# NoSQL : JSON & JSON Schema

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

nicolas.travers@devinci.fr

# Document-oriented Data with JSON

## Javascript Object Notation

{JSON}

SWAPI

The Star Wars API

# JSON: Javascript Object Notation

- XML used for Machine-to-Machine structured communiations
  - ▫Too verbose & uses too much space
  - ▫Dedicated to Web services

Vs

- JSON (JavaScript Object Notation)
  - ▫Designed for communications between Web browsers & Web servers
  - ▫Lightweight, text-oriented, query langage independent
  - ▫Used for some Web services (Google API, Twitter API) or dynamic web (Ajax)

# JSON: Javascript Object Notation

## Key-values & Typing

### Key + Value

- "lastname" : "Travers"
- Keys with **quotations**

### Identifiers

- **" _id "** commonly used
- Overwrite already stored ids
- Can be automaticaly generated
  - Ex MongoDB: "_id" : ObjectId(1234567890)

### Objects/Documents

- Collection of **key/values**
- { "_id" : 1234,
  "lastname" : "Travers",
  "firstname" : "Nicolas", "kind" : 1}

### Scalar

- String, Integer, float, boolean, null…

### Documents

- Objects **{…}**

### List

- Arrays **[ … ]**
- No typing
  "lessons" : **[** "SQL", 1, 4.2, null, "NoSQL" **]**
- Can nest documents
  "doc" : **[** {"test" : 1},
  {"test" : {"nesting" : 1.0}},
  {"key" : "text", "value" : null} **]**

# JSON: Javascript Object Notation

## Complete example

```
{
        "_id" : 1234,
        "lastname" : "Travers", "firstname" : "Nicolas",
        "employers" : [
                { "company" : "ESILV", "starting_date" : "2018-09",
                  "location" : {"street" : "12 avenue Léonard de Vinci","city" : "Paris La Défense", "zip" : 92916}},
                { "company" : "CNAM", "starting_date" : "2007-09", "end_date" : "2018-08",
                  "location" : {"street" : "2 rue Conté","city" : "Paris", "zip" : 75141}},
                { "company" : "UCP", "starting_date" : "2006-09", "end_date" : "2007-08",
                  "location" : {"street" : "2 rue Conté","city" : "Cergy-Pontoise", "zip" : 91000}},
                { "company" : "UVSQ", "starting_date" : "2004-01", "end_date" : "2006-08",
                  "location" : {"street" : "45 avenue des Etats-Unis","city" : "Versailles", "zip" : 78000}}    ],
        "fields" : [ "DB" , "DB optimization", "XML", "NoSQL", "IR" ],
        "hobbies" : ["Star Wars", "BZH"]
}
```

# JSON Schema

- Storing JSON documents needs a schema for strong document typing
- Structure validation
- Automatic checking (availability, inclusion, validity, queries)

- Needs a grammar and typing
  - Objects, lists,
  - Mandatory or optional keys
  - Additional keys

http://json-schema.org/

# JSON Schema: Objects

```
{
        "type" : "object",
         "properties" : {
                "key1" : {"type": … },
                "key2" : {"type": … }
        }
}
```

Object keys

Values' type
"number" (option – "min", "max")
"string" (option - "pattern")
"anyOf"
"not"

Other options : "definitions" + "$ref"

# JSON Schema: Keys

```
{
        "type" : "object",
         "properties" : {
                "key1" : {"type": … },
                "key2" : {"type": … }
        },
        "required" : ["key1"],
        "additionalProperties" : false
}
```

Mandatory keys
Not included = optional

Forbids
Additional keys

Other options : "minProperties", "maxProperties", "patternProperties"

# JSON Schema: Lists

```
"arrayKey" : {
        "type" : "array",
        "items" : [ {"type" : ... } ],
        "minItems" : 1,
        "maxItems" : 10
}
```

Other options
"uniqueItems"
"additionalItems"

# JSON Schema: Example

```
{ "type": "object",
  "properties": {
    "_id": { "type": "integer" },
    "lastname": { "type": "string" },
    "firstname": { "type": "string" },
    "employers": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
            "company": { "type": "string" },
            "starting_date": { "type": "string" },
            "end_date": { "type": "string" } },
…
```

```
…
          "location": {
               "type": "object",
               "properties": {
                   "street": { "type": "string" },
                   "city": { "type": "string" },
                   "zip": { "type": "integer" } },
               "required": [ "street", "city", "zip" ] },
        "required": [ "company", "starting_date", "location" ] } },
  "fields": { "type": "array", "items": { "type": "string" } },
  "hobbies": {
      "type": "array",
      "items": { "type": "string" } } },
  "required": [ "_id", "lastname", "firstname", "employers",
                "fields", "hobbies" ]
}
```

# JSON Schema: Miscellaneous

- Programing language oriented schema: JOI
  - https://github.com/hapijs/joi
- Schema extractor
  - https://www.liquid-technologies.com/online-json-to-schema-converter
  - https://app.quicktype.io/#l=schema
- Schema validator
  - https://www.jsonschemavalidator.net/