fastjson

温绍锦（温高铁）

http://weibo.com/wengaotie

Alibaba Group

How to get?

- http://repo1.maven.org/maven2/com/alibaba/fastjson/
- Maven2

```
<dependency>
 <groupId>com.alibaba</groupId>
 <atifactId>fastjson</artifactId>
 <version>1.1.xxx</version>
</dependency>
```

Sourcecode

https://github.com/alibaba/fastjson

Alibaba Group

特点

- 最快
- API最简洁
- 功能强大
- 扩展性最好
- 稳定

Alibaba Group

一直最快

- 第三方测试表明，fastjson是Java语言中最快的JSON库。
- 两年来从未被超越。

	序列化耗时	反序列化耗时	总耗时	大小
kryo	588	814	1403	214
protobuf	1103	684	1787	238
fastjson	**1201**	**1216**	**2417**	**486**
jackson	1842	2421	4262	485
hessian	3812	6708	10519	501
bson/jackson	5645	6895	12541	506
gson	7421	5065	12485	486
java-built-in	5608	29649	35257	889
json-lib	27555	87292	114848	485

https://github.com/eishay/jvm-serializers/wiki

Alibaba Group

假定有序，Key不用读取，只做匹配

```
{“id”:123,”name”:”gaotie”}
```

类型预判，优化读取

```
public class VO {
    public int id;
    public String name;
}
```

Alibaba Group

内置裁剪版本的ASM实现算法

还能更快么？

能！

Alibaba Group

Fastjson可以做什么？

- Web框架处理JSON参数返回JSON结果
- Cache缓存对象
- 远程方法调用RPC
- Android/阿里云手机处理JSON
- MessageQueue传输对象
- 配置文件代替XML
- 保存数据到磁盘、数据库、Hbase

Alibaba Group

功能完备

- 支持序列化和反序列化
- 支持循环引用
- 支持泛型
- 能够定制序列化，可以过滤和修改
- 支持代理对象，cglib和javassist
- 自动识别各种日期格式
- 支持GetOnly的List/Map反序列化
- Stream API支持超大对象和JSON文本

Alibaba Group

API简洁

- 当前Java JSON库中最简洁的API

```
public abstract class JSON {
    static <T> T parseObject(String text, Class<T> clazz)；// JSON文本 -> JavaBean
    static String toJSONString(Object object)；  // JavaBean -> JSON文本

    static Object parse(String text)；// JSON文本 -> JSON对象
    static Object toJSON(Object javaObject)；  // Java对象 -> JSON对象
    static <T> T toJavaObject(JSON json, Class<T> clazz)；  // JSON对象 -> Java对象
}
```

Alibaba Group

序列化

```
User user = new User();
user.setId(123);
user.setName("wenshao");
String text = JSON.toJSONString(user);
System.out.println(text);
```

```
{"id":123,"name":"wenshao"}
```

Alibaba Group

反序列化

```
String text = "{id:123,name:'wenshao'}";
User o = JSON.parseObject(text, User.class);
```

```
String text = "[{'id':
123,'name':'wenshao'}]";
List<User> users = JSON.parseObject(text, new
TypeReference<List<User>>() {});
```

泛型支持

Alibaba Group

循环引用

```
Object[] array = new Object[1];
array[0] = array;

String text = JSON.toJSONString(array);
Assert.assertEquals("[{\"$ref\":\"@\"}]", text);

Object[] a2 = JSON.parseObject(text, Object[].class);
Assert.assertSame(a2, a2[0]);
```

Alibaba Group

支持双引号、单引号、无引号

标准，针对性优化，在fastjson中性能最好

{“id”:123,”name”:”wenshao”}

{’id’:123, ’name’:’wenshao’}　兼容

{id:123,name:”wenshao”}　兼容

Alibaba Group

```
public interface NameFilter extends SerializeFilter {
  String process(Object source, String name, Object value);
}
```

修改Name

```
public interface ValueFilter extends SerializeFilter {
  Object process(Object source, String name, Object value);
}
```

修改Value

```
public interface PropertyFilter extends SerializeFilter {
  boolean apply(Object source, String name, Object value);
}
```

根据Name和Value判断是否序列化

```
public interface PropertyPreFilter extends SerializeFilter {
  boolean apply(JSONSerializer serializer, Object source, String name);
}
```

根据Name判断是否序列化

Alibaba Group

序列化时修改Key

```
NameFilter filter = new NameFilter() {
    public String process(Object source, String name, Object value) {
        if (name.equals("id")) {
            return "ID";
        }
        return name;
    }
};

Map<String, Object> map = new HashMap<String, Object>();
map.put("id", 0);

String text = JSON.toJSONString(map, filter);
Assert.assertEquals("{\"ID\":0}", text);
```

按需要修改Key

Alibaba Group

序列化时修改Value

```
ValueFilter filter = new ValueFilter() {
    public Object process(Object source, String name, Object value) {
        if (name.equals("id")) {
            return 123;
        }
        return value;
    }
};

Map<String, Object> map = new HashMap<String, Object>();
map.put("id", 0);

String text = JSON.toJSONString(map, filter);
Assert.assertEquals("{\"id\":123}", text);
```

按需要修改Value

Alibaba Group

根据Key和Value判断是否序列化

```
class VO {
    public int    id;
    public String name;
}

PropertyFilter filter = new PropertyFilter() {

    public boolean apply(Object source, String name, Object value) {
        return "id".equals(name);
    }
};

VO vo = new VO();
vo.id = 123;
vo.name = "gaotie";

String text = JSON.toJSONString(vo, filter);
Assert.assertEquals("{\"id\":123}", text);
```

按需要修改Value

Alibaba Group

按名称过滤

```
class VO {
    public int getId() { throw new RuntimeException(); }
}

PropertyPreFilter filter = new PropertyPreFilter () {
 boolean apply(JSONSerializer serializer, Object source, String name) {
    return false;
}
};

VO vo = new VO();

String text = JSON.toJSONString(vo, filter);
Assert.assertEquals("{}", text);
```

只按Key过滤，不会调用get方法

Alibaba Group

序列化定制添加内容

```
public abstract class BeforeFilter implements SerializeFilter {
    protected final void writeKeyValue(String key, Object value) {
        // ... ...
    }

    public abstract void writeBefore(Object object);
}
```

```
BeforeFilter filter = new BeforeFilter() {
    @Override
    public void writeBefore(Object object) {
        this.writeKeyValue("id", 123);
        this.writeKeyValue("name", "wenshao");
    }
};

String text = JSON.toJSONString(new Object(), filter);
Assert.assertEquals("{\"id\":123,\"name\":\"wenshao\"}", text);
```

按需要添加内容

Alibaba Group

通过Annotation定制

```
public static class User {
    @JSONField(name="ID")
    private int id;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
}
```

配置在Field上

```
public static class User {
    private int id;

    @JSONField(name="ID")
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
}
```

配置在Getter/Setter上

还可以配置interface的Getter/Setter上

Alibaba Group

支持Proxy对象

- 支持常见的Proxy
  - java.reflect.Proxy
  - cglib Proxy
  - javaassist Proxy
- hibernate对象序列化不会导致拖库

Alibaba Group

自动识别各种日期格式

- ISO-8601日期格式
- yyyy-MM-dd
- yyyy-MM-dd HH:mm:ss
- yyyy-MM-dd HH:mm:ss.SSS
- 毫秒数字
- 毫秒数字字符串
- .NET JSON日期格式
- new Date(198293238)

Alibaba Group

GetOnly反序列化支持

```
public class VO {
  final AtomicInteger       counter = new AtomicInteger();
  final List<Object>        items   = new ArrayList<Object>();
  final Map<String, Object> values  = new HashMap<String, Object>();

  public AtomicInteger getCounter() { return counter; }
  public List<Object> getItems() { return items; }
  public Map<String, Object> getValues() { return values; }
}
```

AtomcXXX、List、Map这几个类型，没有setter也支持反序列化

Alibaba Group

序列化和反序列化相关字段

- 标准JavaBean Getter/Setter (public)
- public field
- get_/set_
- getfXX/setfXX
- 自动识别getter/setter相关的field
  - 标准javaBean的fieldName
  - 下划线_fieldName
  - m_前缀 m_fieldName

Alibaba Group

Stream Writer API

```
JSONWriter writer = new JSONWriter(new FileWriter("/tmp/huge.json"));
writer.startArray();
for (int i = 0; i < 1000 * 1000; ++i) {
      writer.writeValue(new VO());
}
writer.endArray();
writer.close();
```

```
JSONWriter writer = new JSONWriter(new FileWriter("/tmp/huge.json"));
writer.startObject();
for (int i = 0; i < 1000 * 1000; ++i) {
      writer.writeKey("x" + i);
      writer.writeValue(new VO());
}
writer.endObject();
writer.close();
```

支持巨大JSON

Alibaba Group

Stream Reader API

```
JSONReader reader = new JSONReader(new FileReader("/tmp/huge.json"));
reader.startArray();
while(reader.hasNext()) {
      VO vo = reader.readObject(VO.class);
      // handle vo ...
}
reader.endArray();
reader.close();
```

```
JSONReader reader = new JSONReader(new FileReader("/tmp/huge.json"));
reader.startObject();
while(reader.hasNext()) {
      String key = reader.readString();
      VO vo = reader.readObject(VO.class);
      // handle vo ...
}
reader.endObject();
reader.close();
```

支持巨大JSON

Alibaba Group

代码规模和覆盖率

Lines of code

18,593

26,751 lines

10,637 statements

190 files

Classes

198

8 packages

1,042 methods

88 accessors

Unit tests coverage

97.4%

99.0% line coverage

94.1% branch coverage

Unit test success

100.0%

0 failures

0 errors

2,151 tests

14.9 sec

Alibaba Group

大小恰当

Json库	大小
jackson-core-2.2.2.jar jackson-annotations-2.2.2.jar jackson-databind-2.2.2.jar jackson-module-afterburner-0.7.1.jar	188k 33k 864k 111k
fastjson-1.1.33.jar	**350k**
fastjson-1.1.33-android.jar	255k
gson-2.2.4.jar	190k
json-lib-2.4.jar	159k