|  |
| --- |
| EPAM Systems |
| Data formating: YAML, GeoJSON, BSON, Google Protobuf |
| [Document subtitle] |

|  |
| --- |
| Oksana Spolnyk  [Date] |

# Формат JSON

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом.[1]

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается языконезависимым и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

Данные в формате JSON представляют собой:

1. JavaScript-объекты { ... } или
2. Массивы [ ... ] или
3. Значения одного из типов:

* строки в двойных кавычках,
* число,
* логическое значение true/false,
* null.

Почти все языки программирования имеют библиотеки для преобразования объектов в формат JSON. Основные методы для работы с JSON в JavaScript — это:

* JSON.parse — читает объекты из строки в формате JSON. Вызов JSON.parse(str) превратит строку с данными в формате JSON в JavaScript-объект/массив/значение.

var numbers = "[0, 1, 2, 3]";

numbers = JSON.parse(numbers);

alert( numbers[1] ); // 1

* JSON.stringify — превращает объекты в строку в формате JSON, используется, когда нужно из JavaScript передать данные по сети. Метод JSON.stringify(value, replacer, space) преобразует («сериализует») значение в JSON-строку.

var event = {

title: "Конференция",

date: "сегодня"

};

var str = JSON.stringify(event);

alert( str ); // {"title":"Конференция","date":"сегодня"}

# JSONP & JSONPP

JSONP (JSON Padding) или «JSON с подкладкой» является расширением JSON, когда имя функции обратного вызова указывается в качестве входного аргумента.

В основу технологии положен тот факт, что политика безопасности браузера не запрещает использовать тег

<script type="text/javascript" src="…"></script>

для обращения к серверам, отличным от сервера, с которого произошла загрузка страницы.

Без использования технологии JSONP (то есть используя просто JSON кодирование данных) сервер может вернуть только данные. Например, так:

{"paper": "A4", "count": 5}

Однако это только данные, и они не могут влиять на браузер.

Используя технику JSONP, стороннему серверу передается в строке вызова (GET) имя callback функции:

<script type="text/javascript" src="http://example.com/getjson?jsonp=parseResponse"></script>

Здесь параметр jsonp содержит имя callback функции parseResponse.

Теперь посторонний сервер example.com может вернуть следующий код:

parseResponse({"paper": "A4", "count": 5})

Теперь код вызывает javascript-функцию первого домена.

# XML

XML (англ. eXtensible Markup Language — расширяемый язык разметки; произносится [икc-эм-эль]) — рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете.

*Физическая и логическая структуры документа.* С физической точки зрения документ состоит из сущностей (англ. entities), из которых каждая может отсылать на другую сущность. Единственный корневой элемент — документная сущность. Содержание сущностей — символы.

С логической точки зрения документ состоит из комментариев (англ. comments), объявлений (англ. declarations), элементов (англ. elements), ссылок на сущности (англ. character references) и инструкций обработки (англ. processing instructions). Всё это в документе структуризуется разметкой (англ. markup).

Текстовый формат XML позволяет передавать структурированные данные, которые можно вставлять в различные части HTML страницы. XML позволяет разделить оформление и данные. Рассмотрим пример получения XML данных и их обработку в JavaScript.

XML-файл

<?xml version="1.0" encoding="windows-1251"?>

<messages>

<message>

<text>текст</text>

<sender>имя</sender>

<date>дата</date>

</message>

</messages>

HTML-файл (необходима библиотека jQuery)

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<title>Messenger</title>

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

function getMessage(){

$.ajax({

type: "GET",

url: "message.php",

success: function(msg){

update(msg);

}

});

}

function update(msg){

var messagesNode = msg.getElementsByTagName('messages')[0];

var messageNode = messagesNode.getElementsByTagName("message");

if(messageNode)

for(var i=0; i<messageNode.length; i++){

var message = messageNode[i];

var text = message.getElementsByTagName("text")[0].childNodes[0].nodeValue;

var sender = message.getElementsByTagName("sender")[0].childNodes[0].nodeValue;

var date = message.getElementsByTagName("date")[0].childNodes[0].nodeValue;

$('#message').prepend('<tr><td>'+text+'</td><td>'+sender+'</td><td>'+date+'</td></tr>');

}

}

</script>

</head>

<body>

<input type="button" value="Новое сообщение" onClick="getMessage();" />

<table id="message" border="1"></table>

</body>

</html>

# YAML

YAML (рекурсивный акроним YAML Ain't Markup Language — «YAML — Не язык разметки») — «дружественный» формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования.

В трактовке названия отражена история развития: на ранних этапах YAML расшифровывался как Yet Another Markup Language («Ещё один язык разметки») и даже позиционировался как конкурент XML, но позже был переименован с целью[источник?] акцентировать внимание на данных, а не на разметке документов.

*Использование.* YAML в основном используется как формат для файлов конфигурации. Применяется для настройки веб-каркасов Ruby on Rails, Dancer, Symfony, GAE framework, Google App Engine и Dart. Также является основным языком описания классов, ресурсов и манифестов для пакетов приложений OpenStack Murano Project .

В YAML эта конфигурация может быть представлена следующим образом:

bindings:

- ircEvent: PRIVMSG

method: newUri

regexp: '^http://.\*'

- ircEvent: PRIVMSG

method: deleteUri

regexp: '^delete.\*'

- ircEvent: PRIVMSG

method: randomUri

regexp: '^random.\*'

*Сравнение с XML.* Для сравнения, в XML-представлении, данная конфигурация может быть представлена следующим образом:

<bindings>

<binding>

<ircEvent>PRIVMSG</ircEvent>

<method>newUri</method>

<regex>^http://.\*</regex>

</binding>

<binding>

<ircEvent>PRIVMSG</ircEvent>

<method>deleteUri</method>

<regex>^delete.\*</regex>

</binding>

<binding>

<ircEvent>PRIVMSG</ircEvent>

<method>randomUri</method>

<regex>^random.\*</regex>

</binding>

</bindings>

Альтернативный вариант, использующий атрибуты:

<bindings>

<binding ircEvent="PRIVMSG" method="newUri" regex="^http://.\*" />

<binding ircEvent="PRIVMSG" method="deleteUri" regex="^delete.\*" />

<binding ircEvent="PRIVMSG" method="randomUri" regex="^random.\*" />

</bindings>

Говоря об отличиях YAML от XML, также следует отметить, что вложенные XML-элементы могут использоваться для отображения произвольных структур, а YAML более близок к отображению типичных моделей данных из Perl, Python, Java, позволяя описывать свободные сочетания последовательностей, сопоставлений и скалярных типов — то есть ближе к реальным структурам данных языков программирования, и не требует различных соглашений про DOM-отображения структур данных на документы и обратно, как требуется в XML.

# GeoJSON

GeoJSON - формат представления различных структур географических данных. Объект GeoJSON может быть представлен геометрией (geometry), объектом (feature) или коллекцией объектов (feature collection). GeoJSON поддерживает следующие геометрические типы: Point (точка), LineString (ломаная), Polygon (полигон), MultiPoint (мультиточка), MultiLineString (мультиломаная), MultiPolygon (мультиполигон) и GeometryCollection (коллекция геометрий). Объект (feature) в GeoJSON состоит из геометрии и дополнительных свойств, коллекция объектов (feature collection) – из набора объектов (feature).

Завершенная структура данных GeoJSON – это всегда объект (в терминах JSON). В GeoJSON объект состоит из набора пар ключ/значение, также называемых свойствами. Имя каждого свойства – строка. Значение свойства может представлять собой строку, число, объект, массив или один из литералов: «true», «false» и «null». Массив состоит из элементов, где каждый элемент может принимать одно из значений, описанных выше.

Коллекция элементарных объектов GeoJSON:

{ "type": "FeatureCollection",

"features": [

{ "type": "Feature",

"geometry": {"type": "Point", "coordinates": [102.0, 0.5]},

"properties": {"prop0": "value0"}

},

{ "type": "Feature",

"geometry": {

"type": "LineString",

"coordinates": [

[102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]

]

},

"properties": {

"prop0": "value0",

"prop1": 0.0

}

},

{ "type": "Feature",

"geometry": {

"type": "Polygon",

"coordinates": [

[ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],

[100.0, 1.0], [100.0, 0.0] ]

]

},

"properties": {

"prop0": "value0",

"prop1": {"this": "that"}

}

}

]

}

GeoJSON всегда представляет собой единственную сущность: геометрию, элементарный объект или коллекцию элементарных объектов.

* Объект GeoJSON может иметь произвольное количество свойств (пар ключ/значение).
* Объект GeoJSON должен иметь свойство «type». Значение этого свойства – строка, содержащая тип объекта GeoJSON.
* Значение свойства «type» должно принимать одно из следующих значений: «Point», «MultiPoint», «LineString», «MultiLineString», «Polygon», «MultiPolygon», «GeometryCollection», «Feature» или «FeatureCollection». Регистр символов значения поля «type» имеет значение.
* Объект GeoJSON может иметь необязательное свойство «crs», значение которого должно содержать объект системы координат (см. 3. Объекты системы координат).
* Объект GeoJSON может иметь свойство «bbox», значение которого представляет массив координат вершин ограничивающего прямоугольника.

# BSON

BSON (англ. Binary JavaScript Object Notation) — формат электронного обмена цифровыми данными, основанный на JavaScript, бинарная форма представления простых структур данных и ассоциативных массивов (которые в контексте обмена называют объектами или документами). Является надмножеством JSON, включая дополнительно регулярные выражения, двоичные данные и даты.

BSON-документы (объекты) состоят из сортированных списков элементов. Каждый элемент состоит из имени поля, типа и значения. Имена полей — это строки. Типы включают:

1. string — строка,
2. int — целое число,
3. double — число с плавающей запятой двойной точности,
4. DateTime — дата,
5. byte[] — массив байтов (бинарные данные),
6. bool — [булевые](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B8%D0%BF) (True и False),
7. null — «[Null](https://ru.wikipedia.org/wiki/NULL_(%D0%A1%D0%B8))» (специальное значение),
8. BsonObject — BSON-объект,
9. BsonObject[] — массив BSON-объектов.

Не все эти типы доступны в [JSON](https://ru.wikipedia.org/wiki/JSON), в котором, например, нет массива с типом «бинарные данные», но из-за ограничений по длине некоторые действительные значения JSON (такие как очень длинные строки) не являются действительными значениями BSON

В сравнении с JSON, BSON является эффективным как в плане размера хранения данных, так и сканирования. Большие элементы в документе BSON имеют префикс с длиной документа[уточнить] для облегчения сканирования.

BSON во многом аналогичен Protocol Buffers — реализации языково- и платформно-независимого формата для обмена данными, но BSON является более свободным от схемы данных. Тем самым, бо́льшая гибкость BSON уменьшает преимущества в производительности в случае, когда схема определена

# Protocol Buffers

Protocol Buffers — язык описания данных, предложенный Google, как альтернатива XML. Разработчики сообщают, что Protocol Buffers проще, компактнее и быстрее чем XML.

По замыслу разработчиков сначала должна быть описана структура данных, которая затем компилируется в классы. Вместе с классами идёт код их сериализации в компактный формат представления. Чтение и запись данных доступна в высокоуровневых языках программирования — таких как Java, C++ или Python.

В 2010 году бэкенд Twitter перешёл на Protocol Buffers. По заявлению разработчиков Twitter, база в триллион твитов на XML занимала бы десять петабайт вместо одного.

По заявлениям Google, Protocol Buffers по сравнению с XML:

* Проще
* От 3 до 10 раз меньше
* От 20 до 100 раз быстрее
* Более однозначный
* Позволяет создавать классы, которые в дальнейшем легче использовать программно

Protocol Buffers не предназначен для чтения пользователем и представляет собой двоичный формат. Для десериализации данных необходим отдельный .proto-файл, в котором определяется формат сообщения.

*Примеры использования.* Для того, чтобы определить структуру сериализуемых данных, необходимо создать .proto-файл с исходным кодом этой структуры. Ниже приведен пример .proto-файла для описания информации о машине: марка, тип кузова, цвет, год выпуска и информация о предыдущих владельцах.

message Car {

required string model = 1;

enum BodyType {

sedan = 0;

hatchback = 1;

SUV = 2;

}

required BodyType type = 2 [default = sedan];

optional string color = 3;

required int32 year = 4;

message Owner {

required string name = 1;

required string lastName = 2;

required int64 driverLicense = 3;

}

repeated Owner previousOwner = 5;

}

После того, как файл с нужной структурой данных создан, необходимо скомпилировать его компилятором для вашего языка программирования, чтобы сгенерировать класс доступа к этим данным. Этот класс будет содержать простейшие методы доступа ко всем полям типа get/set, а также методы для сериализации и десериализации вашей структуры данных в/из массива байтов.

Примечательно, что можно добавлять к уже созданной структуре данных новые поля без потери совместимости с предыдущей версией: при парсинге старых записей новые поля просто будут игнорироваться.

*Реализация.* На данный момент компанией Google созданы компиляторы для языков программирования C++, Java и Python[3]. Но существует ряд проектов сторонних разработчиков, которые создали компиляторы для следующих языков программирования: Action Script, C, C#, Clojure, Common Lisp, D, Erlang, Go, Haskell, Haxe, JavaScript, Lua, Matlab, Mercury, Objective C, Swift, OCaml, Perl, PHP, Python, Ruby, Scala, Visual Basic, Delphi.

# Источники