



Восьмая лекция
java for web
JSON

JSON (*JavaScript Object Notation*)

JSON - простой, основанный на использовании текста, способ хранить и передавать структурированные данные.

Как и многие другие текстовые форматы, JSON легко читается людьми.

Формат JSON был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования.

Для многих языков существует готовый код для создания и обработки данных в формате JSON.

С помощью простого синтаксиса вы можете легко хранить все, что угодно, начиная от одного числа до строк, массивов и объектов, в простом тексте. Также можно связывать между собой массивы и объекты, создавая сложные структуры данных.

Для каких целей используется JSON?

Более всего, json используется для обмена данными между javascript и серверной стороной .

Другими словами, для технологии ајах. Это очень удобно, когда вы передаете несколько переменных или целые массивы данных.

Как это выглядит на примере:

- пользователь кликает по превьюшке картинки
- javascript обрабатывает это событие и посылает ајах запрос к серверу, передавая ID картинки.
- на сервере, java получает описание картинки, имя картинки, адрес к большому изображению и другую информацию из базы данных. Получив, преобразовывает в JSON формат и отправляет обратно на страницу пользователя.
- Javascript получает ответ в виде JSON, обрабатывает данные, формирует html код и выводит увеличенное изображение с описанием и другой информацией.

Так происходит увеличение картинки, без перезагрузки страницы в браузере. Это очень удобно, когда нам необходимо получить частичные данные, или передать небольшой объем информации на сервер.

Создание строки JSON

Есть несколько основных правил для создания строки JSON:

Строка JSON содержит либо массив значений, либо объект (ассоциативный массив пар имя/значение).

Массив заключается в квадратные скобки ([и]) и содержит разделенный запятой список значений.

Объект заключается в фигурные скобки ({ и }) и содержит разделенный запятой список пар имя/значение.

Пара имя/значение состоит из имени поля, заключенного в двойные кавычки, за которым следует двоеточие (:) и значение поля.

Значение в массиве или объекте может быть:

- Числом (целым или с плавающей точкой)

- Строкой (в двойных кавычках)

- Логическим значением (true или false)

- Другим массивом (заключенным в квадратные скобки)

- Другой объект (заключенный в фигурные скобки)

- Значение null

Чтобы включить двойные кавычки в строку, нужно использовать обратную косую черту: \". Так же, как и во многих языках программирования, можно помещать управляющие символы и шестнадцатеричные коды в строку, предваряя их обратной косой чертой

Простой пример строки JSON

Пример оформления заказа в формате JSON:

```
{
  "orderId": 12345,
  "shopperName": "Ваня Иванов",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productId": 34,
      "productName": "Супер товар",
      "quantity": 1
    },
    {
      "productId": 56,
      "productName": "Чудо товар",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Простой пример строки JSON

Рассмотрим строку подробно:

Мы создаем объект с помощью фигурных скобок (`{ }`).

В объекте есть несколько пар имя/значение:

`"orderId": 12345` - Свойство с именем `"orderId"` и целочисленным значением `12345`

`"shopperName": "Ваня Иванов"` - Свойство с именем `"shopperName"` и строковым значением `"Ваня Иванов"`

`"shopperEmail": "johnsmith@example.com"` - Свойство с именем `"shopperEmail"` и строковым значением `"ivanov@example.com"`

`"contents": [...]` - Свойство с именем `"contents"`, значение которого является массивом

`"orderCompleted": true` - Свойство с именем `"orderCompleted"` и логическим значением `true`

В массиве `"contents"` есть 2 объекта, представляющие отдельные позиции в заказе. Каждый объект содержит 3 свойства: `productId`, `productName`, и `quantity`.

Библиотеки для работы с JSON

[JSON.simple](#) от Yidong Fang. Небольшая и легковесная библиотека для кодирования и декодирования JSON, несмотря на свою простоту, выполняет свою работу на высоком уровне.

[GSON](#) от Google. Библиотека, которая умеет конвертировать Java объекты в JSON и наоборот. Не требует специальным образом аннотировать классы, а также в качестве бонуса имеет полную поддержку Java Generics. Отсутствие необходимости добавления аннотаций упрощает реализацию и даже может быть главным требованием, если вы собираетесь сериализовывать объекты, не имея для них исходного кода.

[Jackson](#) от FasterXML. Набор инструментов для обработки данных, основанный на потоковом JSON-парсере и генераторе. Предназначенная для Java библиотека умеет работать не только с JSON. Имеет самый популярный JSON-парсер (исходя из статистики использования на GitHub).

[JSONP](#) от Oracle. API для работы с JSON, а именно для генерации и разбора потоковых JSON-текстов. Эталонная реализация JSR353 с открытым исходным кодом.

Создание JSON объектов в java с помощью библиотеки GSON

Установим зависимость в pom.xml

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.6.2</version>
</dependency>
```

Создадим простейший класс Cat с открытыми полями.

```
public class Cat {

    public String name; // имя
    public int age; // возраст
    public int color; // цвет

    // Конструктор
    public Cat(){

    }
}
```

```
Cat murzik = new Cat();
murzik.name = "Мурзик";
murzik.age = 9;
murzik.color = Color.BLACK;
```

```
Gson gson = new Gson();
String str = gson.toJson(murzik);
System.out.println(str);
// обратное преобразование
Cat murzik = gson.fromJson(jsonText, Cat.class);
```