

JSON Schema, Jackson

И другие слова на букву J

Евгений Меркурьев

Java 10+

Scrum master 5+

Lead 3+

Tech Trainer 3+

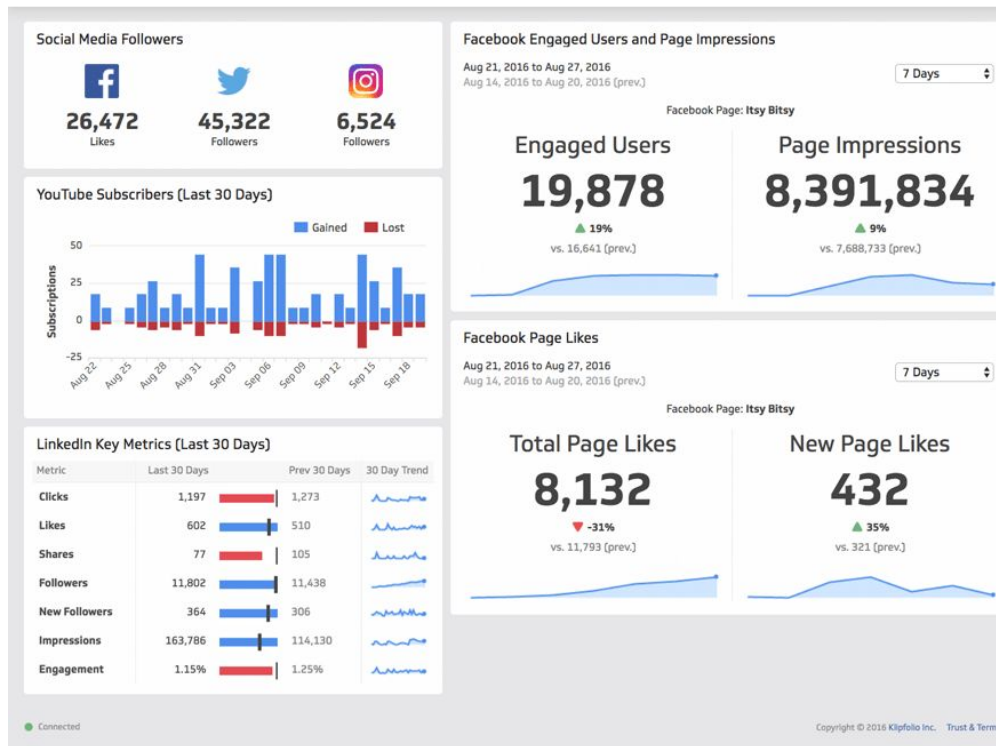


Hash UseCase

Вводная

- API для приложения-dashboard'
- Stateless - ибо кластеризация
- Пользователь может менять L'n'F
- только POST-запросы
- Мало времени
- нужно загрузить FE команду

Social Media - Overview



Как
разрабатывать?

Последовательно!

Ад переделок API
С опережением
Долго

Как
разрабатывать?

Параллельно!

Ад интеграции

Ад фиксов

Плохой дизайн

**Нужна
Единая Точка
Интеграции**

Единая точка интеграции

- Одинаково родная и для BE и для FE
- Готовится до разработки функционала
- Совместное владение командами

JSON

JavaScript Object Notation

- текстовый формат обмена данными
- легко читается людьми
- в JavaScript коде быстро десериализуем

```
{  
  
  "firstName": "Иван",  
  
  "lastName": "Иванов",  
  
  "address": {  
  
    "street": "Невский 11-кв.101",  
  
    "city": "СПб",  
  
  },  
  
  "phoneNumbers": [  
  
    "812 123-1234",  
  
    "916 123-4567"  
  
  ]  
  
}
```

Спец. символы

{ } [] , : “ ”

Проблемы $FE \leftrightarrow VE$

Проблемы FE ↔ BE

- стандарт **JSON** не накладывает многих ограничений на поля, значения, порядок
- на сервере имена полей объектов статичны
- на клиенте имена полей могут динамически меняться
- на сервере проще работать с массивами, листами
- на клиенте проще работать со словарями, мапами
- на сервере camelCase, на клиенте lower-case
- разный стандарт представления даты по умолчанию
- часть данных из конверта серверу может быть вообще не нужна
- отличие доменных объектов в деталях

Tools?

GSON

GSON

8000+ звезд на GitHub

Библиотека от Google

Поддържа Generics

Кастомизируема

```
Gson gson = new Gson();
```

```
String json = gson.toJson(obj);
```

```
BagOfPrimitives obj2 = gson.fromJson(json, BagOfPrimitives.class);
```

Jackson

Jackson

2700+ звезд на GitHub

Модульная

Есть поддержка Guava, Date Time API (Java 8)

Включена как библиотека по умолчанию в Spring Boot

```
ObjectMapper mapper = new ObjectMapper();
```

```
Person person = mapper.readValue(content, Person.class);
```

Виды парсеров JSON'a

Data binding

Передаем парсеру **JSON** и класс, объект который нам нужно получить при десериализации

Допускается использование аннотаций на полях для кастомизации

При сериализации просто указываем объект

Похож на JAXB

Demo

DataBindTest

Data binding

Плюсы

Простота

Требуются только классы
модели

Которые в свою очередь можно
сгенерировать по **JSON**

Минусы

Reflection, может быть
медленно

Весь **JSON** в объект as is,
может потребоваться много
памяти

Tree model

Представляет **JSON** в виде дерева объектов класса Node или JsonElement

Навигация по дереву за программистом

Похож на DOM парсеры в xml

Demo

TreeModelTest

Tree model

Плюсы

быстрее биндинга

относительно простой

Минусы

сложнее Data bind

проблемы с памятью остаются

Streaming API

самый низкоуровневый способ

ручной разбор токенов **JSON**

Похож на SAX-парсеры

Demo

StreamingAPITest

Streaming API

Плюсы

производительность

минимальное потребление
памяти

Минусы

сложность использования

JsonPath

Язык запросов по дереву **JSON**

Похож на XPath

Demo

PathTest

JsonPath

Плюсы

позволяет быстро получить информацию из **JSON** по сложным критериям

Минусы

не очень подходит, когда нужна вся информация из **JSON**

JSON Path

JSON Path

`$.store.book[*].author`

все авторы книг

`$.store..price`

цены всех товаров

`$..book[-1:]`

последняя в списке книга

`$..book[?(@.price<10)]`

найти все книги дешевле 10

```
{ "store": {  
  "book": [{  
    "category": "fiction",  
    "author": "J. R. R. Tolkien",  
    "title": "The Lord of the Rings",  
    "isbn": "0-395-19395-8",  
    "price": 22.99  
  } ],  
  "bicycle": {  
    "color": "red",  
    "price": 19.95  
  }  
}}
```

Demo

JSONPathTest

JSON Pointer

JSON Path

/store/book/1/author

автор первой по порядку книги

/store/bicycle/price

цена велосипеда

```
{ "store": {  
  "book": [{  
    "category": "fiction",  
    "author": "J. R. R. Tolkien",  
    "title": "The Lord of the Rings",  
    "isbn": "0-395-19395-8",  
    "price": 22.99  
  } ],  
  "bicycle": {  
    "color": "red",  
    "price": 19.95  
  }  
}}
```


Demo

JsonPointerTest

jsonquerytool.com

**Нужна
Единая Точка
Интеграции**

JSR-303*

JSR-303

@NotNull

@Min

@Max

@Past

@Future

@Pattern

@Valid

Плюсы

- easy to use
- поддерживается Spring MVC из коробки

Минусы

- ограниченный набор правил валидации
- нужно делать классы под все-все поля
- нетривиально кастомизируется вывод ошибок валидации

Swagger

swagger.io

Плюсы

- easy to use
- набирает популярность
- самодокументируемый (польза для QA)
- генерит все сам (в одном из режимов)

Минусы

- генерит все сам
- как следствие не всякий выверт поддержит
 - сложная иерархия,
 - not-RESTful веб-сервис
- свой стандарт описания API

JSON Schema

json-schema.org

Объекты

```
"address": {  
  "type": "object",  
  "properties": {  
    "streetAddress": {  
      "type": "string"  
    }  
  }  
}
```

Ссылки

```
"address": {  
    "$ref": "#/definitions/address"  
}
```

Ссылки на файлы

```
"address": {  
    "$ref": "definitions.json#/address"  
}
```


Массивы

```
"phoneNumbers": {  
  "type": "array",  
  "items": {  
    "type": "string"  
  }  
}
```


www.jsonschemavalidator.net

JSON Schema Validator

[json-schema.org/
implementations.html](https://json-schema.org/implementations.html)

Demo

SchemaValidationTest

Плюсы

Самый популярный (700+ звезд)

Простое API

Минусы

Два года не развивается

Поддерживает только draft04, а уже есть draft06 schema

jsonschema2pojo

www.jsonschema2pojo.org

Версионирование схемы

Лайфхаки

Demo

SomeEnumTest

Polymorphism

```
@JsonTypeInfo(use = JsonTypeInfo.Id.NAME,      include = JsonTypeInfo.As.PROPERTY, property = "type")
```

```
@JsonSubTypes({
```

```
    @JsonSubTypes.Type(value = ChildA.class, name = "a-class"),
```

```
    @JsonSubTypes.Type(value = ChildB.class, name = "b-class")})
```

```
class BaseClass {
```

```
    public String baseName = "base name";
```

```
}
```

Demo

PolymorphicDeserializationTest

Custom serializer

```
class Pojo {
```

```
    public String foo;
```

```
    @JsonValue
```

```
    public String bar;
```

```
}
```

CustomSerializtoinTest

Custom Deserializer

```
@JsonDeserialize(using = ItemDeserializer.class)
```

```
class Item {
```

```
}
```

```
class ItemDeserializer extends StdDeserializer<Item> {
```

```
    @Override
```

```
    public Item deserialize(JsonParser jp, DeserializationContext ctxt) {
```

```
    }
```

```
}
```

github.com
/zamonier
/dev-labs-2017-json

Q&A

Спасибо за
внимание