[Json对象和Json字符串的区别](https://www.cnblogs.com/wxh0929/p/11132073.html)

**1、Json概念**

* 概念：JSON (JavaScript Object Notation) 一种简单的数据格式，是一种比较轻量级的数据存储格式；
* 优点：
* 缺点：

**2、Json与JS对象的对比**

|  |  |  |
| --- | --- | --- |
| **区**  **别** | **JSON** | **JavaScript** |
| 含    义 | 仅仅是一种数据格式 | 类的实例 |
| 传    输 | 可以跨平台数据传输，速度快 | 不能传输 |
| 表    现 | * 键值对的方式，属性必须加双引号 * 值不可以是方法函数、undefined以及NAN | * 键值对方式，属性不加引号 * 值可以是函数、对象、字符串、数字、boolean 等 |
| 相    互    转    换 | Json字符串转换为JS对象   * JSON.parse(str),不兼容IE7 * eval("(" + str + ")");（兼容所有浏览器，但是不安全，会执行json里面的表达式） | JS对象转换为Json字符串  JSON.stringify(obj) |

**3、Json对象**

* 最显著的特征：对象的值可以用**“对象.属性”**进行访问；
* typeOf(json对象)       ===>    Object类型
* 例子如下：
* var person={"name":"shily","sex":"女","age":"23"}//json对象  
  console.log(person);

console.log(person.name);  
console.log(typeof person);

输出结果是：

**https://img2018.cnblogs.com/blog/1730357/201907/1730357-20190704144459509-462759770.png**

**4、Json字符串（所谓字符串：单引号或者双引号引起来）**

* 例子如下：
* var person='{"name":"shily","sex":"女","age":"23"}';//json字符串
* console.log(person)
* console.log(person.name)

console.log(typeof person)

输出结果是：

https://img2018.cnblogs.com/blog/1730357/201907/1730357-20190704144838907-505782.png

**5、Json对象与Json字符串的相互转化**

* Json字符串转化为Json对象（ JSON.parse() ）
* var str = '{"name":"shily","sex":"女","age":"23"}';
* var strToObj = JSON.parse(str);
* console.log(strToObj);
* console.log(typeof strToObj);

console.log(strToObj.name)

输出结果为：

https://img2018.cnblogs.com/blog/1730357/201907/1730357-20190704145855981-1630181138.png

* Json对象转化为Json字符串（ JSON.stringify() ）
* var obj = {"name":"shily","sex":"女","age":"23"}//json对象
* var objToStr = JSON.stringify(obj);
* console.log(objToStr);

console.log(typeof objToStr)

输出结果为：

https://img2018.cnblogs.com/blog/1730357/201907/1730357-20190704150303585-1564111215.png

**6、Json数据在哪种格式下会发生转换**

* 调用第三方接口返回json字符串，我们自己处理成所需格式的json数据？
* 往JSONObject对象中添加属性及属性值，最后将JSONObjecct对象转成json数据返回？

**7、SpringMvc接受Json字符串类型**

* 基于Rest风格的开发，前端ajax提交的是一个json格式的字符串，后台以@RequestBody接受的json对象

[复制代码](javascript:void(0);)

<script type="text/javascript">

$(document).ready(function(){

var saveDataAry=[];

var data1={"userName":"test","address":"gz"};

var data2={"userName":"ququ","address":"gr"};

saveDataAry.push(data1);

saveDataAry.push(data2);

$.ajax({

type:"POST",

url:"user/saveUser",

dataType:"json",

contentType:"application/json",

data:JSON.stringify(saveData),

success:function(data){

}

});

});

</script>

[复制代码](javascript:void(0);)

上面代码，首先push方法将其封装到数组中，其表现格式：

[

{"userName":"test","address":"gz"},

{"userName":"ququ","address":"gr"}

]

JSON.stringify(saveData)将其转换为json字符串：同时ajax请求的时候也要指定dataType: “json”,contentType:”application/json” 这样就可以轻易的将一个对象或者List传到Java端。

* 后台接收

[复制代码](javascript:void(0);)

@Controller

@RequestMapping(value = "saveUser", method=RequestMethod.POST )

@ResponseBody

public void saveUser(@RequestBody List<User> users) {

userService.batchSave(users);

}

[复制代码](javascript:void(0);)

后台用@RequestBody将其封装到List<User>中。然后进入Service层。

GET、POST方式提时， 根据request header Content-Type的值来判断:

application/x-www-form-urlencoded， 可选（即非必须，因为这种情况的数据@RequestParam, @ModelAttribute也可以处理，当然@RequestBody也能处理）；

multipart/form-data, 不能处理（即使用@RequestBody不能处理这种格式的数据）；

其他格式， 必须（其他格式包括application/json, application/xml等。这些格式的数据，必须使用@RequestBody来处理）。