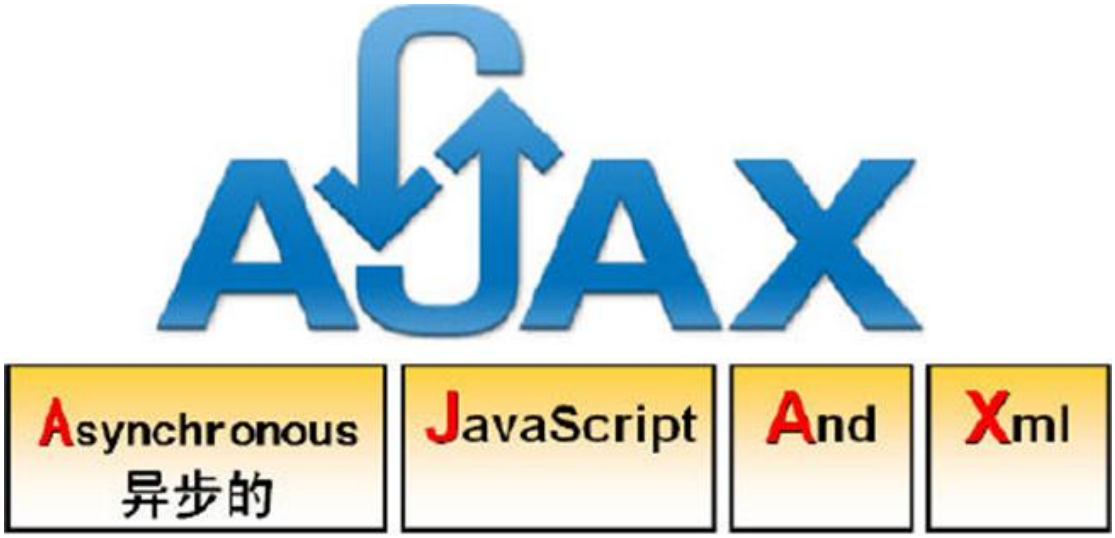
第五讲 AJAX

一、Ajax 的工作流程

1. 什么是 Ajax

Ajax：异步刷新技术



打开百度，打开 chrome 浏览器的开发工具窗口

点击 network

进入百度首页，先把之前的请求全部清空，

在搜索框中输入人工智能，你会发现，你没有做任何的点击按钮的操作，它就已经向服务器发送请求了

并且页面中的内容，也开始刷新了，很显然这种刷新不是页面中所有的内容都刷新，如果是页面中所有内容都刷新，就没办法在文本框中输入内容了

ajax 提供了一个新的请求发送的方式
之前已经给大家介绍了几种请求发送发送方式，一个是页面的提交按钮和超链接
另外一个呢就是通过 js 中 form 对象的 submit 方法向服务器端提交请求
这里是第三种方式，并且和以前的请求有区别，以前的两种请求发送方式，叫做同步请求，ajax 提供的请求发送方式叫做异步请求

ajax 除了提供一种新的请求发送方式以外，还提供了一种新的接收服务器响应内容的方式
之前接收服务器的响应结果，是不做任何操作的，因为服务器是把整个页面全部返回到浏览器，浏览器直接显示，我们不需要在这个过程中进行任何操作
而现在服务器返回的是页面的部分内容，接收后，需要自己将这部分内容，在页面上展现出来

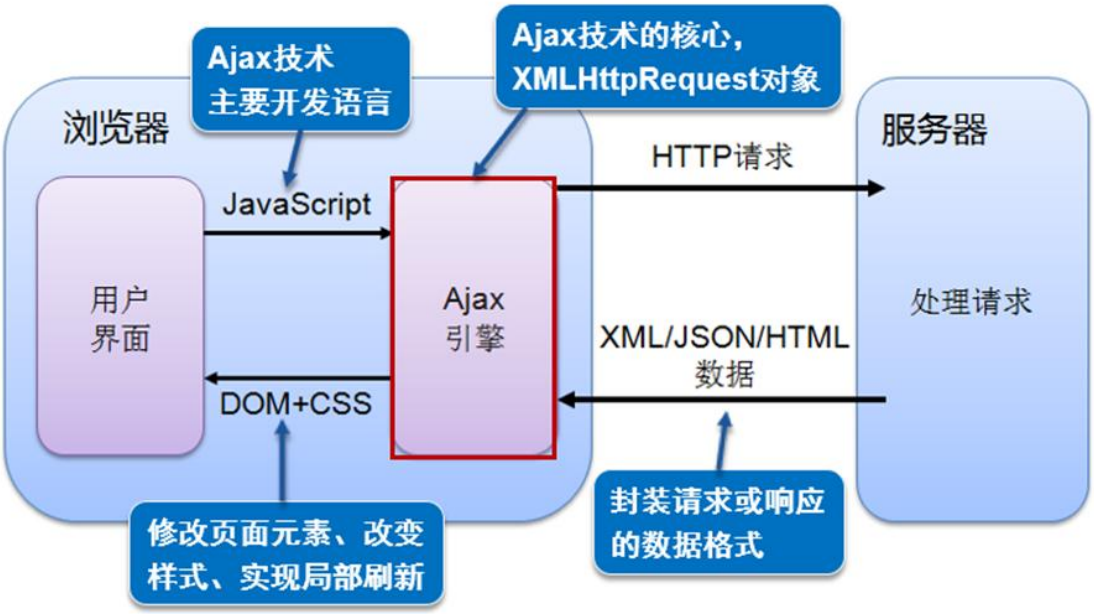
那有同学说，这样做的好处是啥呢
传统的 web 技术，发送的请求是同步请求，拿到的是完整页面，那么只有等到页面所有元素显示完毕后，我们才能继续操作，所以用户操作并不连贯，体验差
有了 ajax 之后，我们的请求是异步请求，拿到的内容是页面中部分需要更新的信息，所以并不需要不断的刷新页面，用户甚至感觉不到页面发送了多少请求，拿到了多少响应数据，操作具有连贯性，用户体验高

差异	方式	说　　明
发送请求方式不同	传统 Web	浏览器发送同步请求
	Ajax 技术	异步引擎对象发送请求
服务器响应不同	传统 Web	响应内容是一个完整页面
	Ajax 技术	响应内容只是需要的数据
客户端处理方式不同	传统 Web	需等待服务器响应完成并重新加载整个页面后用户才能进行操作
	Ajax 技术	可以动态更新页面中的部分内容

		不影响用户在页面进行其他操作

2. Ajax 的工作流程

发送异步请求，接收页面的需要进行更新的内容，这个流程是怎么完成的呢



异步请求的发送，还是通过 JavaScript 完成，这次使用的是一个叫做 XMLHTTPRequest 对象，向服务器端发送请求

服务器端接收到请求之后，返回响应信息，和以前不同，以前是返回整个页面的内容，现在返回的大多数是 json 格式的对象，字符串等等

浏览器接收到这部分数据后，依然使用 javascript，去动态的改变页面的内容，这部分操作就是之前我们学过的使用 javascript 操作 dom 对象，什么添加一个元素，删除元素，更改元素属性等操作

这就是我们 ajax 的工作流程

在来看看这个流程，寻找其中的关键点

可以看到重点并不在服务器端

重点在浏览器这一部分，它有两个任务需要完成

第一得知道如何利用 xmlhttpRequest 对象发送请求

第二得知道服务器端什么时候把数据返回到浏览器

至于把数据放到页面上，刚才也说了，就是 javascript 的 dom 操作

二、Ajax 的核心对象——XMLHttpRequest

整个 Ajax 技术的核心
提供异步发送请求的能力

1. 案例

创建工程：AjaxPrj
添加页面 a.html，以及处理异步请求的 Servlet -- TestServle
打开 ajax 的代码模板，直接把代码复制进去

首先要关注的就是请求的发送，需要用到的对象是 XMLHTTPRequest，
看下创建的对象 createXmlHttpRequest
这部分代码一般没什么变化，你可以认为就是固定写法
主要考虑的是不同浏览器版本，在实例化 XMLHTTPRequest 对象的方式使不同的，照着这个写，就没事

有了对象之后，要设置对象的属性，并发送请求了
首先这里设置了一个回调函数，就是服务器返回响应信息后，得调用哪个方法去处理这个返回数据，这里就是调用 callback 方法
接下来通过 open 方法，传递请求方式和请求的路径，第三个参数直接写 true 就可以了，不用多管，
有了请求的路径和方式等信息之后，就可以发送请求了
调用的是 XMLHttpRequest 对象的 send 方法

当服务器的响应内容到达客户端后，调用 callback 方法处理
处理的时候需要注意，第一是怎么知道响应内容已经全部到达浏览器了，XMLHttpRequest 提供了一个属性 readyState，如果属性值是 4，代表服务器端的响应内容全部发送完毕，到达客户端了
接下来，得区分一下，这个响应信息的信息是不是我们想要的，或者说服务器端是不是正确响应了我们的请求，这里使用的是前面说过的状态码，状态码为 200 代表服务器端返回的是正确响应
紧接着要取出响应的内容，使用 xmlHttpRequest.responseText 可以获得服务器端返回的内容，获得的是一个文本形式的数据，这里直接用 alert 语句把它输出出来

完成服务器端 servlet 的编写

服务器端处理用户请求的代码没有区别，还是通过 request 对象的 getParameter 方法获取请求参数的值

需要注意的是，这里的响应就不能是页面跳转了，如果是页面跳转你返回的就是页面的全部内容，而不是部分内容，就做不到局部刷新的效果了，所以这里的响应方式得用 io 流对象，把数据以文本字符串的方式，返回给客户端

```
String value = request.getParameter("key");
PrintWriter writer = response.getWriter();
writer.print(value);
writer.flush();
writer.close();
```

2. XMLHttpRequest——方法

方　　法	说　　明
open(String method, String url,boolean async)	创建一个新的 HTTP 请求
send(String data)	发送请求到服务器端
setRequestHeader(String header, String value)	设置请求的某个 HTTP 头信息
getResponseHeader(String header)	获取响应的指定 HTTP 头信息

3. XMLHttpRequest——事件

事件 onreadystatechange：指定回调函数

相关属性 readyState：XMLHttpRequest 的状态信息

就绪状态码	说　　明
0	XMLHttpRequest 对象未完成初始化
1	XMLHttpRequest 对象开始发送请求
2	XMLHttpRequest 对象的请求发送完成
3	XMLHttpRequest 对象开始读取响应

4	XMLHttpRequest 对象读取响应结束

4. XMLHttpRequest——属性

- status：HTTP 的状态码

状态码	说　　明
200	服务器正确返回响应
404	请求的资源不存在
500	服务器内部错误

- statusText：返回当前请求的响应状态
- responseText：以文本形式获得响应的内容
- responseXML：将 XML 格式的响应内容解析成　DOM 对象返回

5. GET 请求和 POST 请求的区别

步　骤	请求方式	实　现　代　码
初始化组件	GET	xmlHttpRequest.open("GET", url, true);
	POST	xmlHttpRequest.open("POST", url, true); xmlHttpRequest .setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
发送请求	GET	xmlHttpRequest.send(null);
	POST	xmlHttpRequest.send("key=xxx&type=12&year=2018");

三、应用 jquery 简化 Ajax 实现

1. 使用$.ajax()

```
$.ajax( {
        "url"    :   "url",                          // 要提交的 URL 路径
        "type" :   "get",                          // 发送请求的方式
        "data"    :   data,                          // 要发送到服务器的数据
        "dataType" :   "text",                       // 指定传输的数据格式
        "success"    :   function(result) {          // 请求成功后要执行的代码
             },
         "error"    :   function() {                // 请求失败后要执行的代码
             }
} );
```

参　数	类　　型	说　　明
url	String	发送请求的地址，默认为当前页地址
type	String	请求方式，默认为 GET
data	PlainObject 或 String 或 Array	发送到服务器的数据
dataType	String	预期服务器返回的数据类型，包括：XML、HTML、Script、JSON、JSONP、text
success	Function (jqXHR jqxhr, PlainObject settings)	发送请求前调用的函数
error	Function(任意类型 result, String textStatus,jqXHR jqxhr)	请求成功后调用的函数 参数 result：可选，由服务器返回的数据

举例：

新建文件夹 js

复制 jq1.8

新建页面 jq1.html

引入 jq <script type="text/javascript" src="js/jquery-1.8.2.min.js"></script>

添加文本框，取 id 属性:txt

<input id="txt" type="text" value="" />

编写 jquery 代码，测试引入成功

```
<script type="text/javascript">
    $(function(){
        $("#txt").blur(function(){
            alert("ok");
        });
    })
</script>
```

复制 ppt 上的模板，添加 ajax 方法

```
    $(function() {
        $("#txt").blur(function() {
            var value = this.value;
            $.ajax({
                "url" : "TestServlet", //要提交的 URL 路径
                "type" : "GET", //发送请求的方式
                "data" : "key=" + value, //要发送到服务器的数据
                "dataType" : "text", //指定返回的数据格式
                "success" : callBack, //响应成功后要执行的代码
                "error" : function() { //请求失败后要执行的代码
                    alert("出现错误");
                }
            });

            //响应成功时的回调函数
            function callBack(data) {
                alert(data);
            }//end of callBack()
        });
    })
```

2. $.get()用法

```
$.get( url, data, function( result ) {
    // 省略将服务器返回的数据显示到页面的代码
} );
```

复制 jq1.html 将 ajax 方法换成 get 方法

```
<script type="text/javascript">
    $(function() {
        $("#txt").blur(function() {
            var value = this.value;
            $.get("TestServlet", "key="+value, callBack);

            //响应成功时的回调函数
            function callBack(data) {
                alert(data);
            }//end of callBack()
        });
    })
</script>
```

3. $.post()用法

```
$.post( url, data, function( result ) {
    // 省略将服务器返回的数据显示到页面的代码
} );
```

举例:

复制 jq1.html 将 ajax 方法换成 post 方法

```
    $(function() {
        $("#txt").blur(function() {
            var value = this.value;
            $.post("TestServlet", "key="+value, callBack);
```

```
            //响应成功时的回调函数
            function callBack(data) {
                alert(data);
            }//end of callBack()
        });
    })
```

四、Json 对象

1. 认识 JSON

- JSON（JavaScript Object Notation）
  通常用于在客户端和服务器之间传递数据
- 定义 JSON 对象
  var JSON 对象 = { "name" : value, "name" : value, …… };
- 定义 JSON 数组
  var JSON 数组 = [value, value, ……];

2. 举例：

新建一个 json.html 页面

引入 jquery <script type="text/javascript" src="js/jquery-1.8.2.min.js"></script>

先定义一个 json 格式的对象

```
$(function() {
    //1、定义 JSON 格式的 user 对象，并在 div 中输出
var user = {
        "id" : 1,
        "name" : "张三",
        "pwd" : "000"
    };
```

在页面中设置打印 json 值的 div

一、JSON 格式的 user 对象:

回到 js，使用对象名.属性名的方式，取出 json 对象的属性
然后再 div 中打印出来

```
$("#objectDiv").append("ID：" + user.id + "<br>")
                .append("用户名：" + user.name + "<br>")
                .append("密码：" + user.pwd + "<br>");
```

好，我们完成了 json 对象的处理

接下来，看下 json 数组怎么用
json 中使用中括号来表示一个数组，数组的每个元素的值用逗号分隔
先定义一个简单数组看下，数组里面的元素都是字符串

```
var ary = [ "中", "美", "俄" ];
```

在页面上，能够数组的数据一般展现在 ul 无序列表，下拉列表框，还有表格
这里是字符串数组，一般就显示在 ul 和 select 标签中
所以我们在页面上定义两个标签，用于输出 json 数据

```
JSON 格式的字符串数组:&amp;nbsp;&amp;nbsp;<select id="arraySel"></select>
<ul id="arrayUl"></ul>
```

我们把 dom 对象改成 jq 对象，接下来就可以使用数组的迭代方法了

```
var $ary = $(ary);
var $ul = $("#arrayUl"); //  展示数据的 ul 元素
var $sel = $("#arraySel"); //  展示数据的 select 元素
$ary.each(function() { $ul.append("<li>"+this+"</li>"); });
$ary.each(function(i) {
        $sel.append("<option value='"+(i+1)+"'>"+this+"</option>");
});
```

最后写个复杂点的，就是定义一个 json 数组，数组中的元素是 json 对象

```
//3、定义 JSON 格式的 user 对象数组，并使用<table>输出
    var userArray = [ {
        "id" : 2,
        "name" : "admin",
        "pwd" : "123"
    }, {
        "id" : 3,
        "name" : "詹姆斯",
```

```
        "pwd" : "11111"
    }, {
        "id" : 4,
        "name" : "梅西",
        "pwd" : "6666"
    } ];
```

这种情况就适用于 table 输出

在 html 中定义一个 div 用于显示表格

三、JSON 格式的 user 对象数组:

定义一个 table

```
var $table = $("<table border='1'></table>").append(
            "<tr><td>ID</td><td>用户名</td><td>密码</td></tr>");
```

向 table 中添加数据

```
    $(userArray).each(function() {
        $table.append("<tr><td>" + this.id + "</td><td>"
                    + this.name + "</td><td>"
                    + this.pwd + "</td></tr>");
    });
    $("#objectArrayDiv").append($table);
```

五、FastJSON 组件

1. FastJSON 简介

- 一个性能很好的、Java 实现的 JSON 解析器和生成器
- 将 Java 对象序列化成 JSON 字符串
- 将 JSON 字符串反序列化得到 Java 对象　https://github.com/alibaba/fastjson/releases

2. FastJSON API

(入口类：com.alibaba.fastjson.JSON)方　　法	说　　明

public static String toJSONString (Object object)	将 Java 对象序列化成 JSON 字符串
public static String toJSONString (Object object, boolean prettyFormat)	prettyFormat 为 true 时生成带格式的 JSON 字符串
public static String toJSONString (Object object, SerializerFeature… features)	可以通过参数 features 指定更多序列化规则
public static String toJSONStringWithDateFormat (Object object, String dateFormat, SerializerFeature… features)	可以通过参数 dateFormat 指定日期类型的输出格式

FastJSON API——SerializerFeature

枚 举 值	说 明
QuoteFieldNames	为字段名加双引号，默认即使用
WriteMapNullValue	输出值为 null 的字段，默认不输出
WriteNullListAsEmpty	将值为 null 的 List 字段输出为[ ]
WriteNullStringAsEmpty	将值为 null 的 String 字段输出为 “”
WriteNullNumberAsZero	将值为 null 的数值字段输出为 0
WriteNullBooleanAsFalse	将值为 null 的 Boolean 字段输出为 false
SkipTransientField	忽略 transient 字段，默认即忽略
PrettyFormat	格式化 JSON 字符串，默认不格式化

举例：

添加 JsonServlet

修改 Order 类，只保留 id 和 userAddress 和 createTime

回到 JsonServlet 将 Order 对象转换成 json 对象

```
Order order1 = new Order(1,null,new Date());

String orderJson = JSON.toJSONStringWithDateFormat(order1, "yyyy-MM-dd",
SerializerFeature.WriteNullStringAsEmpty);

PrintWriter writer = response.getWriter();
```

```
writer.println(orderJson);

System.out.println(orderJson);

writer.flush();
writer.close();
```

复制 jq2.html 修改代码

```
    $(function() {
        $("#txt").blur(function() {
            $.get("JsonServlet", "", callBack,"json");

            //响应成功时的回调函数
            function callBack(data) {
                alert(data.id + "    " + data.userAddress + "    " + data.createTime);
            }//end of callBack()
        });
    })
```

3. $.getJSON()用法

```
$.getJSON( url, data, function( result ) {
    // 省略将服务器返回的数据显示到页面的代码
} );
```

举例：

复制 jq4，修改 get 方法为 getJSON 方法，删除参数 json

```
    $(function() {
        $("#txt").blur(function() {
            $.getJSON("JsonServlet", "", callBack,);

            //响应成功时的回调函数
            function callBack(data) {
                alert(data.id + "    " + data.userAddress + "    " + data.createTime);
```

```
        }//end of callBack()
    });
})
```

如果这里的 order 不止一个，而是一个集合呢，又该怎么处理

回到 JsonServlet，添加一个 Order 的 List 集合

转换成 json 格式的数组

String orderListJson = JSONArray.toJSONStringWithDateFormat(orderList, "yyyy-MM-dd", SerializerFeature.WriteNullStringAsEmpty);

复制页面，打印输出 json 数组

```
$(function() {
    $("#txt").blur(function() {
        $.getJSON("JsonServlet", "", callBack);

        //响应成功时的回调函数
        function callBack(data) {
            $(data).each(function(){
                alert(this.id+"     "+this.userAddress+"    "+this.createTime);
            });

        }//end of callBack()
    });
})
```

六、使用 Ajax 优化购物街注册页面

进入注册页面, 我们都知道注册用户的用户名是不能重复的, 希望在用户填写完用户名之后, 就能够立刻检查出用户名是否可用，这个该怎么做呢,

思路是这样子的，当用户填写完用户名后，光标自然要移出用户名文本框，所以这里就可以有一个文本框失去焦点事件

当事件发生后，向服务器发送异步请求，服务器检查是否有重名，并把结果返回给页面，页面根据服务器的返回值，决定在页面上显示什么样的值

所以我们先添加一个 servlet，用于验证用户名是否重名

添加 Servlet -- ValidateLoginNameServlet

```
String result = "true";
String loginName = request.getParameter("loginName");
if(loginName!=""&&loginName!=null){
    if(loginName.equals("aaa")){   // 此处应该查数据库
        result = "false";
    }
}

PrintWriter writer = response.getWriter();

writer.print(result);

writer.flush();
writer.close();
```

回到注册页面

用户名文本框上添加 id="loginName"

再给 loginName 所在的 td 加上一个 id="tdLoginName"

我们要把报错信息显示在这个单元格中

```
    $(function() {
        $("#loginName").blur(function() {
            var loginName = this.value;
            $("#tdLoginName div").remove();
            if(loginName != ""){
                $.get("ValidateLoginNameServlet",  "loginName="+loginName,  callBack);

                //响应成功时的回调函数
                function callBack(data) {
                    if(data == "false"){
                        $("#tdLoginName").append("<div><span style='color:red'>用户名已经被注册</span></div>");
```

```
                    }
                }//end of callBack()
            }else{
                $("#tdLoginName").append("<div><span style='color:red'>用户名不能为空
</span></div>");
            }
        });
    })
```