What are JSON objects?

JSON objects contain key-value pairs.

For example:

{

“Id” : 2

“Name” : “Anchita sharma”

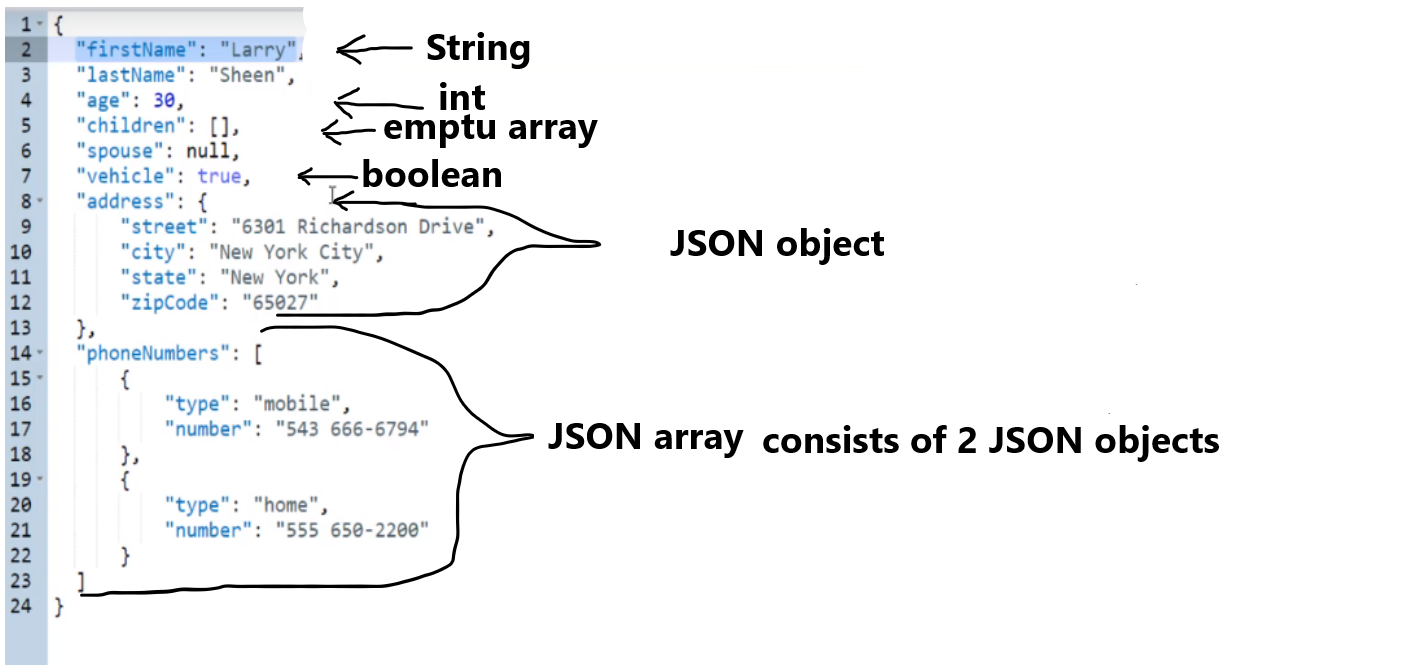
“Emai” l: “anchita.sharma@codingninjas.com”

}

Id, name, and email are called keys. The answers to these keys are called values. A key-value pair is separated by a colon.

The collection of key-value pairs is stored in JSON objects.

It is surrounded by curly brackets {}.



Every JSON object is enclosed in curly braces.

On a left side it has a key, and on the right side value

The key will always be a String ! and always unique

The value of any data-type that is supported.

When you store JSON in a file it should have a .JSON extension.

**Groovy implementation of JSON path (used by default by Rest Assured by default)**

JSON path implementation by default uses Groovy`s syntax (it is called - Gpath), and it uses dot-object notation. We do not need to specifically add dependency for that, because it is included transitively when we add Rest Assured dependency. JsonSlurper is a class that parses JSON into Groovy data structures (objects) such as maps, lists and primitive types like Integer, Double, Boolean and String. In other words it became a Java objects, and we can process them further.

If we want to fetch any JSON object or JSON array or individual key, we can use something called JSON path.

The difference between Gpath and xpath is that the xpath uses ‘/’ as a separator, and Jpath uses dot.

It is useful to use onlyne json parser that can parse json object and convert it to Groovy object and then we can use Jpath expressions to fetch the individual objects.

how groovy converts the json to object:

{} in json –> a Map

[] in json –> a List

<https://www.jdoodle.com/execute-groovy-online/> ← JSON online parser.

example:

import groovy.json.JsonSlurper ← this is Groovy programming

def object = new JsonSlurper().parseText(

'''

{

"workspaces": [

{

"id": "4ae9fb48-d9dc-497b-8347-ca6f9b2bdf8f",

"name": "My Workspace",

"type": "personal",

"visibility": "personal"

},

{

"id": "71622f8e-c238-4b98-8aaf-48493bbacb4d",

"name": "Team Workspace",

"type": "team",

"visibility": "team"

},

{

"id": "9dd28d39-8330-430a-a4b3-8db221d3cc2e",

"name": "JJ's Workspace",

"type": "team",

"visibility": "team"

}

]

}

'''

)

def query = object ← will print all the JSON object

def query = object.workspaces[0].visibility ← personal (the field from the first element)

def query = object.workspaces.visibility ← [personal, team, team] (the field from the every element returned in the form of ArrayList).

def query = object.workspaces ← will print all the JSON array (the name of the array is workspaces)

println query

<https://github.com/json-path/JsonPath> ← link of the source

**Jayway JsonPath ← Json Path (different) implementation**

In order to use this implementation we need to add this as a dependency in a rest assured project. JsonPath is available at the Central Maven Repository. Maven users add this to your POM.

<dependency>

<groupId>com.jayway.jsonpath</groupId>

<artifactId>json-path</artifactId>

<version>2.9.0</version>

</dependency>

The "root member object" in JsonPath is always referred to as $ regardless if it is an object or array.

JsonPath expressions can use the dot–notation

$.store.book[0].title

or the bracket–notation

$['store']['book'][0]['title']

So both implementations work in Java with Rest Assured and we can use any of them. Groovy`s implementation goes by default with Rest Assured. If we want to use the JaywayJsonPath implementation we need to add a specific dependency apart from Rest Assured. And both implementations differ syntactically.