AJAX

1、基本了解

ajax 不是一个新的技术，它是对现有几种技术的一个整合的应用。

ajax 叫异步的 js 和 xml。

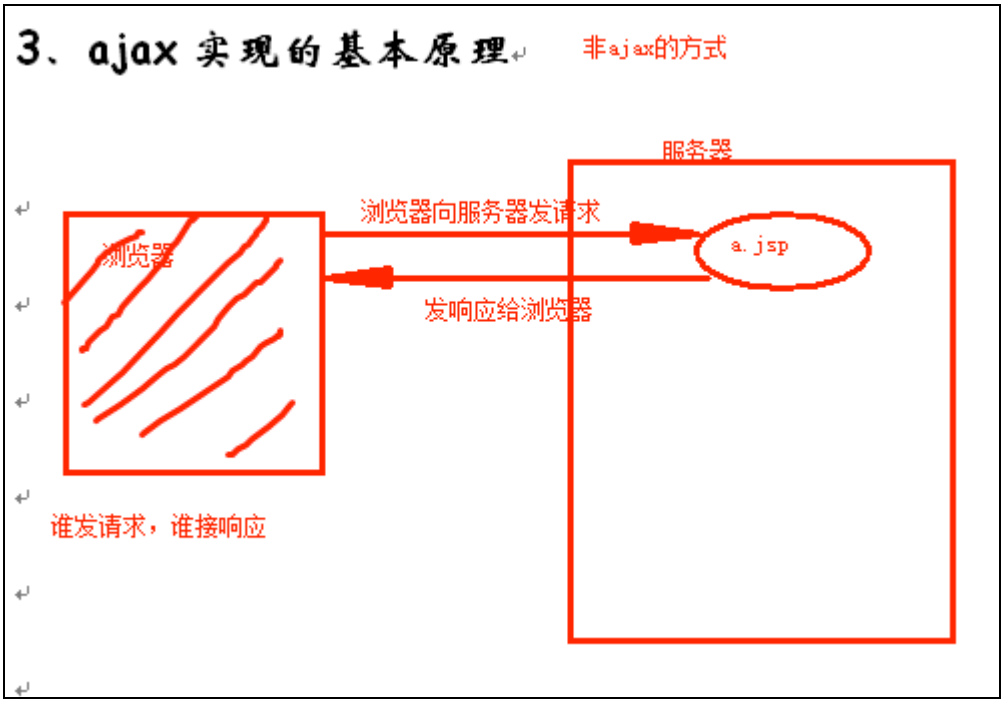
2、作用

异步刷新-局部刷新

视频网站。

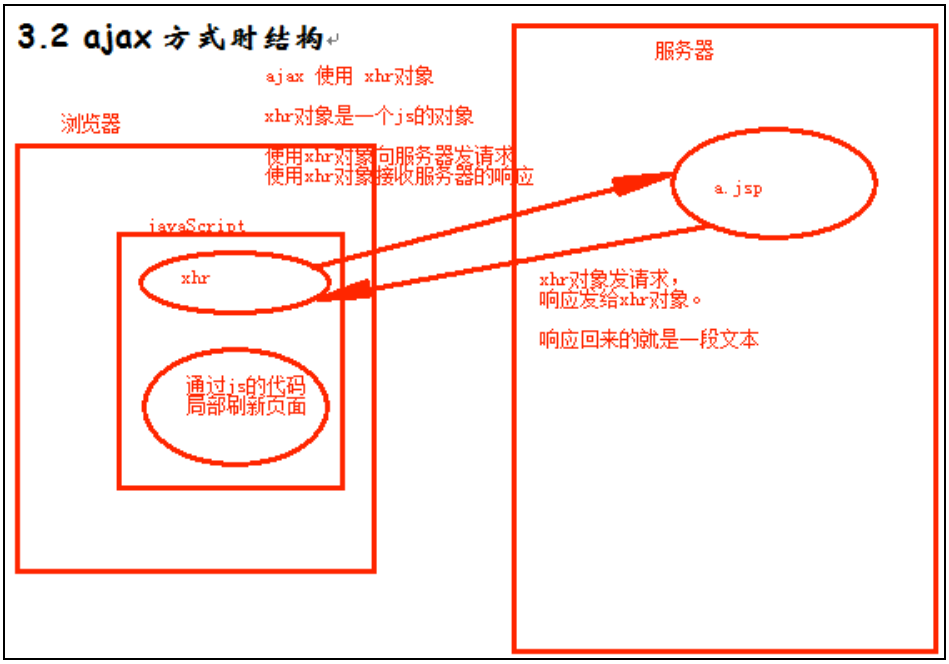
3、ajax 实现的基本原理

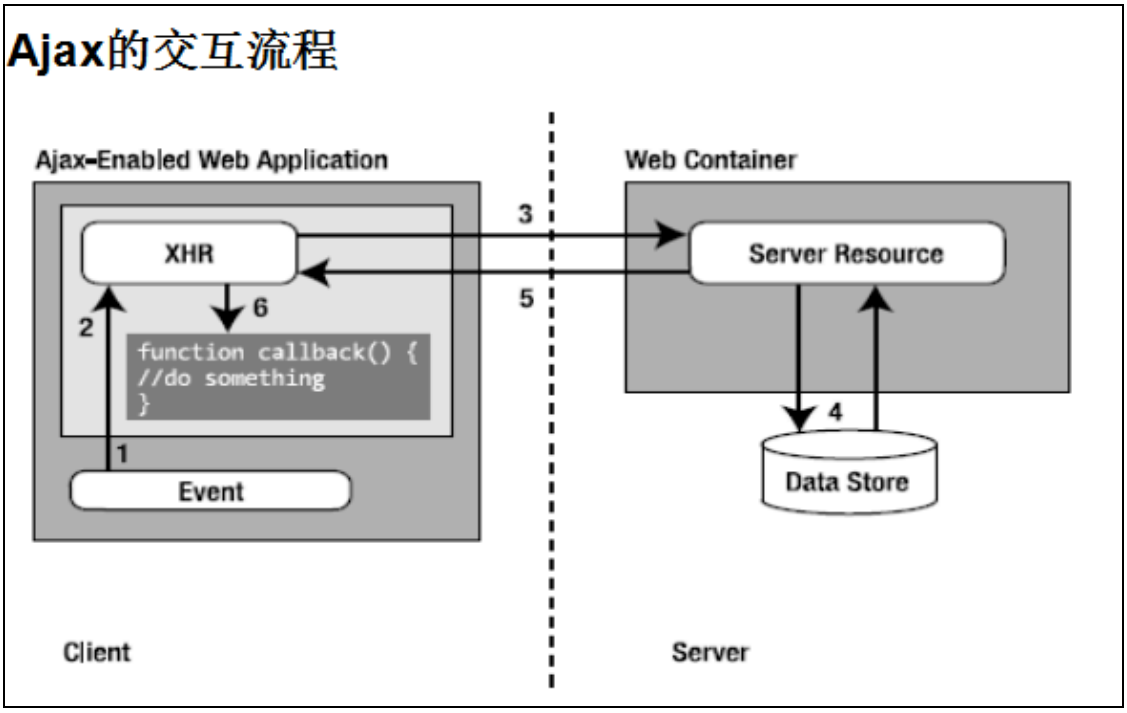
3.1 以前使用非 ajax 方式时的结构



3.2 ajax 方式时结构的基本原理

ajax 方式-异步请求方式实现原理：在客户端通过 js 对象 xhr 向服务器发送请求，xhr 也要接收服务器发送回来的响应（响应就是一段文本）。再通过 js 代码操作 xhr 对象获得的响应文本来实现页面的局部刷新。





4、ajax 是异步刷新技术

异步：二个人手拉手一起走。界面的显示与服务器之间的交互没有影响。

同步：界面显示就不能交互，交互界面就不显示。

5、XHR 对象

5.1 XHR 对象的创建

针对不同的浏览器 XHR 对象的创建方式不同。

5.2 open 方法创建一个请求

open(method, url, asynchronous)

method：请求类型。可以是 get 或 post

url：请求地址。可以使用绝对、相对地址，地址可以附带查询字符串

asynchronous：可选参数，请求同步还是异步。默认为 true（异步）

5.3 send 方法发送一个请求

send(null);

5.4 onreadystatechange 事件

状态改变事件。

5.5 readyState 属性

状态属性 4 表示完成状态，请求发出去并响应接收回来。

我们让 xhr 从创建一个请求到接收响应这整个过程中，xhr 对象会经历好几个状态。

5.6 status 属性

响应的状态属性。

404 没找到 500 服务器错误 200 正确

5.7 responseText 属性

xhr 对象的响应文本。

6、实现异步 ajax 的基本步骤-用户名验证

6.1 第一步：创建 XHR 对象

```
function createXHR() {
    var xhr = null;
    if (window.ActiveXObject) {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    } else if (window.XMLHttpRequest) {
        xhr = new XMLHttpRequest();
    } else {
        alert("can't create xhr object!");
    }
    return xhr;
}
```

6.2 第二步：创建事件源并调用验证的函数

```
function checkusername(username){
    var value = username.value;
    alert(value);
}
<input type="text" name="username" value="" onblur="checkusername(this)">
```

6.3 第三步：接收请求的 jsp 页面-checkname.jsp

```
<body>
  This is my CHECKNAME.JSP page. <br>
</body>
```

6.4 第四步：实现 ajax 异步发送请求

```
function checkusername(username){
```

```
        var value = username.value;
        //通过xhr向checkname.jsp页面发送请求
        var xhr = createXHR();
        //1 创建一个请求的连接。
        xhr.open("get", "checkname.jsp");
        //2 走你
        xhr.send(null);
    }
```

6.5 checkname.jsp 页面响应是什么

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">
  </head>
  <body>
    <% System.out.println("checkname.jsp页面被请求了一次！！！！"); %>
    This is my CHECKNAME.JSP page. <br>
  </body>
</html>
```

当前页面的响应：

```
1
2
3 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
4 <html>
5   <head>
6     <base href="http://192.168.7.199:80/ajax0511/">
7   </head>
8
9   <body>
10
11    This is my CHECKNAME.JSP page. <br>
12  </body>
13 </html>
14
```

6.6 第五步：实现 ajax 异步请求的响应接收

```
var xhr = createXHR();
function checkusername(username){
    var value = username.value;
    //通过 xhr 向 checkname.jsp 页面发送请求
    //1 创建一个请求的连接。
    xhr.open("get", "checkname.jsp");
    //2 走你
    xhr.send(null);
    //3 为 xhr 对象的状态改变事件注册回调函数叫 rollback.
    xhr.onreadystatechange = rollback;
}
//因为这个函数是在 xhr 的状态改变时调用的函数，叫"回调函数"
function rollback(){
    if(xhr.readyState == 4){
        if(xhr.status == 200){
            alert(xhr.responseText);
        }
    }
}
```

```
1
2
3 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
4 <html>
5   <head>
6     <base href="http://192.168.7.199:80/ajax0511/">
7   </head>
8
9   <body>
10
11    This is my CHECKNAME.JSP page. <br>
12  </body>
13 </html>
14
```

6.7 第六步：实现用户名验证

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%><%@page
import="java.net.URLDecoder"%><%
    String username = URLDecoder.decode(
            request.getParameter("username"), "utf-8");
    if ("admin".equals(username)) {
%>true<%
```

```
    } else {
%>false<%
    }
%>
```

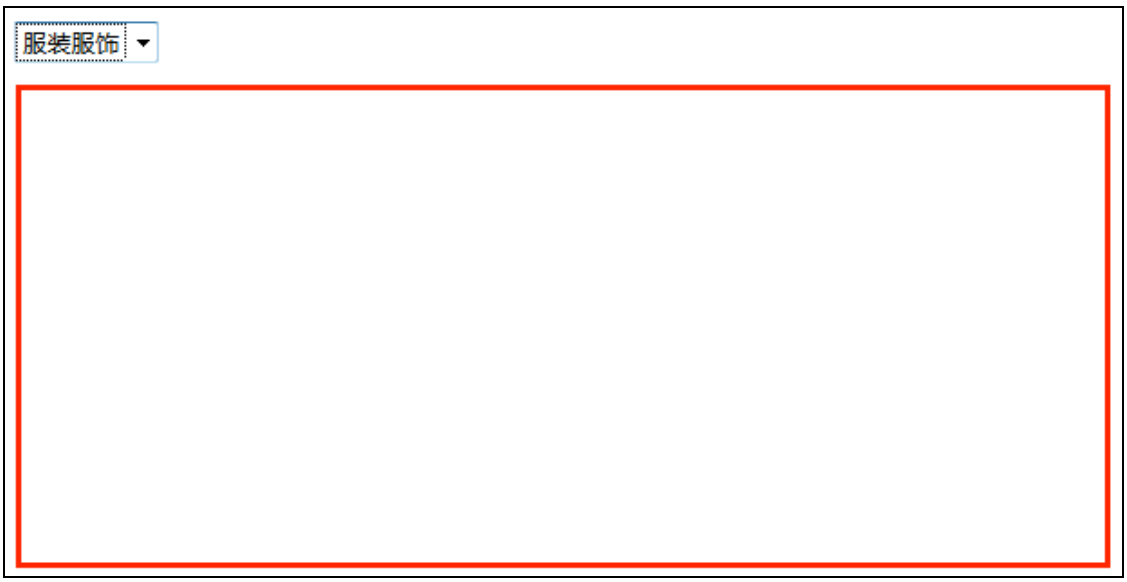
```
    var xhr = createXHR();
    function checkusername(username){
        var value = username.value;
        //通过 xhr 向 checkname.jsp 页面发送请求
        //1 创建一个请求的连接。
        xhr.open("get",       encodeURI("checkname.jsp?username="+value+"&time="+new
Date()) );
        //2 走你
        xhr.send(null);
        //3 为 xhr 对象的状态改变事件注册回调函数叫 rollback.
        xhr.onreadystatechange = rollback;
    }
    //因为这个函数是在 xhr 的状态改变时调用的函数，叫"回调函数"
    function rollback(){
        if(xhr.readyState == 4){
            if(xhr.status == 200){
                //alert(xhr.responseText);
                var responseText = xhr.responseText;
                if(responseText == "true"){
                document.getElementById("span1").innerHTML="<font color=red>用户名不
可用！ </font>";
                }else{
                document.getElementById("span1").innerHTML="<font color=green>用户名
可用！ </font>";
                }
            }
        }
    }
```

6.8 checkname.jsp 页面换成 checkname.do 的 Servlet

```
@WebServlet("/checkname.do")
public class CheckUserNameServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("CheckUserNameServlet 执行！ ");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String username = URLDecoder.decode(request.getParameter("username"),
```

```
                    "utf-8");
            if ("管理员".equals(username)) {
                out.print(true);
            } else {
                out.print(false);
            }
            out.flush();
            out.close();
        }
}
```

7、使用 ajax 实现查询的局部刷新




```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">

<script type="text/javascript">
    function createXHR() {
        var xhr = null;
```

```
        if (window.ActiveXObject) {
            xhr = new ActiveXObject("Microsoft.XMLHTTP");
        } else if (window.XMLHttpRequest) {
            xhr = new XMLHttpRequest();
        } else {
            alert("can't create xhr object!");
        }
        return xhr;
    }
    var xhr = createXHR();
    function fun() {
        xhr.open("get",
                "${pageContext.request.contextPath}/findgoods.do?goodsType="
                        + document.getElementById("mysql").value);
        xhr.send(null);
        xhr.onreadystatechange = rollback;
    }
    function rollback() {
        if (xhr.readyState == 4) {
            if (xhr.status == 200) {
                var responseText = xhr.responseText;
                mydiv.innerHTML = responseText;
            }
        }
    }
    function loadfun() {
        fun();
    }
</script>
</head>
<body onload="loadfun()">
    <select id="mysql" name="goodsType" onchange="fun()">
        <option value="1">家用电器</option>
        <option value="2">数码产品</option>
    </select>
    <br>
    <div id="mydiv"></div>
</body>
</html>
```

```
@SuppressWarnings("serial")
@WebServlet("/findgoods.do")
public class FindGoods extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        int goodsType = Integer.parseInt(request.getParameter("goodsType"));
        List<Goods> goodsList = new ArrayList<Goods>();
        switch(goodsType){
        case 1:
            goodsList.add(new Goods(1, "手电筒", 100));
            goodsList.add(new Goods(2, "电视", 200));
            break;
        case 2:
            goodsList.add(new Goods(3, "HTC", 1000));
            goodsList.add(new Goods(4, "APPLE", 2000));
            break;
        }
        request.setAttribute("goodsList", goodsList);
        request.getRequestDispatcher("/showgoods.jsp").forward(request, response);

    }


}
```

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<table>
    <tr>
        <th>编号</th>   <th>名称</th>   <th>价格</th>
    </tr>
    <c:forEach items="${requestScope.goodsList }" var="goods">
        <tr>
            <td>${goods.goodsId }</td>
            <td>${goods.goodsName }</td>
            <td>${goods.goodsPrice }</td>
        </tr>
    </c:forEach>
</table>
```

8、使用 jQuery 的 ajax

8.1 使用 jQuery 先导入

8.2 可以使用三种方式实现 ajax 操作

8.2.1 $.ajax()

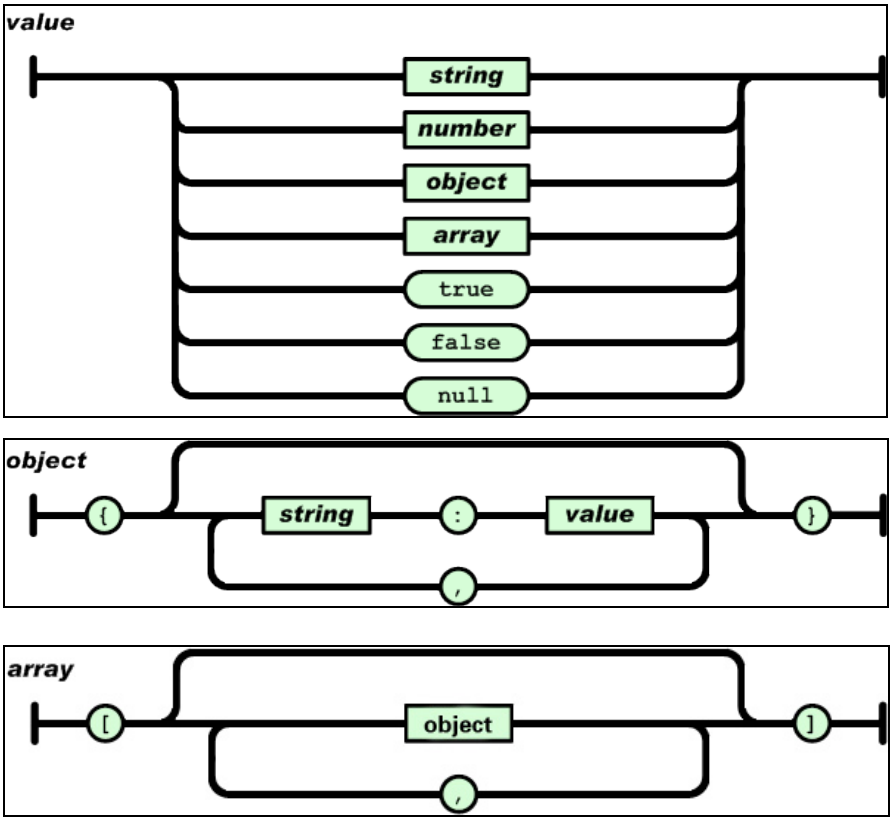
```
<script type="text/javascript" src="js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    $(function() {
        $("#mysql").change(function() {
            $.ajax({
                type : "GET",
                url : "findgoods.do",
                data : "goodsType="+$("#mysql").val(),
                dataType : "html",
                success : function(data) {
                    $("#mydiv").html(data);
                }
            });
        });
        $("#mysql").change();
    });
</script>
</head>
<body>
    <select id="mysql" name="goodsType">
        <option value="1">家用电器</option>
        <option value="2">数码产品</option>
    </select>
    <br>
    <div id="mydiv"></div>
```

9、JSON

JSON：JavaScript 对象表示法（JavaScript Object Notation）

在客户端与服务器之间进行数据传递。

10、JSON 格式



json 表示数据时，都是 key-value 的方式

格式： { "key":"value" , "key":"value" }

案例：var jsonobj = {"name":"peter","age":25};

使用时，直接使用 json 对象打点访问属性。

输出的 name 属性的值 ：alert(jsonobj.name);

```
<script type="text/javascript">
    var jsonobj = {"name":"peter","age":25};
    alert(jsonobj.name);
    var jsonarr = [ {"name":"sun","age":25} , {"name":"xu","age":25} , jsonobj];
    for(var i = 0 ; i < jsonarr.length ;i++){
        alert(jsonarr[i].name);
    }
    var jsonobj1 = {"name":"peter","birthday":{"year":1990,"month":1,"day":1} }
    alert(jsonobj1.birthday.year);
```

```
        var jsonobj2 = {"name":"peter","isLogin":true};
        if( true == jsonobj2.isLogin){
            alert(jsonobj2.isLogin);
        }
    </script>
```

11、JSON 对象与 JSON 文本

JSON 对象： var jsonobj = {"name":"peter","age":25};

JSON 文本： var jsontxt = "{"name":"peter","age":25}";

一个满足 JSON 对象格式的文本字符串。

请求/响应只能是文本格式。

以响应方式发送的内容也一定是文本格式。

所以当我们使用 ajax 时，接收的都是 JSON 文本。

怎么将一个 json 文本转换成一个 json 对象。

使用到一个叫 eval()的 javascript 函数。

var jsonobj = eval("("+jsontxt+")");

12、在 JAVA 类中使用 JSON

能 java 对象转换成 json

能 java 集合转换成 json

12.1 导入第三方 jar 文件

12.2 将 java 对象转换成了 json 文本

```
//创建一个 java 对象
Goods goods = new Goods(1, "HTC", 10.5);
```

```
//使用 josn 对象
JSONObject jsonobj = JSONObject.fromObject(goods);
//转换成 json 文本
String jsontxt = jsonobj.toString();
System.out.println(jsontxt);
```

12.3 将 json 文本转换成 java 对象

```
String jsontxt1 = "{\"goodsId\":2,\"goodsName\":\"KFC\",\"goodsPrice\":18.8}";
JSONObject jsonobj1 = JSONObject.fromObject(jsontxt1);
Goods goods1 = (Goods) JSONObject.toBean(jsonobj1, Goods.class);
System.out.println(goods1.getGoodsName());
```

12.4 将 java 集合转换成 json 文本

12.5 将 json 文本转换成 java 集合

```
List<Goods> goodsList = new ArrayList<Goods>();
goodsList.add(new Goods(1, "HTC", 10.5));
goodsList.add(new Goods(2, "KFC", 18.8));

//将 java 集合转换成 json 文本
JSONArray jsonarray = JSONArray.fromObject(goodsList);
String jsontxt = jsonarray.toString();
System.out.println(jsontxt);

List<Goods> goodsList1 = (List<Goods>) JSONArray.toList(jsonarray, Goods.class);
for (Goods goods : goodsList1) {
    System.out.println(goods.getGoodsName());
}
```

13 JsonConfig

在进行转换时配置的对象。

```
        Goods goods1 =  new Goods(1, "HTC", 10.5);
        Goods goods2 = new Goods(2, "KFC", 18.8);
        Types type = new Types(1, "食品");
        goods1.setGoodsType(type);
        goods2.setGoodsType(type);
        type.getGoodsList().add(goods1);
```

```
        type.getGoodsList().add(goods2);

        JsonConfig jsonConfig = new JsonConfig();
        jsonConfig.setExcludes(new String[]{"goodsList"});
        JSONObject typejsonobj = JSONObject.fromObject(type,jsonConfig);
        System.out.println(typejsonobj.toString());
        JSONObject goodsjsonobj = JSONObject.fromObject(goods1,jsonConfig);
        System.out.println(goodsjsonobj.toString());
```

jsonConfig.setExcludes(new String[]{"goodsList"}); 表示不包含字符串数组中属性。

14 案例：8、使用 jQuery 的 ajax 和 JSON

14.1 编写 Servlet，返回 JSON 文本

```
@WebServlet("/findgoodsbyjson.do")
public class FindGoodsByJSON extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setCharacterEncoding("utf-8");
        PrintWriter out =    response.getWriter();

        int goodsType = Integer.parseInt(request.getParameter("goodsType"));
        List<Goods> goodsList = new ArrayList<Goods>();
        switch(goodsType){
        case 1:
            goodsList.add(new Goods(1, "手电筒", 100));
            goodsList.add(new Goods(2, "电视", 200));
            break;
        case 2:
            goodsList.add(new Goods(3, "HTC", 1000));
            goodsList.add(new Goods(4, "APPLE", 2000));
            break;
        }

        JSONArray jsonarray = new JSONArray();
        String jsontxt = JSONArray.fromObject(goodsList).toString();
        out.write(jsontxt);
        out.flush();
        out.close();
    }
```

14.2 在 JSP 页面上使用 jQuer 发送 ajax 请求，并接收 json.

```
<script type="text/javascript">
    $(function() {
        $("#mysql").change(function() {
            $.ajax({
                type : "GET",
                url : "findgoodsbyjson.do",
                data : "goodsType="+$("#mysql").val(),
                dataType : "json",
                success : function(data) {
                    for(var i = 0 ;i <data.length;i++){
                        alert(data[i].goodsName);
                    }
                }
            });
        });
        $("#mysql").change();
    });
</script>
```

14.3 通过 JS 脚本刷新页面

```
<table width="400px" >
<thead>
    <tr>
        <th>编号</th><th>名称</th><th>价格</th>
    </tr>
</thead>
<tbody id="databody" ></tbody>
</table>
```

```
success : function(data) {
    var mydatabody = document.getElementById("databody");
    mydatabody.innerHTML="";
    for(var i=0;i<data.length;i++){
        var mytr = mydatabody.insertRow();
        var idtd = mytr.insertCell();
        var nametd = mytr.insertCell();
        var pricetd= mytr.insertCell();
        idtd.innerHTML=data[i].goodsId;
        nametd.innerHTML = data[i].goodsName;
        pricetd.innerHTML = data[i].goodsPrice;
    }
```

```
}
```