



org.json - parsing

Parsing JSON using org.json



What is “parsing”?

A parser is a software component that takes input data (frequently text) and builds a data structure – often some kind of parse tree, abstract syntax tree or other hierarchical structure – giving a structural representation of the input, checking for correct syntax in the process.

Wikipedia

So, take some text, like JSON, build a tree or some other data type from it, and check that it is well-formed (syntactically correct).

When do we need to “parse” JSON?

If our Java application gets data from some service or file in the form of a JSON document, we need to transform it from application/JSON to some kind of representation in Java objects.

The goal is to get the JSON from somewhere and create a representation of that data in Java.

So we need to “parse” the JSON and traverse the tree and create some Java data type object from it (like a List for instance).

In this simple example, we'll just get the data out from the JSON object, piece-by-piece.

Example schedule for substitutes

We have an application which should read JSON about the schedule for substitute teachers.

A schedule is a list of assignments, and one assignment could be:

```
{
  "date": "2018-01-17 08:00:00",
  "substitute": {
    "name": "Rikard"
  },
  "school": {
    "school_name": "Yrigo",
    "address": "Lärdomsgatan 3 402 72 GÖTEBORG"
  }
}
```

The data comes from a database with three tables

```
CREATE TABLE substitute(substitute_id integer primary key,  
name text);  
CREATE TABLE schedule(day datetime, substitute_id integer,  
school_id integer);  
CREATE TABLE "school"(school_id integer primary key,  
school_name text, address text);
```

The data comes from a database - school table

```
CREATE TABLE "school"(school_id integer primary key, school_name text,  
address text);
```

```
sqlite> select * from school;
```

```
1|Yrigo|Lärdomsgatan 3 402 72 GÖTEBORG
```

```
2|ITHS|Ebbe Lieberathsgatan 18 c 412 65 Göteborg
```

```
3|Jensen|Kruthusgatan 17 411 04 Göteborg
```

```
4|Angeredsgymnasiet|Grepattan 9 424 65 Angered
```

```
5|Aniaragymnasiet|Kyrkogatan 46 411 15 Göteborg
```

```
6|Bernadottegymnasiet|Skånegatan 18 411 40 Göteborg
```

```
7|Bräckegymnasiet|Uppfinnaregatan 1 417 56 Göteborg
```

```
8|Burgårdens utbildningscentrum|Skånegatan 20 402 29 Göteborg
```

The data comes from a databases - substitute table

```
CREATE TABLE substitute(substitute_id integer primary key, name text);
```

```
sqlite> select * from substitute;
```

```
1|Rikard  
2|Henrik  
3|Anders  
4|Nahid  
5|Conny  
6|Svante  
7|Elisabeth  
8|Eva  
9|Kristina  
10|Bengt
```

The data comes from a database - schedule table

```
CREATE TABLE schedule(day datetime, substitute_id integer, school_id integer);
```

```
sqlite>
```

```
select * from schedule;
```

```
2018-01-15 08:00:00|1|1  
2018-01-16 08:00:00|1|1  
2018-01-17 08:00:00|1|1  
2018-01-18 08:00:00|1|2  
2018-01-16 08:00:00|2|1  
2018-01-17 08:00:00|2|1  
2018-01-18 08:00:00|2|3  
2018-01-16 08:00:00|3|2  
2018-01-17 08:00:00|3|4  
2018-01-18 08:00:00|3|5  
2018-01-16 08:00:00|4|8
```


The JSON file - schedule for 2018-01-17

```
[
  {
    "date": "2018-01-17 08:00:00",
    "substitute": {
      "name": "Rikard"
    },
    "school": {
      "school_name": "Yrgo",
      "address": "Lärdomsgatan 3 402 72
GÖTEBORG"
    }
  },
  {
    "date": "2018-01-17 08:00:00",
    "substitute": {
      "name": "Henrik"
    },
    "school": {
      "school_name": "Yrgo",
      "address": "Lärdomsgatan 3 402 72
GÖTEBORG"
    }
  },

```

```

  {
    "date": "2018-01-17 08:00:00",
    "substitute": {
      "name": "Anders"
    },
    "school": {
      "school_name": "Angeredsgymnasiet",
      "address": "Grepkatan 9 424 65
Angered"
    }
  }
]
```

Goal - parse JSON, create List<Schedule>

```
public class Schedule {
    private String date;
    private Substitute substitute;
    private School school;
    public Schedule(String date, Substitute substitute, School school) {
        this.date = date;
        this.substitute = substitute;
        this.school = school;
    }
    public String date() { return date; }
    public Substitute substitute() { return substitute; }
    public School school() { return school; }
    public String toString() {
        return date + " " + substitute + " " + school;
    }
}
```

Goal - parse JSON, create List<Schedule>

```
public class Substitute {  
    private String name;  
    public Substitute(String name) {  
        this.name = name;  
    }  
  
    public String name() {  
        return name;  
    }  
  
    public String toString() {  
        return name;  
    }  
}
```

Goal - parse JSON, create List<Schedule>

```
public class School {
    private String name;
    private String address;

    public School(String name, String address) {
        this.name = name;
        this.address = address;
    }

    public String name() { return name; }

    public String address() { return address; }

    public String toString() {
        return name + " " + address;
    }
}
```

Goal - parse JSON, create List<Schedule>

```
public class Parser {  
    // stuff  
}
```

The JSON file - schedule for 2018-01-17

```
[
  {
    "date": "2018-01-17 08:00:00",
    "substitute": {
      "name": "Rikard"
    },
    "school": {
      "school_name": "Yrgo",
      "address": "Lärdomsgatan 3 402 72
GÖTEBORG"
    }
  },
  {
    "date": "2018-01-17 08:00:00",
    "substitute": {
      "name": "Henrik"
    },
    "school": {
      "school_name": "Yrgo",
      "address": "Lärdomsgatan 3 402 72
GÖTEBORG"
    }
  },

```

```
{
  "date": "2018-01-17 08:00:00",
  "substitute": {
    "name": "Anders"
  },
  "school": {
    "school_name": "Angeredsgymnasiet",
    "address": "Grepgatan 9 424 65
Angered"
  }
}
]
```

Strategy:

iterate over each schedule in the array

from schedule:

get date

get substitute

get school

create Schedule object

add to list

return the List<Schedule>

Parser - parse JSON, create List<Schedule>

```
// 1. Read JSON-file to a String
// 2. create JSONArray:
JSONTokener jt = new JSONTokener(sb.toString());
JSONArray scheduleArray = new JSONArray(jt);
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 1. Read JSON-file to a String
StringBuilder sb = new StringBuilder();
String ls = System.getProperty("line.separator");
BufferedReader reader = new BufferedReader(new FileReader (JSON_FILE));
String line;
while((line = reader.readLine()) != null) {
    sb.append(line);
    sb.append(ls);
}
reader.close();
```


Parser - parse JSON, create List<Schedule>

```
// 1. Read JSON-file to a String
StringBuilder sb = new StringBuilder();
String ls = System.getProperty("line.separator");
BufferedReader reader = new BufferedReader(new FileReader (JSON_FILE));
String line;
while((line = reader.readLine()) != null) {
    sb.append(line);
    sb.append(ls);
}
reader.close();

// 2. create JSONArray:
JSONTokener jt = new JSONTokener(sb.toString());
JSONArray scheduleArray = new JSONArray(jt);
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    // Get a whole schedule object:
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                              schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    // Get the date as a String
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    // Get the Substitute object:
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    // Get the School object:
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    // Create the Java Substitute object from your own class:
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    // Create the Java School object from your own class:
    School school = new School(schoolJSON.getString("school_name"),
        schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    // Create the Java Schedule object from your own class:
    Schedule currentSchedule = new Schedule(date, substitute, school);
    schedules.add(currentSchedule); // a List<Schedule>
}
```


Parser - parse JSON, create List<Schedule>

```
// 3. Loop over each schedule object, create Java Schedule object
for(int i=0;i<scheduleArray.length();i++){
    JSONObject schedule = scheduleArray.getJSONObject(i);
    String date = schedule.getString("date");
    JSONObject substituteJSON = schedule.getJSONObject("substitute");
    JSONObject schoolJSON = schedule.getJSONObject("school");
    Substitute substitute = new Substitute(substituteJSON.getString("name"));
    School school = new School(schoolJSON.getString("school_name"),
                               schoolJSON.getString("address"));
    Schedule currentSchedule = new Schedule(date, substitute, school);
    // Add the Java Schedule object to the List<Schedule>
    schedules.add(currentSchedule); // a List<Schedule>
}
```

Parser - full code

```
private static final String JSON_FILE = "2018-01-17.json";

private static String JSONfromFile() throws IOException {
    StringBuilder sb = new StringBuilder();
    String ls = System.getProperty("line.separator");
    BufferedReader reader = new BufferedReader(new FileReader (JSON_FILE));
    String line;
    while((line = reader.readLine()) != null) {
        sb.append(line);
        sb.append(ls);
    }
    reader.close();
    return sb.toString();
}
```

Parser - full code

```
public static List<Schedule> parse() {
    List<Schedule> schedules = new ArrayList<>();
    try {
        JSONTokener jt = new JSONTokener(JSONFromFile());
        JSONArray scheduleArray = new JSONArray(jt);
        for(int i=0;i<scheduleArray.length();i++){
            JSONObject schedule = scheduleArray.getJSONObject(i);
            String date = schedule.getString("date");
            JSONObject substituteJSON = schedule.getJSONObject("substitute");
            JSONObject schoolJSON = schedule.getJSONObject("school");
            Substitute substitute = new Substitute(substituteJSON.getString("name"));
            School school = new School(schoolJSON.getString("school_name"),
                                      schoolJSON.getString("address"));

            Schedule currentSchedule = new Schedule(date, substitute, school);
            schedules.add(currentSchedule);
        }
    } catch (IOException|JSONException e) {
        System.err.println("Parse error: " + e.getMessage());
        e.printStackTrace();
    }
    return schedules;
}
```

Main class - full code

```
import java.util.List;
```

```
public class ParseSchedule {  
    public static void main(String[] args) {  
        List<Schedule> schedules = Parser.parse();  
        schedules.stream().forEach(System.out::println);  
    }  
}
```

```
$ javac -cp ./org.json.jar *.java && java -cp ./org.json.jar ParseSchedule  
2018-01-17 08:00:00 Rikard Yrgo Lärdomsgatan 3 402 72 GÖTEBORG  
2018-01-17 08:00:00 Henrik Yrgo Lärdomsgatan 3 402 72 GÖTEBORG  
2018-01-17 08:00:00 Anders Angeredsgymnasiet Grepkatan 9 424 65 Angered
```

```
# note: use -cp ".;org.json.jar" (with a semicolon) on Windows
```

org.json api calls used

Create the JSONTokener object from the JSON String:

```
JSONTokener jt = new JSONTokener(a Java String with JSON);
```

org.json api calls used

Create a JSONArray object from the JSNTokener:

```
JSONArray scheduleArray = new JSONArray(jt);
```

org.json api calls used

Parse a whole JSON schedule object from the Array in a loop, e.g.:

```
{
  "date": "2018-01-17 08:00:00",
  "substitute": {
    "name": "Rikard"
  },
  "school": {
    "school_name": "Yrgo",
    "address": "Lärdomsgatan 3 402 72 GÖTEBORG"
  }
}
```

```
for(int i=0;i<scheduleArray.length();i++){
  JSONObject schedule = scheduleArray.getJSONObject(i);
  ...
}
```

org.json api calls used

Getting a String from a JSONObject string, e.g.:

```
"date": "2018-01-17 08:00:00"
```

```
String date = schedule.getString("date"); // date is the named string  
// date now contains "2018-01-17 08:00:00" as a Java String
```


org.json api calls used

Getting a named JSON object, e.g.:

```
"substitute": {  
  "name": "Rikard"  
}
```

```
JSONObject substituteJSON = schedule.getJSONObject("substitute");
```

Getting the JSON string for "name" later becomes:

```
substituteJSON.getString("name")
```

org.json api calls used

Getting the school JSON object, e.g.:

```
"school": {  
  "school_name": "Yrgo",  
  "address": "Lärdomsgatan 3 402 72 GÖTEBORG"  
}
```

```
JSONObject schoolJSON = schedule.getJSONObject("school");
```

org.json api calls used

Getting the school_name and address from e.g.:

```
"school": {  
  "school_name": "Yrgo",  
  "address": "Lärdomsgatan 3 402 72 GÖTEBORG"  
}
```

```
School school = new School(schoolJSON.getString("school_name"),  
                           schoolJSON.getString("address"));
```

org.json api calls used

Recap and overview:

- Create a **JSONTokener** from a String with JSON:
`JSONTokener jt = new JSONTokener(some Java String with JSON);`
- Create a **JSONArray** from the **JSONTokener**:
`JSONArray scheduleArray = new JSONArray(jt);`
- Loop over each **JSONObject** (schedule) in the array:

```
for(int i=0;i<scheduleArray.length();i++){  
    JSONObject schedule = scheduleArray.getJSONObject(i);  
    ...  
}
```
- Get a nested json object from the schedule object:
`JSONObject substituteJSON = schedule.getJSONObject("substitute");`
- Get a Java String from a named json name/value pair:
`String date = schedule.getString("date");`

Further reading

- <http://www.studytrails.com/java/json/java-org-json/>
- https://www.codevoila.com/post/65/java-json-tutorial-and-example-json-java-orgjson#toc_5
- Getting org.json jar file:
<https://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22org.json%22%20AND%20a%3A%22json%22>
- API documentation: <https://stleary.github.io/JSON-java/>
- Presentation source code:
<https://github.com/progund/java-web/tree/master/java-json/org.json/parsing>