

Json example 1

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the
Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour ",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
```

Json example 1 (cont)

```
> "price": 19.95
}
}
}
```

Extract value from Example 1

```
- Read the json: String --> JsonPath
JsonPath jsonPath = JsonPath.from(res);

- First category
jsonPath.get("store.book[0].category");
//reference

- All category
jsonPath.getList("store.book.category");
//[reference, fiction, fiction, fiction]

- All author that have price > 10
jsonPath.getList("store.book.findAll
{it.price > 10}.author");
//[Evelyn Waugh, J. R. R. Tolkien]

- All author that have category = fiction
jsonPath.getList("store.book.findAll
{it.category == 'fiction' }.author");
//[Evelyn Waugh, Herman Melville, J. R. R.
Tolkien]

- Count number of author
jsonPath.getInt("store.book.author.size()");
//4

- First author that have price > 10
jsonPath.getString("store.book.find
{it.price > 10}.author");
//Evelyn Waugh

- Color of bicycle
jsonPath.getString("store.bicycle.color");
//red
```

Convert to Array or List of Object

- Response

```
[
  {
    "id": "11",
    "name": "abc"
  }
]
```

- POJO

```
@Data
public class Root {
    private String id;
    private String name;
}
```

- Convert to object

```
Root root = given().extract().as(Root.class)[0];
or
Root root = given().extract().jsonPath().getList("", Root.class).get(0);
```

Json Example 2 - extract

```
{
  "foo.bar.baz": {
    "0.2.0": "test"
  }
}

JsonPath.from(json).getString("foo.bar.baz." + "0.2.0")
//test

{
  "a-b": "minus",
  "a.b": "dot",
  "a.b-c": "both"
}

JsonPath.from(json).getString("a.b-c");
//both
```

Json Example 2 - extract (cont)

```
> {
  "map": {
    "true": 12.3,
    "false": 15.0
  }
}

JsonPath.from(json).getFloat("map.false")
//15.0f

JsonPath.from(json).getFloat("map.true")
//12.3f
```

Json Example 3 - extract

```
{
  "semester": "Fall 2015",
  "groups": [
    {
      "url": "http://cp.hbu.sinn-ess-jb.clo.uda.pp.net/CA2/",
      "error": "NO AUTHOR /CLASS -INFO"
    },
    {
      "url": "http://ca2-ebk-ir.hcl.oud.com/C A2N ew/",
      "author": "Ebbe, Kasper, Christ-offer",
      "class": "A klasse n",
      "group": "Gruppe: Johns Llama Herders A/S"
    },
    {
      "url": "http://ca2-c.hri-sli.nd.rhc.lou.d.com/CA2.Fin.al/",
      "error": "NO AUTHOR /CLASS -INFO"
    },
    {
      "url": "http://ca2-bern-ill.e.r.hcl.oud.com/N YCA 2/",
      "author": "Marta, Jeanette, Pernille",
      "class": "DATA",
    }
  ]
}
```

Json Example 3 - extract (cont)

```
> "group": "Group: MJP"
},
{
  "siteUrl": "https://ca2-afn.rhcloud.com:8443/company.jsp",
  "error": "NO AUTHOR/CLASS-INFO"
},
{
  "siteUrl": "http://ca-smcphbusiness.rhcloud.com/ca2/index.jsp",
  "authors": "Mikkel, Steffen, B Andersen",
  "class": "A Class Computer Science",
  "group": "1"
}
]
}
JsonPath.from(json).getList("groups.getAt('class')");
//[null, A klassen, null, DAT A, null, A Class Computer Science]
- Additional: how to move null from a list
list.removeAll(Collections.singleton(null));
System.out.println("list = " + list);
//[A klassen, DAT A, A Class Computer Science]
```

Special cases (cont)

```
> var text = ""
[
  [
    "Test1",
    "Test2"
  ],
  [
    "Test3",
    "Test4"
  ]
]
"";
List<String> json = JsonPath.from(text).get("collect{it.getAt(0)}");
System.out.println("json = " + json);
//json = [Test1, Test3]
```

Special cases

```
//Multiple nested arrays
JsonPath.from(response)
.get("communicationPolicy.rules.flatten()
  .findAll {it.test1 == 'TRANSACT IONAL'}
  .value.flatten().findAll {it.channel ==
'EMAIL '}.value");
System.out.println(value);
//Check contains
"response.data.attributes.findAll{
  it.triggeredBy.contains(restrictedJsonRobotObject.response.data.attributes.find{
    it.name.equals('Investigate Suggestions')}.id)}.name "
restAssured.jsonRobotObject --> refer to root of json
```

