JSON

Definició

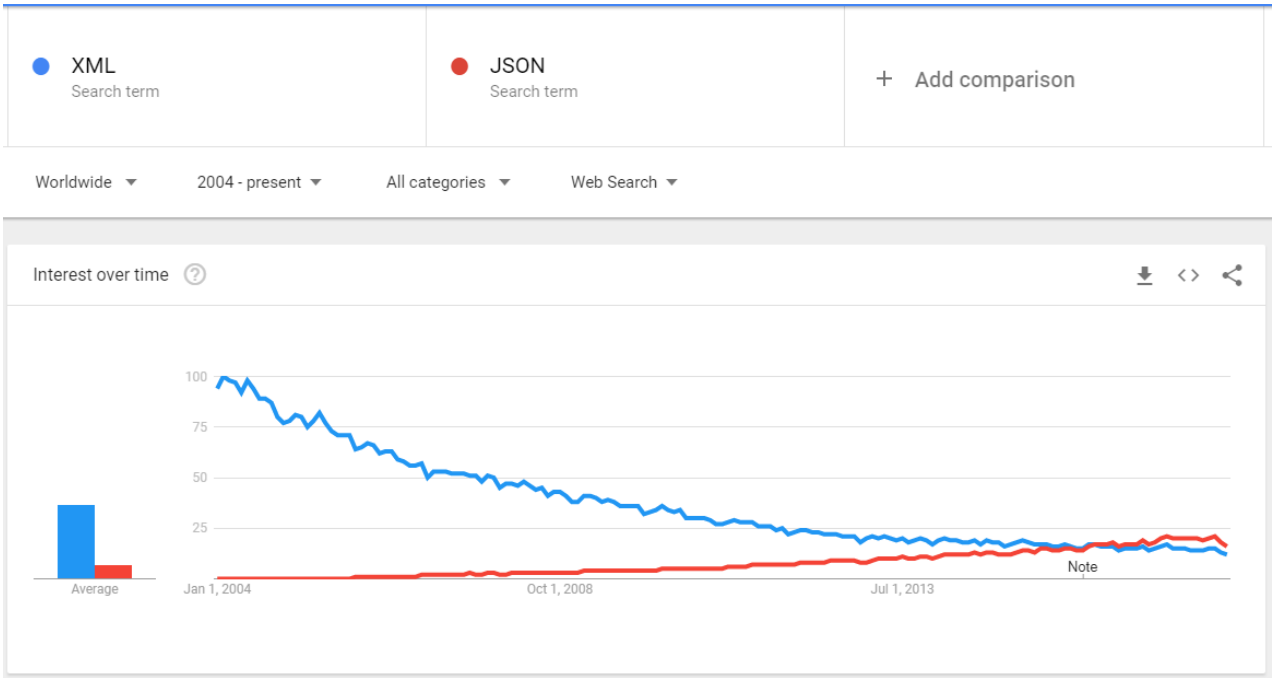
JSON (acrònim de JavaScript Object Notation) és un estàndard obert basat en text dissenyat per a intercanvi de dades llegible per humans. Deriva del llenguatge script JavaScript, per a representar estructures de dades simples i llistes associatives, anomenades objectes. Malgrat la seva relació amb el JavaScript, té implementacions per a gran part dels llenguatges de programació.

El format JSON s'utilitza habitualment per serialitzar i transmetre dades estructurades en una connexió de xarxa. S'utilitza principalment per intercanviar dades entre un servidor i una aplicació web, sent una alternativa a l'XML. S'utilitza freqüentment en aplicacions Ajax.

La principal avantatja que ofereix el llenguatge és que representa millor l'estructura de les dades i requereix menys codificació i computació.

El format JSON a guanyat presencia en els últims anys i a dia d'avui té uns nivells d'utilització iguals que el XML, podem veure una gràfica de Google Trends:

https://trends.google.com/trends/explore?date=all&q=XML,JSON



Tipus de dades

JSON permet els següents formats de dades:

- Número

```
"preu" : 9.95
```

- String

```
"nom" : "John"
```

- Boolea

```
"active" : true
```

- Nul

```
"photo" : null
```

- Objecte

```
{"firstName":"John", "lastName":"Doe"}
```

- Array

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

Exemples

Un **exemple** d'un fitxer json de dades personals és:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "student": true,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

Podem veure un exemple de JSON generat per **l'API de Google Maps**:

```
{"markers": [
        {
           "point":new GLatLng(40.266044,-74.718479),
           "homeTeam":"Lawrence Library",
           "awayTeam":"LUGip",
           "markerImage":"images/red.png",
           "information": "Linux users group meets second Wednesday of each month.",
           "fixture":"Wednesday 7pm",
           "capacity":"",
           "previousScore":""
        },
        {
           "point":new GLatLng(40.211600,-74.695702),
           "homeTeam":"Hamilton Library",
           "awayTeam":"LUGip HW SIG",
           "markerImage":"images/white.png",
           "information": "Linux users can meet the first Tuesday of the month to work
out harward and configuration issues.",
           "fixture":"Tuesday 7pm",
           "capacity":"",
           "tv":""
        },
        {
           "point":new GLatLng(40.294535,-74.682012),
           "homeTeam":"Applebees",
           "awayTeam":"After LUPip Mtg Spot",
           "markerImage":"images/newcastle.png",
           "information": "Some of us go there after the main LUGip meeting, drink brews,
and talk.",
           "fixture":"Wednesday whenever",
           "capacity":"2 to 4 pints",
           "tv":""
        },
] }
```

Aquí tenim una comparació entre un fitxer XML i JSON amb les mateixes dades:

Com podeu veure el fitxer JSON és més entenedor i facil de llegir que el fitxer XML, ja que no

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<employees>
    <employee>
        <id>1</id>
        <firstName>Leonardo</firstName>
        <lastName>DiCaprio</lastName>
        <photo>http://1.bp.blogspot.com/-zvS_6Q1IzR8/T516qvnRmcI
/AAAAAAAABcc/HXO7HDEJKo0/s200/Leonardo+Dicaprio7.jpg</photo>
    </employee>
    <employee>
        <id>2</id>
        <firstName>Johnny</firstName>
        <lastName>Depp</lastName>
        <photo>http://4.bp.blogspot.com/_xR71w9-qx9E/SrAz--pu0MI
/AAAAAAAAC38/2ZP28rVEFKc/s200/johnny-depp-pirates.jpg</photo>
    </employee>
    <employee>
        <id>3</id>
        <firstName>Hritik</firstName>
        <lastName>Roshan</lastName>
        <photo>http://thewallmachine.com/files/1411921557.jpg</photo>
    </employee>
</employees>
```

```json
{
    "employees": {
        "employee": [
            {
                "id": "1",
                "firstName": "Leonardo",
                "lastName": "DiCaprio",
                "photo": "http://1.bp.blogspot.com/-zvS_6Q1IzR8
/T516qvnRmcI/AAAAAAAABcc/HXO7HDEJKo0/s200/Leonardo+Dicaprio7.jpg"
            },
            {
                "id": "2",
                "firstName": "Johnny",
                "lastName": "Depp",
                "photo": "http://4.bp.blogspot.com/_xR71w9-qx9E/SrAz
--pu0MI/AAAAAAAAC38/2ZP28rVEFKc/s200/johnny-depp-pirates.jpg"
            },
            {
                "id": "3",
                "firstName": "Hritik",
                "lastName": "Roshan",
                "photo": "http://thewallmachine.com/files/1411921557
.jpg"
            }
        ]
    }
}
```

Codi exemple

JavaScript

JSON va ser creat com part de JavasScript i la forma de llegir un document s'utilitza la sentencia parse()

```
<script>
var text = '{"employees":[' +
'{"firstName":"John","lastName":"Doe" },' +
'{"firstName":"Anna","lastName":"Smith" },' +
'{"firstName":"Peter","lastName":"Jones" }]}';

obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Aquest codi imprimeix en l'etiqueta HTML demo el text: "Anna Smith".

PHP

PHP incorpora funcions per tractar XML i JSON com la majoria de llenguatges de programació. Aquest codi llegeix un fitxer amb format XML, el transforma en JSON i finalment el transforma en un array de PHP.

```
<?php
$xml = simplexml_load_file("Prescripcion.xml");

$json = json_encode($xml);

$array = json_decode($json, TRUE);

?>
```

Una vegada tenim la informació en format XML, JSON o array podem recorrer o cercar dins l'objecte.

Eines per al tractament de fitxers json

Editor Online per a JSON: http://www.jsoneditoronline.org/

App Editor Online per a JSON integrat en el navegador: JSON Editor

Conversor XML-JSON: http://www.utilities-online.info/xmltojson

Plugin per a Firefox o Chrome per a la correcta visualització de fitxers JSON: JSONView

Lectura d'un fitxer json amb java

Per a poder llegir un fitxer JSON amb Java haurem d'importar la llibreria json que ens permet passar un json a objecte Java i a l'inrevés.

Preparem ECLIPSE per poder utilitzar la llibreria externa gson de google

Crea un nou projecte amb *Maven (gestor de dependències i gestor de projectes)* a ECLIPSE.

Crear el projecte

```
Eclipse > File > New... > Others > Maven > Maven project > Use default Workspace... > org.apache.maven.archetyper maven-archetype-quickstart > Group ID: dominiempres Artifact Id: nom del projecte > Finish
```

Maven

Clica sobre `pom.xml`, clica sobre la última pestanya de baix del document on posa pom.xml i busca:

```
<dependencies>
	Aquí dins hi haurà codi.
</dependencies>
```

Ves a https://github.com/google/gson i busca Gson amb Maven i copia el codi i l'enganxes dins de l'arxiu pom obert anteriorment.

```
<dependencies>
	Aquí dins hi haurà codi.
	<dependency>
	  <groupId>com.google.code.gson</groupId>
	  <artifactId>gson</artifactId>
	  <version>2.8.2</version>
	</dependency>
</dependencies>
```

Guarda

Ara vés al projecte i dins de Maven dependències veuràs les llibreries gson.

Per saber més sobre Maven: https://jarroba.com/maven-en-eclipse/

https://blog.openalfa.com/como-leer-y-escribir-ficheros-json-en-java

Crea el següents fitxers per fer la prova:

Donat el següent fitxer JSON:

```
{
"responsable":
    {
    "Nombre" : "Juan",
    "Edad": 28,
    "Aficiones": ["Música", "Cine", "Tenis"],
    "Residencia": "Madrid"
    },
"empleados":
    [
        {
        "Nombre" : "Elena",
        "Edad": 26,
        "Aficiones": ["Música", "Cine"],
        "Residencia": "Madrid"
        },
        {
        "Nombre" : "Luis",
        "Edad": 31,
        "Aficiones": ["Teatro", "Cine", "Fútbol"],
        "Residencia": "Madrid"
        }
    ]
```

```
}
```

Un possible codi Java de lectura del seu contingut seria:

```
import java.io.FileReader;

import com.google.gson.JsonParser;

import com.google.gson.JsonElement;

import com.google.gson.JsonObject;

import com.google.gson.JsonArray;

import com.google.gson.JsonPrimitive;

import java.util.Map.Entry;


public class lee_json {

    public static void main(String args[]) throws java.io.IOException {

        JsonParser parser = new JsonParser();

        FileReader fr = new FileReader("datos.json");

        JsonElement datos = parser.parse(fr);

        dumpJSONElement(datos);

    }


public static void dumpJSONElement(JsonElement elemento) {

    if (elemento.isJsonObject()) {

        System.out.println("Es objeto");

        JsonObject obj = elemento.getAsJsonObject();

        java.util.Set<java.util.Map.Entry<String,JsonElement>> entradas =
obj.entrySet();

        java.util.Iterator<java.util.Map.Entry<String,JsonElement>> iter =
entradas.iterator();

        while (iter.hasNext()) {

            java.util.Map.Entry<String,JsonElement> entrada = iter.next();

            System.out.println("Clave: " + entrada.getKey());

            System.out.println("Valor:");

            dumpJSONElement(entrada.getValue());

        }


    } else if (elemento.isJsonArray()) {
```

```
        JsonArray array = elemento.getAsJsonArray();
        System.out.println("Es array. Numero de elementos: " + array.size());
        java.util.Iterator<JsonElement> iter = array.iterator();
        while (iter.hasNext()) {
            JsonElement entrada = iter.next();
            dumpJSONElement(entrada);
        }
    } else if (elemento.isJsonPrimitive()) {
        System.out.println("Es primitiva");
        JsonPrimitive valor = elemento.getAsJsonPrimitive();
        if (valor.isBoolean()) {
            System.out.println("Es booleano: " + valor.getAsBoolean());
        } else if (valor.isNumber()) {
            System.out.println("Es numero: " + valor.getAsNumber());
        } else if (valor.isString()) {
            System.out.println("Es texto: " + valor.getAsString());
        }
    } else if (elemento.isJsonNull()) {
        System.out.println("Es NULL");
    } else {
        System.out.println("Es otra cosa");
    }
}
}
```

Executa el codi

~~Ens ha de quedar clar que la lectura sobre arrays continguts en JSON es realitzarà sempre amb una estructura iterativa (en aquest cas, un for), mentre que l'accés als valors concrets es portarà a terme mitjançant l'especificació de les keys (claus) dins de claudàtors: objecte ['key'] . En el cas d'accedir a subvalors, l'accés es realitzaria mitjançant: objecte['key']['subkey']~~

Lectura d'un fitxer JSON amb JavaScript

A l'igual que llegim fitxers JSON amb Python, es pot realitzar la seva lectura amb llenguatge JavaScript per així mostrar el seu contingut en una pàgina HTML.

En el següent exemple, es mostra la càrrega i lectura d'un fitxer en JavaScript:

```
<html>

<head>

<script type="text/javascript">
var JSONObject = [
{
"titol": "La plaça del Diamant",
"autor": {
"dataNaixement": "10/10/1908",
"nom": "Mercé Rodoreda"
},
"dataPublicacio": "1962"
},
{
"titol": "1984",
"autor": {
"dataNaixement": "25/06/1903",
"nom": "George Orwell"
},
"dataPublicacio": "1947"
},
{
"titol": "El jardí oblidat",
"autor": {
"dataNaixement": "28/03/1976",
"nom": "Kate Morton"
},
"dataPublicacio": "2010"
}
];
</script>

</head>
<body>
<h2>Llistat de llibres JSON</h2>

<script type="text/javascript">
document.write("<ul>");
for(var i in JSONObject){
var cad = JSONObject[i].titol + "," +
JSONObject[i].autor.dataNaixement + "," +
JSONObject[i].autor.nom + "," +
JSONObject[i].dataPublicacio;
document.write("<li>" + cad + "</li>");
}
document.write("</ul>");
</script>
```

```
</body>

</html>
```

Web Services REST

Un **web service** és una tecnologia que utilitza un conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions. Els web services han evolucionat molt en aquesta darrera dècada i l'apareixement de REST ha estès el seu ús gràcies a la seva simplicitat.

REST (Representational State Transfer, Transferència d'Estat Representacional) és una tècnica SOA (Service Oriented Arquitecture) per a sistemes distribuïts que utilitza HTTP com a protocol de comunicació, sense que siguin necessàries abstraccions addicionals com en RPC, SOAP o Corba.

Els sistemes que segueixen els principis REST es diuen RESTful, per tant, parlarem de web services RESTful, aunque se ha extendido por igual el terme de webservices REST.

REST està basat en els següents tecnologies:

- **HTTP** (Protocol Client/Servidor sense estat): cada missatge HTTP conté tota la informació necessària per comprendre la petició. Com a resultat, ni el client ni el servidor necessiten recordar cap estat anterior de les comunicacions entre missatges. Malgrat això, en la pràctica, moltes aplicacions basades en HTTP utilitzen cookies i altres mecanismes per mantenir l'estat de la sessió.
- **Conjunt d'operacions HTTP**: on destaquen: POST, GET, PUT i DELETE.
- **Sintaxi universal per identificar els recursos**. En un sistema REST cada recurs és dirreccionable únicament per la seva **URI**.
- **Ús d'hipervincles**: pe a la informació de l'aplicació com per a les transicions d'estat de l'aplicació. La representació d'aquest estat en un sistema REST són típicament XML, HTML, JSON o PHP. Com a resultat, és possible navegar a través d'un recurs REST a molts altres, simplement seguint els hipervincles, sense requerir l'us de registres o altres estructures.

Lectura d'un webservice amb Python

En el següent exemple es veurà la consulta d'un service web REST (método GET sobre HTTP) que retorna un fitxer JSON codificat en llenguatge Python:

```
import urllib.request

import json

uri = 'http://api.worldweatheronline.com/free/v1/weather.ashx?q=Reus&format=json&num_of_days=5&key=xxxxxxxxxxxxxxxxxxxxxxx'

response = urllib.request.urlopen(uri).read().decode("utf-8")
```

```
json = json.loads(response)



humitat=json["data"]["current_condition"][0]["humidity"]
temperatura=json["data"]["current_condition"][0]["temp_C"]

print( "La temperatura actual es de "+temperatura+" graus centigrads i

la humitat del "+humitat+"%")
```

Si ens fixem en la variable 'uri' aquesta conté una l'adreça URI de petició de consulta del temps de la pàgina web www.worldweatheronline.com per connectar amb el seu webservice REST en format JSON.

Dins d'aquesta URI un dels paràmetres a facilitar serà 'key'. Aquest valor és inclòs per controlar el número de peticions mensuals que es realitza, amb el objectiu de oferir un servei gratuït per a particulars, però un servei de pagament si es vol realitzar un ús ampli. Per aconseguir aquesta key els usuaris s'han de registrar en la seva web.

Dades Obertes que permeten peticions REST

Actualment es parla molt d'**Open Data**, que és la filosofia que persegueix que la informació sobre dades públiques estiguin a l'abast de la ciutadania, sense restriccions de copyright i patents. Podem obtenir dades socials, geogràfiques, econòmiques, polítiques, de serveis...

Alguns hipervincles d'interès que permeten peticions REST són els següents:

- OpenData BCN: http://opendata.bcn.cat/opendata/
- Idescat (Institut Nacional d'Estadística de Catalunya): http://www.idescat.cat/api/
- Dades Obertes Gencat: http://www20.gencat.cat/portal/site/dadesobertes
- Datos Abiertos nivel estatal: http://datos.gob.es/datos/

Validació de fitxers json amb json schemes

Els esquemes JSON són una especificació per a fitxers JSON que defineixen la seva estructura. Gràcies als esquemes es garanteix com pot ser modificat i proporciona una validació, documentació i control de la interacció sobre dades en format JSON. Els JSON schemes són a JSON l'equivalent dels XML Schemes a XML.

Donat el següent fitxer JSON:

```
{
    "id": 1,
    "name": "Foo",
    "price": 123,
    "tags": [ "Bar", "Eek" ],
    "stock": {
        "warehouse": 300,
        "retail": 20
    }
}
```

Un exemple d'un esquema per al fitxer JSON anterior seria:

```
{
    "name": "Product",
    "properties":
     {
        "id": {
            "type": "number",
            "description": "Product identifier",
            "required": true
        },
        "name": {
            "type": "string",
            "description": "Name of the product",
            "required": true
        },
        "price": {
            "type": "number",
            "minimum": 0,
            "required": true
        },
        "tags": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "stock": {
            "type": "object",
            "properties": {
                "warehouse": {
                    "type": "number"
                },
                "retail": {
                    "type": "number"
                }
            }
        }
    }
}
```

Alguns exemples més: https://spacetelescope.github.io/understanding-json-schema/

Altres links d'interès

- Directori general de webservices: http://www.programmableweb.com/apis/directory/
- Vídeo explicatiu webservices REST: http://vimeo.com/53338309

Webgrafia

http://www.json.org/. Pàgina oficial del projecte i especificació tècnica.

https://en.wikipedia.org/wiki/JSON. Informació genèrica sobre el llenguatge.

http://www.jsoneditoronline.org/. Visualitzador/Validador de JSON

Wikipèdia:

http://en.wikipedia.org/wiki/JSON

http://es.wikipedia.org/wiki/Representational_State_Transfer

W3Schools:

http://www.w3schools.com/json/default.asp