json

1 概述

> JSON(JavaScript Object Notation)，JS 对象表示法， 是一种轻量级的数据交换格式

在数据传递中，使用XML具有很好的优势，可以跨平台、跨语言，还可以对数据进行很详细的描述，有较好的可读性。但是在XML解析的过程，会变的较为麻烦，并且很多时候效率还很低。

使用Json格式的数据进行传输，就可以很好的解决XML传输中所出现的问题，因为Json 提供了更好的简单性和灵活性。

注意，JSON在JavaScript中，只是用来表示对象的一种方法

例如，

```
var obj = {name:"tom",age:20};
```

> 对象中，key的表示不需要双引号，value值的表示需要双引号，但是value值如果是数字，则不需要双引号

这种对象可以很容易的解析，获取其中的key和value

```
console.log(obj.name);
console.log(obj.age);
```

2 转换

2.1 JavaScript

> 在JavaScript中，将JSON对象转为字符串，将字符串转为JSON对象

例如，JSON对象转为字符串

```
var jsonObj = {name:"tom",age:20};
var jsonStr = JSON.stringify(jsonObj);
```

例如，字符串转为JSON对象

```
var jsonStr = "{\"name\":\"tom\",\"age\":20}";
var jsonObj = JSON.parse(jsonStr);
```

> 注意，JSON格式的字符串中，key和value都需要双引号（一定不能使用单引号），如果value是数组可以不用双引号

例如，JSON数组对象转为字符串

```
var jsonArr = [];
var obj1 = {name:"tom1",age:21};
var obj2 = {name:"tom2",age:22};
jsonArr.push(obj1);
jsonArr.push(obj2);
console.log(JSON.stringify(jsonArr));
```

输出的内容为：

```
[{"name":"tom1","age":21},{"name":"tom2","age":22}]
```

格式化后：

```
[
    {
        "name":"tom1",
        "age":21
    },
    {
        "name":"tom2",
        "age":22
    }
]
```

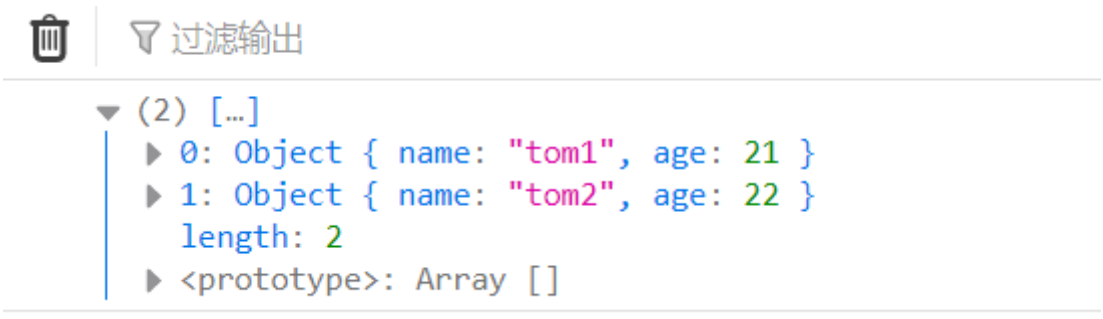
例如，字符串转换JSON数组对象

```
var jsonArrStr = "[{\"name\":\"tom1\",\"age\":21},
{\"name\":\"tom2\",\"age\":22}]";

var jsonArr = JSON.parse(jsonArrStr);

console.log(jsonArr);
```

输出的结果为：



> 注意1，将对象转为字符串的目的是为了，在请求或响应中方便携带数据

> 注意2，将字符串转为对象的目的是为了，在JavaScript或java中方便解析

2.2 java

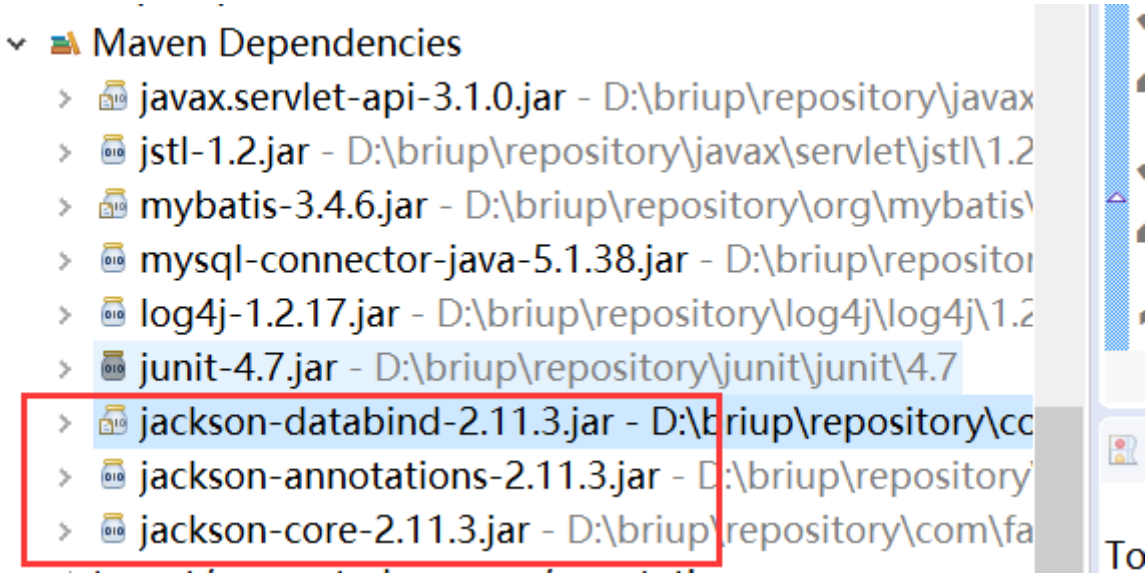
> 在Java中，将JSON对象转为字符串，将字符串转为JSON对象

在java中，完成这个操作，也很容易，有很多第三方jar包都提供了该功能，例如 jackson

1、项目中引入依赖

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.11.3</version>
</dependency>
```

如图：



2、代码中使用

以 `User` 类型为例：

```
@JsonIgnoreProperties({"password"})
public class User{

    private Integer id;
```

```
    private String name;
    @JsonFormat(pattern = "yyyy-MM-dd")
    private Date dob;
    private String password;

    public User() {}

    public User(Integer id, String name, Date dob, String password) {
        this.id = id;
        this.name = name;
        this.dob = dob;
        this.password = password;
    }

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Date getDob() {
        return dob;
    }
    public void setDob(Date dob) {
        this.dob = dob;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", dob=" + dob + ",
password=" + password + "]";
    }

}
```

注意，`@JsonFormat(pattern = "yyyy-MM-dd")` 指定该属性转换json格式字符串的时候的日期格式

注意，`@JsonIgnoreProperties({"password"})` 指定改属性在转json格式字符串的时候可以忽略掉

测试代码：

例如，User对象转换json格式字符串

```
@Test
public void test_obj2json() {
    ObjectMapper mapper = new ObjectMapper();

    User user = new User(1,"tom",new Date(),"123456");

    try {
        String jsonStr = mapper.writeValueAsString(user);
        System.out.println(jsonStr);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }

}
```

输出结果：

```
{"id":1,"name":"tom","dob":"2020-10-29"}
```

格式化：

```
{
    "id":1,
    "name":"tom",
    "dob":"2020-10-29"
}
```

例如，json格式字符串转User对象

```
@Test
public void test_json2Obj() {

    ObjectMapper mapper = new ObjectMapper();

    String jsonStr = "{\"id\":1,\"name\":\"tom\",\"dob\":\"2022-10-29\"}";

    try {
        User user = mapper.readValue(jsonStr, User.class);
        System.out.println(user);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}
```

例如，Map<String,String>对象转json格式字符串

```
@Test
public void test_map2Json_string() {
    ObjectMapper mapper = new ObjectMapper();
```

```
        Map<String, String> map = new HashMap<String,String>();
        map.put("name", "tom");
        map.put("age", "21");

        try {
            String jsonStr = mapper.writeValueAsString(map);
            System.out.println(jsonStr);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

    }
```

输出结果：

```
{"name":"tom","age":"21"}
```

```
{
    "name":"tom",
    "age":"21"
}
```

例如，json格式字符串转Map<String,String>对象

```
    @Test
    public void test_json2Map_string() {
        ObjectMapper mapper = new ObjectMapper();

        String jsonStr = "{\"name\":\"tom\",\"age\":\"21\"}";

        try {
            @SuppressWarnings("unchecked")
            Map<String,String> map = mapper.readValue(jsonStr, Map.class);
            System.out.println(map);

        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

    }
```

例如，map<String,User>对象转Json格式字符串

```
    @Test
    public void test_map2Json_user() {
        ObjectMapper mapper = new ObjectMapper();

        Map<Integer, User> map = new HashMap<>();
        map.put(1, new User(1,"tom1",new Date(),"123456"));
        map.put(2, new User(2,"tom2",new Date(),"123456"));
```

```
        map.put(3, new User(3,"tom3",new Date(),"123456"));

        try {
            String jsonStr = mapper.writeValueAsString(map);
            System.out.println(jsonStr);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

    }
```

输出结果：

```
{"1":{"id":1,"name":"tom1","dob":"2020-10-29"},"2":{"id":2,"name":"tom2","dob":"2020-10-29"},"3":{"id":3,"name":"tom3","dob":"2020-10-29"}}
```

格式化：

```
{
    "1":{
        "id":1,
        "name":"tom1",
        "dob":"2020-10-29"
    },
    "2":{
        "id":2,
        "name":"tom2",
        "dob":"2020-10-29"
    },
    "3":{
        "id":3,
        "name":"tom3",
        "dob":"2020-10-29"
    }
}
```

例如，json格式字符串转Map<String,User>对象

```
@Test
public void test_json2Map_user() {

    ObjectMapper mapper = new ObjectMapper();

    String jsonStr = "{\"1\":{\"id\":1,\"name\":\"tom1\",\"dob\":\"2022-10-29\"},\"2\":{\"id\":2,\"name\":\"tom2\",\"dob\":\"2022-10-29\"},\"3\":{\"id\":3,\"name\":\"tom3\",\"dob\":\"2022-10-29\"}}";

    try {
        //指定要解析转换java类型是Map，已经Map中的俩个泛型的类型分别是String和User
```

```
        JavaType javaType =
mapper.getTypeFactory().constructParametricType(Map.class,
String.class,User.class);
        Map<String, User> map = mapper.readValue(jsonStr, javaType);
        for(User user:map.values()) {
            System.out.println(user);
        }
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }

}
```

例如，List<String> 对象转Json格式字符串

```
@Test
public void test_list2Json_string() {

    ObjectMapper mapper = new ObjectMapper();

    List<String> list = new ArrayList<>();

    list.add("hello");
    list.add("world");
    list.add("briup");

    try {
        String jsonStr = mapper.writeValueAsString(list);
        System.out.println(jsonStr);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }

}
```

输出结果：

```
["hello","world","briup"]
```

例如，Json格式字符串转List<String>对象

```
@Test
public void test_json2List_string() {

    ObjectMapper mapper = new ObjectMapper();

    String jsonStr = "[\"hello\",\"world\",\"briup\"]";

    try {
        @SuppressWarnings("unchecked")
        List<String> list = mapper.readValue(jsonStr,List.class);
```

```
            System.out.println(list);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

    }
```

例如，List<User>对象转Json格式字符串

```
    @Test
    public void test_list2Json_user() {
        ObjectMapper mapper = new ObjectMapper();

        List<User> list = new ArrayList<>();

        list.add(new User(1,"tom1",new Date(),"123456"));
        list.add(new User(2,"tom2",new Date(),"123456"));
        list.add(new User(3,"tom3",new Date(),"123456"));

        try {
            String jsonStr = mapper.writeValueAsString(list);
            System.out.println(jsonStr);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

    }
```

输出结果：

```
[{"id":1,"name":"tom1","dob":"2020-10-29"},{"id":2,"name":"tom2","dob":"2020-10-29"},{"id":3,"name":"tom3","dob":"2020-10-29"}]
```

格式化：

```
[
    {
        "id":1,
        "name":"tom1",
        "dob":"2020-10-29"
    },
    {
        "id":2,
        "name":"tom2",
        "dob":"2020-10-29"
    },
    {
        "id":3,
        "name":"tom3",
        "dob":"2020-10-29"
    }
]
```

例如，json格式字符串转List <User>对象

```
@Test
public void test_json2List_user() {

    ObjectMapper mapper = new ObjectMapper();

    String jsonStr = "[{\"id\":1,\"name\":\"tom1\",\"dob\":\"2022-10-29\"},
{\"id\":2,\"name\":\"tom2\",\"dob\":\"2022-10-29\"},
{\"id\":3,\"name\":\"tom3\",\"dob\":\"2022-10-29\"}]";

    try {
        JavaType javaType =
mapper.getTypeFactory().constructParametricType(List.class, User.class);

        List<String> list = mapper.readValue(jsonStr,javaType);

        System.out.println(list);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }

}
```