoft.com/en-us/download/details.aspx?id=30425/>), для извлечения данных из файлов формата doc использовалась 64-разрядная версия библиотеки Dsofile.dll (<https://support.microsoft.com/en-us/help/224351/the-dsofiledll-files-lets-you-edit-office-document-properties-when-yo/>). Для извлечения микроданных мы создали свой парсер, взяв за основу предварительную версию парсера Chapleau.MicrodataParser (<https://archive.codeplex.com/?p=microdata>) и доработав ее.

Веб-API имеет три контроллера:

- контроллер Metadata – для извлечения отдельных метаданных из конкретных тегов, тегов <meta> и свойств документов формата Word и pdf;
- контроллер Microdata – для извлечения микроданных определенного типа;

- контроллер JsonLd – для извлечения данных формата JSON-LD определенного типа.

Каждый из контроллеров имеет по два POST-метода: `single-uri` (для одного URL) и `multi-uri` (для нескольких URL). Таким образом, возможны шесть видов запросов к REST API. Все они осуществляются методом POST. Например, для извлечения метаданных из нескольких веб-ресурсов выполняется запрос POST `<адрес сервиса>/api/metadata/multi-uri`, для извлечения микроданных – запрос `<адрес сервиса>POST /api/microdata/multi-uri`.

Параметры запроса передаются в формате JSON. Они содержат адреса веб-ресурсов, из которых необходимо извлечь данные, и сведения о том, какие именно данные должны быть извлечены. Для метаданных, извлекаемых из тегов и свойств документов, передаются названия, для микроданных и разметки JSON-LD передается тип из словаря Schema.org.

```
{
  "Url": "http://www.chu-rouen.fr/documed/eahilsantander.html",
  "MetaNames": [ "<Title>", "DC.Title", "Title", "og:title",
                  "DC.modified", "DC.Date.Modified", "dc:modified",
                  "DCTERMS.Modified", "DateLastSaved", "ModDate",
                  "DCTERMS.Created", "Created", "DateCreated",
                  "CreationDate", "DC.Description", "description",
                  "og:description", "Subject" ]
}
```

Листинг 7. Данные, передаваемые при вызове metadata/single-uri

В листинге 7 приведен пример данных в JSON-формате, передаваемых для извлечения метаданных. Названия, заключенные в угловые скобки, означают, что данные должны быть извлечены из текста в теге с таким названием. Названия без угловых скобок относятся к тегам `<meta>` html-документов или, если ресурс представляет собой документ формата Word или PDF, к свойствам этих документов.

В листинге 8 приведен пример данных в JSON-формате, передаваемых для извлечения микроданных.

```
[
  {"Url": "http://keldysh.ru/", "SchemaType": "Organization"},
  {"Url": "https://www.klasko.ru/", "SchemaType": "Organization"},
  {"Url": "http://www.izmailovo.ru/", "SchemaType": "Organization"},
  {"Url": "https://maliandao-tea.ru/", "SchemaType": "Organization"},
  {"Url": "https://incab.ru/", "SchemaType": "Organization"},
  {"Url": "https://www.newreg.ru/", "SchemaType": "Organization"},
  {"Url": "https://materiamedica.ru/", "SchemaType": "Organization"}
]
```

Листинг 8. Данные, передаваемые при вызове microdata/multi-uri

Преимущества предлагаемого нами REST API по сравнению с имеющимися в свободном доступе REST API для извлечения структурированных данных в том, что

- структурированные данные извлекаются не только из html-документов, но и из свойств документов формата Word и pdf;
- из html-документов извлекаются не все структурированные данные, а только те, которые указаны в запросе;
- в одном запросе можно задать как один, так и несколько URL.

Обработка нескольких URL на сервере происходит асинхронно (рис.3), время ответа равно максимальному времени ответа от одного ресурса. Поэтому время ответа на клиентский запрос не больше, чем если бы запросы с клиентской стороны для каждого URL производились отдельно, а за счет сокращения времени на установку отдельных соединений к REST API оно становится меньше, причем разница тем больше, чем больше запрашиваемых ресурсов.

Рис. 3. Запрос к нескольким веб-ресурсам через прокси-API

REST API StructScraper допускает CORS, запросы к нему могут производиться из клиентских частей веб-приложений любых доменов. На данный момент REST API находится в режиме тестирования. Обращения к нему производятся по адресу <http://195.209.147.157:81/>.

Клиентская часть StructScraper: html-разметка и jQuery-плагины для загрузки структурированных данных из различных веб-ресурсов

Работу по загрузке данных на веб-страницу осуществляет JavaScript-код, оформленный в виде плагинов jQuery. Используемые в инструменте плагины размещены в папке plugins репозитория RimmaSkorn/struct-scraper на сайте веб-сервиса GitHub (<https://github.com/RimmaSkorn/struct-scraper>). Плагин stampRefMeta, определенный в файле stamp-ref-meta.js, осуществляет загрузку метаданных, плагин stampRefMicro, определенный в файле stamp-micro.js – загрузку «микроданных», а плагин stampRefJsonld, определенный в файле stamp-jsonld.js, – загрузку данных, содержащихся в разметке формата JSON-LD.

Для работы плагина необходимо подключить библиотеку jQuery ((<https://jquery.com/>)), а также добавить ссылку на файл, в котором содержится плагин. Вызов плагина можно добавить в конец тела `<body>` или в jQuery-функцию `$(document).ready()`. Каждый плагин имеет параметр `options` объектного типа. В объекте `options` содержатся веб-адрес запроса REST API, и дополнительные данные, используемые при формировании значений параметров этого запроса. Схематично вызов плагина представлен в листинге 9.

```
<head>
...
<script src=plugin-file></script>
<script>
    $(document).ready(function () {
        $(document).<plugin>(options);
    });
</script>
...
<head>
```

Листинг 9. Вызов плагина

Для того чтобы плагин загрузил данные на веб-страницу, в нее должна быть добавлена специальная разметка, по которой код плагина определяет, из каких веб-ресурсов должны быть загружены данные и какие именно данные необходимо загрузить. Эта разметка использует атрибуты `class`.

Для того чтобы плагин `stampRefMeta` добавил к гиперссылке метаданные, атрибут `class` html-элемента, содержащего гиперссылку, должен включать класс `«rsrc»`, у самой гиперссылки должен быть класс `«rsrc-ref»`, а html-элементы, в которые будут записаны метаданные, должны иметь класс `«rsrc-<name>»`, где `name` — название для метаданных. С названием метаданных связывается тип данных и список названий, используемых для определения, из каких именно тегов или свойств должны извлекаться данные. Это соответствие записывается в объект со свойствами `name`, `type`, `metas`. Массив таких объектов, содержащий все названия метаданных, передается в свойстве `class_metas` параметра `options` при вызове плагина `stampRefMeta`. Пример такой разметки приведен в листинге 10 в разделе «Сниппеты гиперссылок»

Для того чтобы плагин `stampRefMicro` добавил к гиперссылке микроданные, атрибут `class` гиперссылки должен содержать значение `«micro-url»`, а атрибут `class` обрамляющего html-элемента должен содержать значение `«import-micro»`. В вызов плагина передаются адрес запроса REST API и тип данных из словаря `Schema.org`. Примеры разметки для загрузки микроданных приведены в листинге 12 в разделе «Контактные данные организаций» и в листинге 17 в разделе «Цены на один и тот же товар в разных интернет-магазинах».

Для того чтобы плагин `stampRefJsonld` добавил к гиперссылке данные формата JSON-LD, атрибут `class` гиперссылки должен содержать значение

«jsonld-url», а атрибут class обрамляющего html-элемента должен содержать значение «import-jsonld». В вызов плагина передаются адрес запроса REST API и тип данных из словаря Schema.org. Пример разметки для загрузки данных формата JSON-LD приведен в листинге 15 в разделе «Список статей, размещенных в интернете».

Примеры использования инструмента StructScraper

Сниппеты гиперссылок

Рис. 4. Получение сниппетов гиперссылок

Метаданные, содержащиеся в html-тегах, предназначены в первую очередь для поисковиков. Однако некоторые метаданные, на наш взгляд, могут

оказаться полезными и конечному пользователю. Например, дата создания и дата последнего редактирования, расположенные рядом с гиперссылкой на веб-ресурс, могут помочь пользователю решить, стоит ли переходить по гиперссылке. В некоторых областях знаний, таких как, например, программирование, информация устаревает очень быстро, и обращаться к устаревшему ресурсу не имеет смысла. Может быть полезно и краткое описание ресурса рядом с гиперссылкой.

В примере, расположенном по адресу http://alive.keldysh.ru/Test_Pages/test-meta.html, к каждой гиперссылке добавляются «на лету» заголовок, дата последнего изменения (или создания, если документ не менялся), а также описание ресурса, на который она ссылается. Результат похож на результат запроса, выдаваемого поисковой системой (Рис. 4). Загрузку и расстановку данных осуществляет плагин stampRefMeta, определенный в файле stamp-ref-meta.js.

В листинге 10 приведен пример разметки для одной из гиперссылок.

```
<li class="rsrc">
  <a href="http://www.chu-rouen.fr/documed/eahilsantander.html">
    <h3 class="rsrc-title"></h3>
  </a>
  <a class="rsrc-ref"
    href="http://www.chu-rouen.fr/documed/eahilsantander.html"
    http://www.chu-rouen.fr/documed/eahilsantander.html
  </a>
  <br />
  <span class="rsrc-modified xxx"></span>
  <span class="rsrc-description yyy "></span>
</li>
```

Листинг 10. Разметка для «обогащения» гиперссылки метаданными ресурса

Заголовок добавляется в элемент с классом rsrc-title, дата модификации — в элемент с классом rsrc-modified, описание ресурса — в элемент с классом rsrc-description.

С названием метаданных связывается тип данных и список названий, используемых для определения, из каких именно тегов или свойств должны извлекаться данные. Это соответствие записывается в объект со свойствами name, type, metas. Массив таких объектов, содержащий все названия метаданных, передается в свойстве class_metas параметра options при вызове плагина. Листинг 11 содержит вызов плагина для описываемого примера.

Используя описанную разметку классами и соответствия, заданные в свойстве class_metas, плагин для каждой гиперссылки составляет список запрашиваемых тегов или свойств. Из данных, полученных для всех гиперссылок, формируется массив JSON-объектов и отправляется веб-службе. Получив ответ, плагин расставляет полученные данные в соответствии с разметкой классами. При этом, если получено несколько значений для метаданных, то берется первое по порядку в списке metas.

```
$(document).stampRefMeta({
  apiMultiUri: 'http://195.209.147.157:81/api/metadata/multi-uri',
  class metas: [
    {
      name: "title",
      type: "string",
      metas: ["<Title>", "DC.Title", "Title", "og:title"]
    },
    {
      name: "modified",
      type: "date",
      metas: ["DC.modified", "DC.Date.Modified", "dc:modified",
        "DCTERMS.Modified", "DateLastSaved", "ModDate",
        "DCTERMS.Created", "Created", "DateCreated",
        "CreationDate"]
    },
    {
      name: "description",
      type: "string",
      metas: ["DC.Description", "description", "og:description",
        "Subject"]
    }
  ]
});
```

Листинг 11. Вызов плагина для загрузки метаданных гиперссылок

Для форматирования даты плагин использует библиотеку Moment.js. Ссылка на нее должна быть включена в раздел <head> до вызова плагина. Локализация при форматировании даты определяется по атрибуту lang html-элемента, включающего эту дату. В случае отсутствия элемента с таким атрибутом в качестве локализации используется локализация браузера.

Расширенные сниппеты

Как уже отмечалось выше, микроразметки используются поисковиками для представления информации о найденных ресурсах в виде так называемых «расширенных сниппетов» («rich snippets») [34, 35]. В описываемых ниже примерах мы покажем, как по имеющимся на веб-странице гиперссылкам получить «на лету» структурированные данные из микроразметок и представить их в виде, похожем на расширенные сниппеты поисковиков.

Контактные данные организаций (микроданные)

В примере http://alive.keldysh.ru/Test_Pages/test-org.html при загрузке веб-страницы заполняется контактными данными список организаций (Рис. 5).

Исходный html-код веб-страницы содержит только гиперссылки и специальную разметку классами, как показано в листинге 12. Атрибут class гиперссылки на сайт организации содержит значение «micro-url», а атрибут class обрамляющего html-элемента содержит значение «import-micro».

Рис. 5. Получение контактных данных организаций

```
<li class="import-micro">
  <a class="micro-url"
    href="http://keldysh.ru/">ИПМ им.М.В.Келдыша РАН</a>
</li>
```

Листинг 12. Разметка для «обогащения» гиперссылки «микроданными»

Работу по добавлению данных осуществляет плагин `stampRefMicro`, определенный в файле `stamp-micro.js`. В его вызов (листинг 13) передаются адрес запроса REST API и тип `Schema.org`, в данном случае – "Organization".

```
<script type="text/javascript">
  $(document.ready(function () {
    $(document).stampRefMicro({
      apiMultiUri: 'http://195.209.147.157:81/api/microdata/multi-uri',
      import_schema: "Organization"
    });
  });
</script>
```

Листинг 13. Вызов плагина для добавления «микроданных»

Плагин добавляет все свойства объектов заданного типа, извлеченные с сайтов, вставляя при этом свою разметку «микроданными». Результирующий html-код выглядит как в листинге 14.

Какие из добавленных данных должны быть видны пользователю и в каком виде они должны быть представлены, определяется в css-файле `micro-org.css`. Для этого в нем используются значения атрибута `itemprop`.

В примере с ценами на один и тот же товар, который будет описан ниже, используется этот же плагин, только для другого типа Schema.org – типа `Product`.

```
<li class="import-micro" itemscope=""
      itemtype="http://schema.org/Organization">
  <a class="micro-url" href="http://keldysh.ru/" itemprop="name">
    ИПМ им. М.В.Келдыша РАН
  </a>
  <span itemprop="address" itemscope=""
        itemtype="http://schema.org/PostalAddress">
    <span itemprop="postalCode">125047</span>
    <span itemprop="addressLocality">Москва</span>
    <span itemprop="streetAddress">Миусская пл., д.4</span></span>
    <span itemprop="telephone">+7 499 978-13-14</span>
    <span itemprop="faxNumber">+7 499 972-07-37</span>
  </span>
</li>
```

Листинг 14. Результат обогащения «микроданными»

Список статей, размещенных в интернете (JSON-LD)

Пример http://alive.keldysh.ru/Test_Pages/jsonld-article.html демонстрирует получение списка статей по имеющимся гиперссылкам на статьи (Рис. 6). Данные «на лету» извлекаются из разметки JSON-LD. Этот пример аналогичен описанному выше примеру с контактными данными организаций. Отличия в названиях классов исходной разметки, используемом плагине и передаваемом ему параметре.

Исходная разметка для одной гиперссылки выглядит как в листинге 15. Гиперссылка имеет класс `jsonld-url`, а обрамляющий html-элемент – класс `import-jsonld`.

```
<li class="import-jsonld">
  <a class="jsonld-url"
    href="http://www.imedpub.com/articles/deployment-of-softcomputing-for-system-computing-and-informatics.php?aid=22627">Article</a>
</li>
```

Листинг 15. Разметка для «обогащения» гиперссылки данными JSON-LD

Добавление данных осуществляет плагин `stampRefJsonld`, определенный в файле `stamp-jsonld.js`. Его вызов выглядит следующим образом:

```
<script type="text/javascript">
  $(document.ready(function () {
    $(document).stampRefJsonld ({
      apiMultiUri: 'http://195.209.147.157:81/api/jsonld/multi-uri',
      import_schema: "Article"
    });
  });
</script>
```

Листинг 16. Вызов плагина для добавления данных из разметки JSON-LD

Рис. 6. Получение данных о статьях

В результирующий html-код вставляется разметка «микроданными», а не JSON-LD. Разметка «микроданными» удобнее тем, что может быть использована в css-селекторах. В файле jsonld-article.css по ней определяется, какие данные будут видны пользователю и в каком виде они будут представлены.

В этом примере так же, как и в описанном выше примере с метаданными, для форматирования даты используется библиотека Moment.js.

Дата последней редакции в библиографической ссылке на живую публикацию

Серверная часть инструмента SctructScraper используется в ссылках на живые публикации [36, 37]. Живая публикация – это размещенная в свободном доступе в интернете научная работа, которая постоянно совершенствуется и развивается ее автором. Дата последней редакции является существенным атрибутом такой публикации, показывающим, насколько актуальна работа. Добавление ее в ссылку на публикацию дает потенциальному читателю дополнительную информацию для принятия решения о прочтении работы.

Рис. 7. Результат запроса статей автора «Горбунов-Посадов» с датами последней редакции

Пример можно увидеть в online-библиотеке ИПМ им. М. В. Келдыша на сайте library.keldysh.ru, сделав запрос статей автора «Горбунов-Посадов» (Рис.

7), или на странице <http://keldysh.ru/book/keldysh-books.shtml> в ссылке на работу Семенова И. Л. (Рис.).

Рис. 8. Дата последней редакции в ссылке на книгу

Дата последней редакции живой публикации «на лету» извлекается из тега `<meta>` (для html-публикации) или пользовательского свойства (для публикаций в формате pdf, doc, docx) с именем «DC.modified» и добавляется в ссылки. Так же, как и в примере со сниппетами гиперссылок, извлечение данных из публикаций осуществляет контроллер Metadata веб-API.

Цены на один и тот же товар в разных интернет-магазинах

Смартфон Samsung Galaxy A8		
Интернет-магазин	Модель	Цена
Эльдорадо		
М-Видео		
Ситилинк		
Мегафон		
Самсунг		
lfe-mobile		
Плеер.ру		
ТехноСити		
НИКС		
Технопарк		
Самсунг (2)		
OLDI		

Смартфон Samsung Galaxy A8		
Интернет-магазин	Модель	Цена
Эльдорадо	Смартфон SAMSUNG Galaxy A8+ 2018 Black (SM-A730FZKDSER)	24990
М-Видео	Смартфон Samsung Galaxy A8+ (2018) Black (SM-A730F)	24990
Ситилинк	Смартфон SAMSUNG Galaxy A8+ (2018) 32Gb, SM-A730F, черный	24990
Мегафон	Смартфон Samsung Galaxy A8+ Black	24 990
Самсунг	Galaxy A8+ (2018) 32 Гб золотой	24990
lfe-mobile	Смартфон Samsung Galaxy A8 (2018) SM-A530F/DS 32Gb (Цвет: Black)	18 980
Плеер.ру	Сотовый телефон Samsung SM-A730F Galaxy A8 Plus 2018 Black	19298
ТехноСити	Смартфон Samsung SM-A530F Galaxy A8 2018 Dual Sim LTE, черный (SM-A530FZKDSER)	22999.00
НИКС	Galaxy A8 (2018) SM-A530F Black	21827
Технопарк	Смартфон Samsung Galaxy A8+ SM-A730F black	24890.00
Самсунг (2)	Смартфон Samsung Galaxy A8+ 32 Гб, Черный (SM-A730FZKDSER)	24990
OLDI	Смартфон Samsung Galaxy A8 (2018) SM-A530F (SM-A530FZKDSER)	19990

Рис.9. Получение цен на товар из разных интернет-магазинов

При помощи инструмента StructScraper можно сравнивать цены на один и тот же товар в разных интернет-магазинах, если на сайтах этих интернет-магазинов имеется микроразметка для отдельных товаров. В примере http://alive.keldysh.ru/Test_Pages/test-product.html показываються цены на один и тот же смартфон. Данные «на лету» извлекаются из разметки «микроданными» и отображаются на веб-странице (Рис. 9).

В этом примере загрузку и расстановку данных осуществляет jQuery-плагин stampRefMicro, тот же, что и в описанном выше примере с контактными данными организаций. Отличие в запрашиваемом типе Schema.org: в данном примере используется тип Product.

Исходная разметка для одной гиперссылки в этом примере выглядит следующим образом:

```
<tr class="import-micro">
  <td>Эльдорадо</td>
  <td>
    <a class="micro-url"
      href="https://www.eldorado.ru/cat/detail/71368072/">
    </a>
  </td>
</tr>
```

Листинг 17. Разметка для «обогащения» гиперссылки «микроданными»

Внешний вид страницы определяет файл micro-product.css. В его css-селекторах используется добавляемая при загрузке страницы разметка «микроданными».

Заключение

Модуль метаданных серверной части описанного инструмента StructScraper около двух лет работает в производственном режиме и используется на сайте ИПМ им. М.В. Келдыша для добавления в ссылки на живые публикации даты последней редакции. Модули микроданных и JSON-LD находятся в стадии тестирования. При вводе их в промышленную эксплуатацию планируется для снижения нагрузки на сервер ввести обращения к REST API по ключу и ограничить число запросов в единицу времени с одного ключа.

В дальнейшем мы планируем также расширить возможности инструмента:

- добавить обработку страниц, использующих gzip-сжатие;
- добавить загрузку изображений и гиперссылок;
- добавить возможность задавать список типов из словаря Schema.org;
- добавить возможность задавать список атрибутов конкретного типа из Schema.org;
- добавить возможность извлекать данные из разметки RDFa Lite;
- добавить общий плагин для всех типов микроразметок;

- добавить обработку динамических страниц.

Возможность задания списка типов позволит, в частности, совместить в одном запросе получение данных как общего типа, так и производных от него типов, например, Organization и LocalBusiness, Article и ScholarlyArticle. Возможность задания списка атрибутов сократит объем загружаемых данных и отменит необходимость использовать CSS для того, чтобы не выводить на экран ненужные данные.

Планируется также добавить другие варианты разметок клиентской части, в частности, использовать разметку микроданными или RDFa-Lite, чтобы данные добавлялись в соответствии с уже готовой микроразметкой. В планах также расширение клиентской части инструмента таблицами Microsoft Excel.

* * *

Авторы выражают признательность Е.С. Батаковой за работу по поиску в интернете страниц с микроразметкой, гиперссылки на которые были использованы в примерах.

Литература

1. Fetch Standard. URL: <https://fetch.spec.whatwg.org/>
2. Web data extraction, applications and techniques: A survey / E. Ferrara, P. De Meo, G. Fiumara, R. Baumgartner, Knowledgebased systems, 2014, vol. 70, pp. 301-323.
3. Варламов М. И., Турдаков Д. Ю. Обзор методов извлечения информации из веб-ресурсов // Программирование. — 2016. — № 5. — С. 30–48.
4. Аладин Д.В. ПРИНЦИПЫ РЕШЕНИЯ ЗАДАЧ ПО ИЗВЛЕЧЕНИЮ ДАННЫХ ИЗ ВЕБ-РЕСУРСОВ С ПОМОЩЬЮ WEB SCRAPERS // ТЕОРИЯ. ПРАКТИКА. ИННОВАЦИИ. — 2017. — № 11. URL: <http://www.tpinauka.ru/2017/11/Aladin.pdf>.
5. Аладин Д.В. WEB SCRAPER В ETL-процессах // Молодежный научно-технический вестник. — 2017. — № 5. URL: <http://sntbul.bmstu.ru/doc/859321.html>.
6. Same Origin Policy. URL: https://www.w3.org/Security/wiki/Same_Origin_Policy/.
7. Same-origin policy - Web security | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy.
8. JSONP. URL: https://www.w3schools.com/js/js_json_jsonp.asp.
9. Cross-Origin Resource Sharing. URL: <https://www.w3.org/TR/cors/>.
10. CORS protocol. URL: <https://fetch.spec.whatwg.org/#http-cors-protocol>
11. Monsur Hossain. CORS in Action. Creating and consuming cross-origin APIs. /Foreword by Eric Bidelman. — Shelter Island: Manning, cop. 2014. . — ISBN 978-1-617-29182-1.

12. LINKED DATA. URL: <https://www.w3.org/standards/semanticweb/data>
13. DBpedia. URL: <https://wiki.dbpedia.org/>
14. RDFa. URL: <https://rdfa.info/>.
15. Microformats. URL: <http://microformats.org/>.
16. Microdata. URL: <https://html.spec.whatwg.org/multipage/microdata.html>.
17. RDFa Lite 1.1 - Second Edition. URL: <https://www.w3.org/TR/rdfa-lite/>
18. JSON-LD - JSON for Linking Data. URL: <https://json-ld.org/>
19. JSON-LD 1.1. URL: <https://w3c.github.io/json-ld-syntax/>.
20. Dublin Core Metadata Initiative. URL: <http://dublincore.org/>.
21. Open Graph protocol. URL: <http://ogp.me/>.
22. Schema.org. URL: <https://schema.org/>.
23. О структурированных данных. URL:
<https://developers.google.com/search/docs/guides/intro-structured-data/>.
24. Usage Statistics of Structured Data Formats for Websites. URL:
https://w3techs.com/technologies/overview/structured_data/all.
25. Historical yearly trends in the usage of structured data formats for websites. URL:
https://w3techs.com/technologies/history_overview/structured_data/all/y/.
26. Software for Web Scraping. URL: <http://scraping.pro/software-for-web-scraping/>.
27. Best Web Scraping Tools 2018. URL:
<http://www.garethjames.net/a-guide-to-Web-scraping-tools/>.
28. Scrapy. URL: <https://scrapy.org/>.
29. Scrapyrt (Scrapy realtime). URL: <https://github.com/scrapinghub/scrapyrt>.
30. Apache Any23. URL: <https://any23.apache.org/>.
31. Import.io. URL: <https://www.import.io/>.
32. API Валидатора микроразметки — Технологии Яндекса. URL:
<https://tech.yandex.ru/validator/>.
33. ASP.NET Web API. URL: <https://www.asp.net/web-api/>.
34. Разметка контента. URL:
<https://developers.google.com/search/docs/guides/mark-up-content/>.
35. Schema.org - Вебмастер. Помощь. URL:
<https://yandex.ru/support/webmaster/schema-org/what-is-schema-org.html/>.
36. Горбунов-Посадов М.М. Живая публикация // Открытые системы — 2011. — № 4. — С. 51-52. URL: <http://keldysh.ru/gorbunov/live.htm>
37. Горбунов-Посадов М.М., Скорнякова Р.Ю. Обновляемая дата последней редакции в ссылке на живую публикацию // Препринты ИПИМ им. М.В. Келдыша. 2017. № 82. 14 с. doi:10.20948/prepr-2017-82 URL:
<http://library.keldysh.ru/preprint.asp?id=2017-82>.
38. Angular. URL: <https://angular.io/>.
39. React. URL: <https://reactjs.org/>.
40. WebAssembly. URL: <https://webassembly.org/>.