

Report SWE261P

Team Name: Endloop

Team Memembr: Bin Guo, Xin Tan, Chao Liu

FastJson

Github: <https://github.com/alibaba/fastjson>

- Fastjson, a library taht can be used to convert Java Objects into their JSON representation, is able to convert Json string to an equivalent Java object.
- Fastjson contain 14 packages ro provide the best performance on the server-side and android client and simeple toJSONString() and parseObject() methods to convert Java objects to JSONand vice versa. Also, it allows pre-existing unmodifiable objects to be converted to and form JSON and custom presentations for objects.

Package com.alibaba.fastjson

Interface Summary

| Interface | Description |
|-----------------|---|
| JSONAware | Beans that support customized output of JSON text shall implement this interface. |
| JSONStreamAware | Beans that support customized output of JSON text to a writer shall implement this interface. |

Class Summary

| Class | Description |
|------------------------|--|
| JSON | This is the main class for using Fastjson. |
| JSONArray | |
| JSONObject | |
| JSONPath | |
| JSONPath.FilterSegment | |
| JSONPObjct | |
| JSONReader | |
| JSONWriter | |
| TypeReference<T> | Represents a generic type T. |

Enum Summary

| Enum | Description |
|------------------------|-------------|
| PropertyNamingStrategy | |

Exception Summary

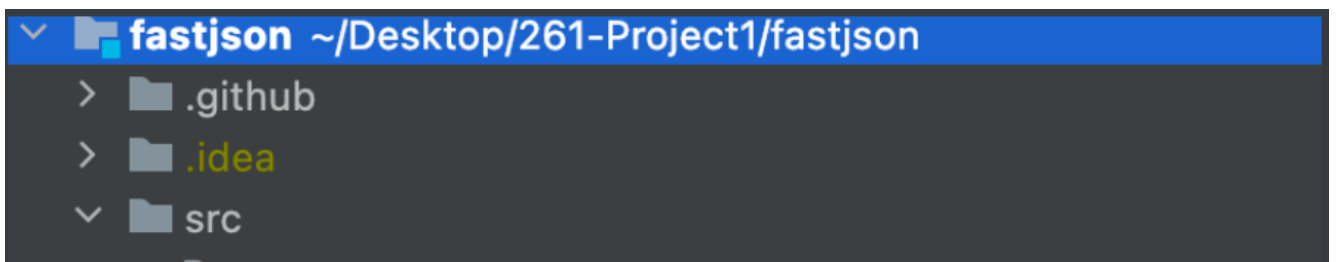
| Exception | Description |
|-------------------|-------------|
| JSONException | |
| JSONPathException | |

- LOC and Language:

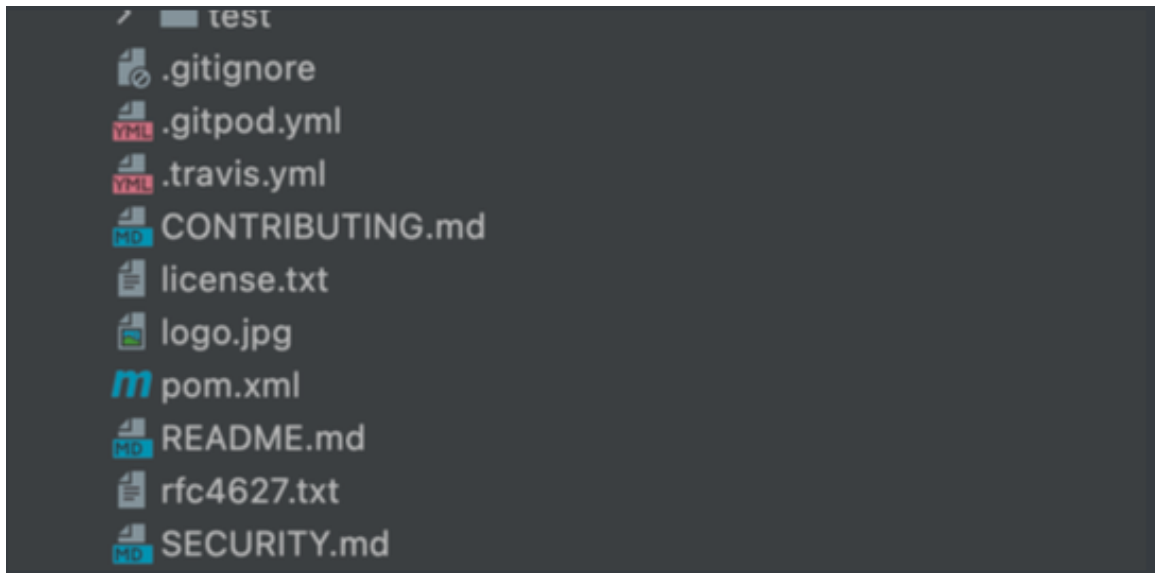
| Language | files | blank | comment | code |
|--------------|-------|-------|---------|--------|
| Java | 3120 | 53055 | 14008 | 186962 |
| JSON | 46 | 14 | 0 | 16340 |
| SQL | 1 | 21 | 12 | 3992 |
| Maven | 1 | 59 | 26 | 588 |
| XML | 17 | 35 | 1 | 385 |
| Markdown | 3 | 52 | 0 | 189 |
| Kotlin | 4 | 36 | 18 | 139 |
| JavaScript | 2 | 1 | 0 | 69 |
| YAML | 3 | 5 | 0 | 50 |
| Properties | 2 | 2 | 4 | 11 |
| Bourne Shell | 2 | 0 | 0 | 4 |
| SUM: | 3201 | 53280 | 14069 | 208729 |

Instructions for build up

- The file structure shown as below



- ▼ main
 - ▼ java
 - ▼ com.alibaba.fastjson
 - > annotation
 - > asm
 - > parser
 - > serializer
 - > spi
 - > support
 - > util
 - JSON
 - JSONArray
 - JSONAware
 - JSONException
 - JSONObject
 - JSONPatch
 - JSONPath
 - JSONPathException
 - JSONPObject
 - JSONReader
 - JSONStreamAware
 - JSONStreamContext
 - JSONValidator
 - JSONWriter
 - PropertyNamingStrategy
 - TypeReference
 - > META-INF
 - > resources



- How to build?
 1. Make a new directory
 2. git clone <https://github.com/alibaba/fastjson>
 3. Choose a IDE and then import the project to the IDE as a maven project. Then find pom.xml to reload the project
- How to run?
 1. In pom.xml, add two dependencies (android or others)

```
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.76</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.1.72.android</version>
</dependency>
```

Systematic functional testing & Partition testing:

- Systematic functional testing:
 - Select inputs that are especially valuable.
 - Usually by choosing representatives of classes that are apt to fail often or not at all. It usually isolate regions with likely failures.
 - functional testing usually implies systematic testing.

Using systematic functional testing can make use of the attributes of classes we have. For example, if our purpose is to estimate the proportion of needles to hay, we can use their weight attribute for filtering. If we just test and sample randomly, it will be huge work.

- Partition testing
 - Partition
 - It is one of the basic principles of general engineering principles.
 - Divide and conquer: divided complex activities into sets of simple activities that can be tackled independently. For example: we can partition testing process into: unit, integration, system, .. testing; we can partition analysis into modeling and analyzing the model.
 - Partition specification space for testing; Partition the program structure for testing.
 - Systematic partition testing
 - Sometimes failures are sparse in the space of possible inputs but dense in some parts of the space. Use systematic partition testing can find the failures with more chances.
 - Boundaries
 - We use data type or classes to be the boundary. Since the Fastjson have the function to convert different types of object to JSON, we can simply partition our test by the types of the input.
- Junit tests:
 -