2 AJAX 技术

2.1 AJAX 基础

工作目标

知识目标

- 理解 AJAX 的概念
- 理解 AJAX 执行过程
- 掌握 XMLHttpRequest 对象相关方法的使用

技能目标

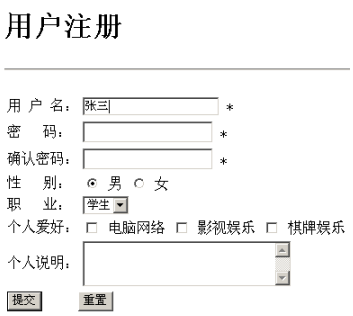
- 在 Web 页面中使用 AJAX 技术进行用户名唯一性的验证

素养目标

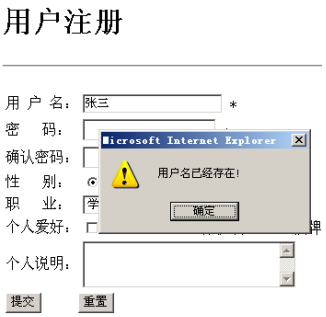
- 培养学生的动手和自学能力

工作任务

在注册页面中增加对输入的用户名的唯一性验证，要求使用 AJAX 技术，当光标离开用户名输入框的时触发该验证；为简化业务逻辑，当输入用户名张三的时候用户名不唯一，输入其他用户名则验证通过。注册的界面如图 2.1-1（a）所示，当用户输入用户名“张三”后切换光标到其他输入框，结果如图 2.1-1（b）所示，当用户输入用户名“李四”后切换光标到其他输入框，结果如图 2.1-1（c）所示。



（a）注册用户名唯一验证-用户名输入



（b）注册用户名唯一验证-验证不通过

图 2.1-1

（c）注册用户名唯一验证-验证不通过

图 2.1-1（续图）

工作计划

任务分析之问题清单

1．AJAX 是什么？

2．如何实现 AJAX？

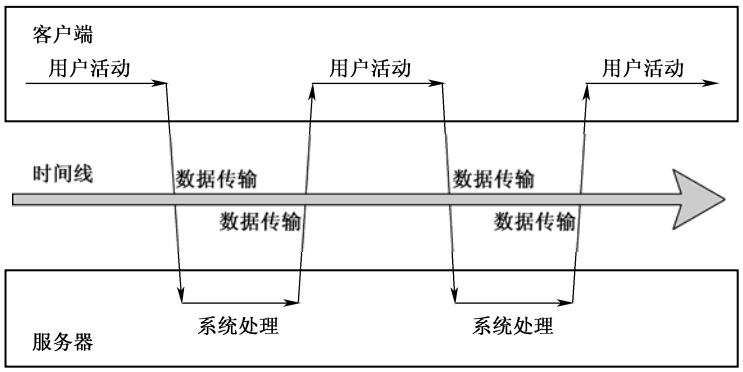
任务解析

1．关于 AJAX

AJAX 是网页可以异步提交的一种技术。

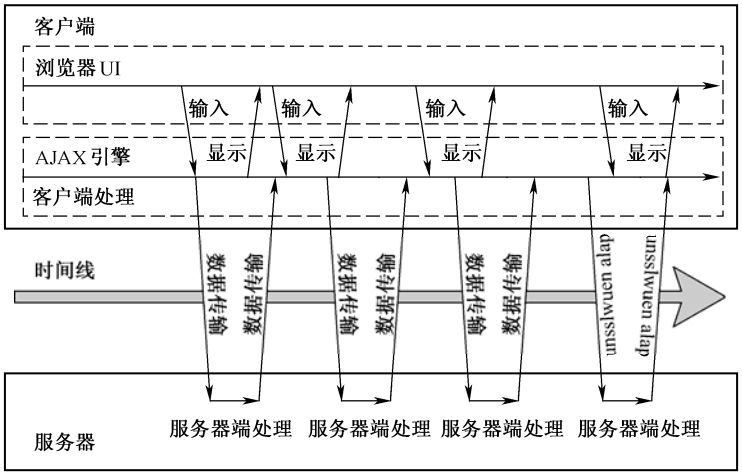
异步提交的解释：异步是相对于同步（如图 2.1-2（a）所示）而言的，一般的网页，在提交到后台处理的时候是整个页面都提交的，在提交到后台进行处理的过程中，用户除了等待结果以外什么也做不了。异步提交表现在于页面并不是全部提交的，只是局部提交，在局部提交进行后台处理的过程中，用户在页面没有进行提交的部分还可以操作，无需等待局部提交的完成，这样提高了用户的操作效率（如图 2.1-2（b）所示）。

AJAX 技术的实现原理是使用 XMLHttpRequest 对象来实现。



（a）经典 Web 应用模型（同步）

图 2.1-2



（b）AJAX Web 应用模型（异步）

图 2.1-2（续图）

XMLHttpRequest 对象是系统对象，它的作用是：负责局部提交到后台；负责跟踪后台的执行情况，并获得后台的执行结果；当后台执行完毕后，负责返回到前台调用用户定义的 javascript 函数进行善后操作。

2．实现 AJAX 技术

AJAX 技术运行的一般程序流程：

- 前面页面触发事件调用 JavaScript 函数（通常自定义的函数）
- 在 JavaScript 函数中编写代码创建 XMLHttpRequest 对象，提交部分数据到后台进行处理
- 后台中使用 Java 代码完成指定的业务逻辑并输出结果
- XMLHttpRequest 获得后台执行的结果，及时调用程序员定义的 javascript 响应函数进行善后工作

根据这个流程，下面以一个“用户你好”的小例子来说明整个过程的代码编写。

【例 2.1-1】制作一个输入用户名的界面（界面中有一个用户名输入框和确定按钮）（如图 2.1-3（a）所示）；当用户输入用户名之后单击“确定”按钮，弹出一对话框提示“xxx，你好！”（xxx 为用户输入的用户名）（如图 2.1-3（b）所示）。在整个过程中要求使用 AJAX 技术局部提交到后台（后台处理：仅仅在控制台输出用户名），而整个页面不能提交（不能跳转页面）。

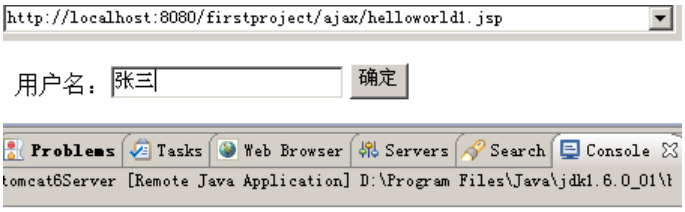
步骤 1：根据流程“前面页面触发事件调用 JavaScript 函数”，编写本例子的页面（一个输入框和一个按钮）；触发事件为单击按钮的 onclick 事件，关键代码如下：

```
<form name="f1" action="表单提交地址" method="post">
  用户名：<input type="text" name="username">
  <input type="button" value="确定" onclick=" sendhelloworld(f1.username. value); ">
</form>
```

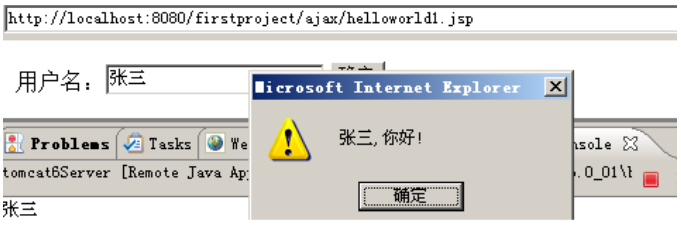
代码中 onclick="sendhelloworld(f1.username.value);"的 sendhelloworld(f1.username.value)就是本例要自定义的 JavaScript 函数。我们需要通过此函数实现 AJAX。

步骤 2：根据流程“在 JavaScript 函数中编写代码创建 XMLHttpRequest 对象，提交部分数据到后台 Struts2 的 action 进行处理”，我们首先定义一个全局变量 XMLHttpReq，接着定义一个名为 sendhelloworld()的函数，参数有一个，为需要传值的用户名，关键代码如下：

```
<script language="javascript">
  var xmlHttpReq;//全局变量，是 XMLHttpRequest 类型的对象
     function sendhelloworld(name) {
     }
</script>
```



（a）用户你好-输入用户名



（b）用户你好-提交结果

图 2.1-3

然后，在函数中编写代码创建 XMLHttpRequest 对象，关键代码如下：

```
//下面代码含义：判断浏览器类型，根据不同浏览器调用系统函数创建 XMLHttpRequest 的对象到 xmlHttpReq。代码固定，照搬即可
if (window.XMLHttpRequest) {
        xmlHttpReq = new XMLHttpRequest();
        if (xmlHttpReq.overrideMimeType) {
                xmlHttpReq.overrideMimeType("text/xml");
        }
} else if (window.ActiveXObject) {
        try {
                xmlHttpReq = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
                xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
        }
}
if (!xmlHttpReq) {
        window.alert("该浏览器不支持 XMLHttpRequest！");
        return;
}
```

XMLHttpRequest 对象 xmlHttpReq 创建完毕后，接下来就是使用对象 xmlHttpReq 的相关方法进行提交，具体提交过程有三个小步骤。

①设置提交参数。调用 XMLHttpRequest 对象 open 方法来设置提交参数。语法格式：

```
XMLHttpRequest 对象.open("提交方式", "提交地址");
```

参数说明：提交方式取值为 get 或 post 与表单提交方式相同，为避免乱码，建议用 post；提交地址是 action 的名称。本例的代码如下：（其中 helloworld1 是 action 的名称）

```
xmlHttpReq.open("POST", "helloworld1");
```

②指定响应函数。给 XMLHttpRequest 对象的 onreadystatechange 属性指定响应函数。语法格式：

```
XMLHttpRequest 对象.onreadystatechange =  响应函数名;
```

响应函数是我们一般需要自定义的一个 JavaScript 函数，它会在程序提交到后台执行 action 完毕后被 XMLHttpRequest 对象自动调用执行，该函数是为了进行 AJAX 处理完毕后的善后工作。本例的代码如下：（响应函数的名字取名为 callbackhelloworld）

```
xmlHttpReq.onreadystatechange = callbackhelloworld;
```

③进行提交。使用 XMLHttpRequest 对象的 send 方法进行提交。语法格式：

```
xmlHttpReq.send("参数名 1=参数值 1&参数名 2=参数值 2&…参数名 n=参数值 n");
```

send 方法里边的参数是前台传递给后台的数据，以参数名=参数值的形式给出，多个数据之间用&符号隔开。

附加的一个小步骤：中文乱码的处理。由于 send 方法中可以传递数据到后台，如果数据中有中文，则需要增加对中文编码格式的处理语句，该语句设置提交的数据格式，必须在提交的代码之前加入：（代码固定不变，照搬即可）

```
xmlHttpReq.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```

最后，本例第三小步骤代码如下，将用户名作为参数传递到后台，参数名为 username。

```
xmlHttpReq.send("username=" + name);
```

步骤 3：根据流程“后台中使用 Java 代码完成指定的业务逻辑并输出结果”编写后台代码。后台代码的业务逻辑是根据参数名 username 获得前台提交的用户名数据，然后打印到控制台，然后将获得的用户名再返回给前台。

编写 action 执行业务逻辑，并使用通过 Struts2 提供的 ServletActionContext 类获得能响应信息给客户端的 PrintWriter 对象，并通过 print 方法将处理结果返回给前台，关键代码如下（类名为 HelloWorld1Action，方法名为 execute）：

```
String username;
public String execute() throws Exception {
        //编码转换
        String str=new String(username.getBytes("ISO8859_1"),"UTF-8");
        System.out.println(str);
        //输出信息：获得 response 对象
        HttpServletResponse response=ServletActionContext.getResponse();
        //设置 response 的编码格式
        response.setCharacterEncoding("UTF-8");
        //得到输出对象 pw
        PrintWriter pw=response.getWriter();
        //使用输出对象 pw 输出信息
        pw.print(str);
        return null;
}
public String getUsername() {
        return username;
}
```

同时在 struts.xml 中声明该 action，配置关键代码如下：

```
<action name="helloworld1" class="ajax.HelloWorld1Action">
</action>
```


注意

因为在 action 的方法中已经调用了 response 对象来将处理后的信息从服务器端返回给客户端，所以在 action 方法中也没有必要再返回一个字符串对象给 struts2 框架让它做页面跳转（示范代码中返回的就是 null）。同时在 struts.xml 里该 action 的配置中,也不需要对该 action 配置任何 result 项，Ajax 是异步传输，配置中加上 result 项是没有意义的。

步骤 4：根据流程“XMLHttpRequest 获得后台执行的结果，及时调用程序员定义的 javascript 响应函数进行善后工作”，首先定义一个在步骤 2 的第二个小步骤所指定的响应函数，本例为 callbackhelloworld，关键代码如下：

```
function callbackhelloworld() {
}
```

然后在函数 callbackhelloworld 中使用 XMLHttpRequest 对象的 readyState 属性判断后台是否执行（取值为 4 表示执行完毕），使用 XMLHttpRequest 对象的 status 属性判断执行是否成功（取值为 200 表示执行成功），关键代码如下：

```
if (xmlHttpReq.readyState == 4) { //判断对象状态
        if (xmlHttpReq.status == 200) { //信息已经成功返回，开始处理信息
        //善后工作的代码在此处编写
      }else { //页面不正常
              window.alert("您所请求的页面有异常。");
        }
}
```

接着，在判断后台执行成功之后的 if 语句里边使用 XMLHttpRequest 对象的 responseText 属性获得后台返回的结果（得到的是一个字符串），关键代码如下：

```
var res=xmlHttpReq.responseText;
```

最后，根据得到的结果，在该 if 语句中编写善后工作的代码，本例是弹出提示“用户 xxx，你好”，关键代码如下：

```
if (res.length>0){
       alert(res+",你好!");
}
```

到此，整个 AJAX 技术的实现过程就完成了。本例运行的效果如图 2.1-3（a）、2.1-3（b）所示。

工作实施

实施方案

1．编写注册界面及输入框失去焦点的触发事件代码

2．编写使用 AJAX 方式提交的 JavaScript 函数

3．编写后台处理代码及相关配置

4．编写进行善后工作的 JavaScript 响应函数

详细步骤

1．在用户名输入框中添加失去焦点的触发事件代码

在用户名输入框中加入失去焦点的触发事件（onblur 事件），触发自定义函数 submitAjax，使用 this.value 传入当前输入框的值，关键代码如下：

```
<s:textfield name="name" onblur="submitAjax(this.value)"></s:textfield>
```

2．编写使用 AJAX 方式提交的 JavaScript 函数

我们首先定义一个全局变量 XMLHttpReq，接着定义一个名为 submitAjax ()的函数，参数为需要验证的用户名，关键代码如下：

```
<script language="javascript">
        var xmlHttpReq;//定义 XMLHttpRequest 类型的对象 XMLHttpReq
    function submitAjax(name) {
    }
</script>
```

然后，在函数中编写代码创建 XMLHttpRequest 对象，关键代码如下：

```
//下面代码含义：判断浏览器类型，根据不同浏览器调用系统函数创建 XMLHttpRequest 的对象到 XMLHttpReq。代码固定，照搬即可
if (window.XMLHttpRequest) {
        xmlHttpReq = new XMLHttpRequest();
        if (xmlHttpReq.overrideMimeType) {
            xmlHttpReq.overrideMimeType("text/xml");
        }
} else if (window.ActiveXObject) {
        try {
            xmlHttpReq = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
        }
}
if (!xmlHttpReq) {
        window.alert("该浏览器不支持 XMLHttpRequest！");
        return;
}
```

接下来就是使用对象 xmlHttpReq 的相关方法进行提交，具体提交过程有三个小步骤。

步骤 1：设置提交参数。调用 XMLHttpRequest 对象的 open 方法来设置提交参数。关键代码如下：（其中 isExist 是提交后台的地址）

```
xmlHttpReq.open("POST", "isExist");
```

步骤 2：指定响应函数。关键代码如下：（响应函数的名字取名为 test）

```
xmlHttpReq.onreadystatechange = test;
```

步骤 3：进行提交与中文乱码处理。关键代码：（传入用户名，参数取为 name）

```
xmlHttpReq.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xmlHttpReq.send("name=" + name);
```

3．编写后台处理代码及相关配置

编写后台 action，在 UserIsExistAction 类中添加用户名的成员变量，并生成 get 及 set 方法。然后添加执行业务逻辑的 userIsExist 方法，在方法中判断前台传入的用户名是否为张三，若为张三则返回字符串 true，否则返回字符串 false，并使用 PrintWriter 对象返回执行结果给前台，关键代码如下：

```
public class UserIsExistAction extends ActionSupport {
private String name; //会员登录名
        public String userIsExist() throws Exception{
            //获得 response 对象
            HttpServletResponse response = ServletActionContext.getResponse();
            //设置 response 的编码格式
            response.setCharacterEncoding("UTF-8");
            //得到输出对象 pw
            PrintWriter pw = response.getWriter();
            //使用输出对象 pw 输出信息
```

```
            // (下面只是举例用户名张三已经存在进行验证，实际中需连接数据库数据进行验证)
            if(this.getName().equals("张三")){
                pw.write("true");
            }else{
                pw.write("false");
            }
            pw.close();
            return null;
        }
        // 省略 name 的 get 及 set 方法
}
```

然后在 struts.xml 文件中编写 action 配置，注意类的包路径，关键代码如下：

```
<action name="isExist"
        class="com.zdsoft.action.UserIsExistAction" method="userIsExist">
</action>
```

4．编写 JavaScript 响应函数进行善后工作

首先定义一个响应函数 test，关键代码如下：

```
function test() {
}
```

然后在函数 test 中使用 XMLHttpRequest 对象的 readyState 属性判断后台是否执行（取值为 4 表示执行完毕），使用 XMLHttpRequest 对象的 status 属性判断执行是否成功（取值为 200 表示执行成功），关键代码如下：

```
if (xmlHttpReq.readyState == 4) { //判断对象状态
        if (xmlHttpReq.status == 200) { //信息已经成功返回，开始处理信息
        //善后工作的代码在此处编写
      }else { //页面不正常
            window.alert("您所请求的页面有异常。");
        }
}
```

接着，在判断后台执行成功之后的 if 语句里边使用 XMLHttpRequest 对象的 responseText 属性获得后台返回的结果（得到的是一个字符串），关键代码如下：

```
var returnParam = xmlHttpReq.responseText;
```

最后，根据得到的结果，在该 if 语句中编写善后工作的代码（提示用户名存在或不存在），关键代码如下：

```
if(returnParam=="true"){
        alert('用户名已经存在!');
}else{
        alert('用户名不存在，可以使用!');
}
```

到此，整个任务就完成了。运行的效果如图 2.1-1（a）、2.1-1（b）所示。

2.2 DWR 框架

工作目标

知识目标

- 理解 DWR 框架的概念
- 理解 DWR 框架的执行过程

- 掌握 DWR 框架的基本使用方法

技能目标

- 在 Web 页面中使用 DWR 技术进行用户名唯一性的验证

素养目标

- 培养学生的动手和自学能力

工作任务

与上一节相同，在注册页面中增加对输入的用户名的唯一性验证，要求使用 AJAX 技术，当光标离开用户名输入框的时触发该验证；为简化业务逻辑，当输入用户名张三的时候用户名不唯一，输入其他用户名则验证通过。注册的界面如图 2.1-1（a）所示，当用户输入用户名“张三”后切换光标到其他输入框，结果如图 2.1-1（b）所示，当用户输入用户名“李四”后切换光标到其他输入框，结果如图 2.1-1（c）所示。

工作计划

任务分析之问题清单

1．DWR 框架是什么？

2．如何使用 DWR 框架？

任务解析

1．关于 DWR 框架

DWR（Direct Web Remoting）是一个用于改善 Web 页面与 Java 类交互的远程服务器端 AJAX 开源框架，可以帮助开发人员开发包含 AJAX 技术的网站。它可以允许在浏览器里的代码使用运行在 Web 服务器上的 Java 类的方法，感觉 Java 类的方法就像在浏览器里一样。

2．使用 DWR 框架

DWR 框架的一般使用流程如下：

1）搭建 DWR 开发环境

2）前面页面触发事件调用 JavaScript 函数

3）后台中使用 Java 代码完成指定的业务逻辑并输出结果

4）配置 dwr.xml 文件，将后台 Java 代码登记到 DWR 框架

5）在页面导入 DWR 核心库及自定义接口函数库

6）在 JavaScript 函数中编写 DWR 方式的代码提交部分数据到后台进行处理

7）获得后台结果后及时调用程序员定义的 JavaScript 响应函数进行善后工作

根据这个流程，下面以一个“用户你好”的例子来说明整个过程的代码编写。

【例 2.2-1】与例 2.1-1 功能相同，制作一个输入用户名的界面（界面中有一个用户名输入框和确定按钮）（如图 2.1-3（a）所示）；当用户输入用户名之后单击“确定”按钮，弹出一对话框提示“xxx，你好！”（xxx 为用户输入的用户名）（如图 2.1-3（b）所示）。在整个过程中要求使用 AJAX 技术局部提交到后台（后台处理：仅仅在控制台输出用户名），而整个页面不能提交（不能跳转页面）。

步骤 1：搭建 DWR 开发环境。

①在项目中添加 DWR 提供的第三方开发包，即在项目的 WEB-INF\lib 文件夹加入 dwr.jar 文件。

注意 在加入 dwr.jar 包的同时要加入 commons-logging.jar 包。

②在 web.xml 文件中加入 DWR 的 servlet 的配置，关键代码如下：

```
<!-- Ajax 框架的配置 引入 DWR 的 servlet -->
    <servlet>
        <servlet-name>dwr-invoker</servlet-name>
        <servlet-class>
            org.directwebremoting.servlet.DwrServlet
        </servlet-class>
            <!-- 指定处于开发阶段的参数 -->
            <init-param>
                <param-name>debug</param-name>
                <param-value>true</param-value>
            </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>dwr-invoker</servlet-name>
        <url-pattern>/dwr/*</url-pattern>
    </servlet-mapping>
<!-- Ajax 框架的配置结束 引入 DWR 的 servlet -->
```

③在 webroot\WEB-INF 文件夹中创建一个 dwr.xml 文件，该文件用于将后台的 Java 代码登记到 DWR 框架中，让其能被 DWR 框架正确定位并加以调用。为了检验 xml 的准确性加入 dtd，同时完成基本结构，即加入 DWR 及 allow 节点的配置，关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 2.0//EN"
    "http://getahead.org/dwr/dwr20.dtd">
<dwr>
<allow>
        <!—此处为 dwr 的具体配置项目 -->
    </allow>
</dwr>
```

步骤 2：编写本例子的页面（一个输入框和一个按钮）；触发事件为点击按钮的 onclick 事件，关键代码如下：

```
<form name="f1" action="表单提交地址" method="post">
    用户名：<input type="text" name="username">
    <input type="button" value="确定" onclick=" sendHelloWorldDwr (f1.user name.value); ">
</form>
```

代码中 onclick=" sendHelloWorldDwr (f1.username.value);"的 sendhelloworld(f1.username.value) 就是本例要自定义的 JavaScript 函数。我们需要通过此函数调用 DWR 方式的 js 代码实现 AJAX。

步骤 3：后台中使用 Java 代码完成指定的业务逻辑并输出结果。

后台代码的业务逻辑是根据参数名 name 获得前台提交的用户名数据，然后打印到控制台，然后将获得的用户名再返回给前台。

编写一个普通的 Java 类，在方法中接收前台页面传过来的字符串，进行字符串拼接后返回一个新的字符串对象。关键代码如下（包名为 ajax，类名为 HelloWorldDwrBean，方法名为 helloDwr）：

```
public class HelloWorldDwrBean {
    public String helloDwr(String username) {
        // 程序员在此写各自的业务逻辑，可以是访问数据库的复杂代码
```

```
            return username + "，你好！";
        }
    }
```

步骤 4：配置 dwr.xml 文件，将后台 Java 代码登记到 DWR 框架。

在 dwr.xml 文件中的<allow>节点中加入后台代码的配置，关键属性以下两点：

①<create>节点中的 JavaScript 属性，可以理解为将后台 Java 类取了一个别名，然后在前台的 JavaScript 代码中，便可以通过该别名访问到该类的方法。

②<para>节点中的 value 属性，该属性需配置后台 Java 类的包路径及类名，即告知 DWR 框架后台的 Java 类的所在位置，以便 DWR 框架能正确调用到后台逻辑处理代码。

配置关键代码如下：

```
<allow>
    <create creator="new" javascript="HelloWorldDwr">
        <param name="class" value="ajax.HelloWorldDwrBean" />
    </create>
</allow>
```

步骤 5：在页面导入 DWR 核心库及自定义接口函数库。

DWR 是一个 AJAX 框架，在使用时需导入它所提供的一些 JavaScript 函数库，其中最基本的是 engine.js，路径是在/工程名/dwr/下，同时还需要导入自定义接口函数库，格式为/工程名/dwr/interface/名称.js，该名称和上一步骤中所提到的 Java 类的别名相同，即<create>节点中的 JavaScript 属性（本例为 HelloWorldDwr），注意自定义接口函数库的路径是在/工程名/dwr/interface/下，和 engine.js 的路径不同。参考代码如下：

```
<script type='text/javascript' src='/工程名/dwr/engine.js'></script>
<script type='text/javascript' src='/工程名/dwr/interface/HelloWorldDwr.js'>
```

注意

engine.js 和 HelloWorldDwr.js 两个 js 文件并不需要我们编写，也不用拷贝到工程中，这两个文件是在运行的过程中由 DWR 框架自动生成的。

步骤 6：在 JavaScript 函数中编写 DWR 方式的代码提交部分数据到后台进行处理。

我们首先定义名为 sendHelloWorldDwr ()的函数，参数有一个，为需要传值的用户名，关键代码如下：

```
<script language="javascript">
    function sendHelloWorldDwr (name) {
    }
</script>
```

然后，在函数中编写 DWR 方式的代码，通过在 dwr.xml 文件中声明的后台 Java 类的别名 HelloWorldDwr，调用该类的逻辑处理方法 helloDwr，传递用户名至后台，并指定返回时的响应函数为 callBackHelloWorldDwr，关键代码如下：

```
<script type="text/javascript">
    function sendHelloWorldDwr(name){
        //方法有两个参数，第一个是向后台传的参数，第二个是指定返回时的响应函数
      HelloWorldDwr.helloDwr(name,callBackHelloWorldDwr);
    }
</script>
```

步骤 7：获得后台结果后及时调用程序员定义的 JavaScript 响应函数进行善后工作。

DWR 框架在成功地执行后台逻辑代码并获得执行的结果后，会调用响应函数来完成一次完整

的 AJAX 调用。

定义一个在步骤 5 中所指定的响应函数，有一个参数 data，该参数即后台返回的结果，该方法名及参数列表的定义即 callbackhelloworld(data)，在方法体中定义的是得到后台返回结果后进行善后处理的 JavaScript 代码，本例是直接将后台的返回结果，通过警告框显示给用户。最终前台 JavaScript 的完整代码如下：

```
<script type="text/javascript">
    function sendHelloWorldDwr(name){
        //方法有两个参数，第一个是向后台传的参数，第二个是指定返回时的响应函数
      HelloWorldDwr.helloDwr(name,callBackHelloWorldDwr);
    }
    function callBackHelloWorldDwr(data){
      //通过 data 参数得到后台处理的结果，下面就是我们自己写的善后工作代码
      alert(data);
    }
</script>
```

到此，整个使用 DWR 框架完成一个 AJAX 应用的实现过程就完成了。本例调试运行的效果如图 2.1-3（a）、2.1-3（b）所示。

工作实施

实施方案

1．搭建 DWR 开发环境，配置 dwr.xml 文件

2．在页面导入 DWR 核心库及自定义接口函数库

3．编写注册界面及输入框失去焦点的触发事件代码

4．编写使用 AJAX 方式提交的 JavaScript 函数

5．编写后台处理代码

6．编写 JavaScript 响应函数进行善后工作

详细步骤

1．搭建 DWR 开发环境，配置 dwr.xml 文件

① 在项目的 WEB-INF 文件夹加入 dwr.jar 文件。

② 在 web.xml 文件中加入 DWR 的 servlet 的配置。关键代码如下：

```
<!-- Ajax 框架的配置  引入 DWR 的 servlet -->
    <servlet>
        <servlet-name>dwr-invoker</servlet-name>
        <servlet-class>
            org.directwebremoting.servlet.DwrServlet
        </servlet-class>
            <!--  指定处于开发阶段的参数  -->
            <init-param>
                <param-name>debug</param-name>
                <param-value>true</param-value>
            </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>dwr-invoker</servlet-name>
        <url-pattern>/dwr/*</url-pattern>
    </servlet-mapping>
<!-- Ajax 框架的配置结束  引入 DWR 的 servlet -->
```


注意

上述代码在一个工程中只需配置一次。

③ 在 WEB-INF 文件夹中创建一个 dwr.xml 文件，完成后台 Java 逻辑处理类的路径以及对应 javascript 别名“UserIsExist”的配置。关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 2.0//EN"
    "http://getahead.org/dwr/dwr20.dtd">
<dwr>
<allow>
        <create creator="new" javascript="UserIsExist">
          <paramname="class" value="com.zdsoft.action.UserIsExistBean" />
        </create>
    </allow>
</dwr>
```

2．在页面导入 DWR 核心库及自定义接口函数库

参考任务解析相关内容编写三行代码：

```
<script type='text/javascript' src='/firstproject/dwr/util.js'></script>
<script type='text/javascript' src='/firstproject/dwr/engine.js'></script>
<script type='text/javascript' src='/firstproject/dwr/interface/UserIsExist.js'>
```


注意

上述代码中的 firstproject 是本任务用到的工程名，若读者自己的工程名与书上的不一致时，请修改为自己的工程名。

3．在用户名输入框中添加失去焦点的触发事件代码

在用户名输入框中加入失去焦点的触发事件（onblur 事件），触发自定义函数 submitDwr，使用 this.value 传入当前输入框的值，关键代码如下：

```
<s:textfield name="name" onblur=" submitDwr(this.value)"></s:textfield>
```

4．编写使用 DWR 方式提交的 JavaScript 函数

首先定义名为 submitAjax ()的函数，参数为需要验证的用户名，关键代码如下：

```
<script language="javascript">
        function submitDwr(name) {
        }
</script>
```

然后，在 submitAjax 函数中编写 DWR 方式的 JavaScript 代码。在该函数可以直接使用之前定义的 JavaScript 别名 UserIsExist 来调用对应的后台 Java 类中的方法，在参数列表中除了要将用户名传给后台，还需在参数列表中声明响应函数的名称。关键代码如下：

```
//方法有两个参数，第一个是向后台传的参数，第二个是指定返回时的响应函数
UserIsExist.userIsExist(name,callBack);
```

5．编写后台处理代码

首先，创建 com.zdsoft.action.UserIsExistBean 类（注意该类的路径及类名必须与 dwr.xml 配置文件中保持一致），在 UserIsExistBean 类中添加执行业务逻辑的 userIsExist 方法，在方法中判断前台传入的用户名是否为张三，若为张三则返回字符串 true，否则返回字符串 false，关键代码如下：

```
public class UserIsExistBean {
        public String userIsExist(String name) throws Exception {
```

```
        // 下面代码只是举例用户名张三已经存在进行验证，实际中需连接数据库数据进行验证
            if ("张三".equals(name)) {
                return "true";
            } else {
                return "false";
            }
        }
    }
```

6．编写 JavaScript 响应函数进行善后工作

首先定义一个在步骤 3 的第二个小步骤所指定的响应函数 callBack，参数列表中有关一个变量 data，该变量即成功调用后台处理后返回的结果，关键代码如下：

```
function callBack (data) {
}
```

然后，在 callBack 函数中编写善后工作的代码，此处是判断返回的结果 data 是否为字符串“true”，提示用户名存在或不存在。关键代码如下：

```
//通过 data 参数得到后台处理的结果，下面就是我们自己写的善后工作代码
    if(data=="true"){
        alert('用户名已经存在!');
    }else{
        alert('用户名不存在，可以使用!');
    }
```

到此，整个 AJAX 技术的实现过程就完成了。本例调试运行的效果如图 2.1-1（a）、2.1-1（b）所示。通过比较上一节的传统使用 XMLHttpRequest 对象的 AJAX 实现方式，我们会发现，用 DWR 框架来开发 AJAX 应用，JavaScript 代码更少，更加简单容易。

2.3 巩固与提高

一、选择题

1．AJAX 技术的实现原理：是使用以下（　　）对象来实现的。

A．AjaxRequest　　B．AjaxResponse

C．AjaxHttpRequest　　D．XMLHttpRequest

2．使用 XMLHttpRequest 对象的某个属性来获得返回的字符串，这个属性是（　　）。

A．onreadystatechange　　B．readyState

C．status　　D．responseText

3．使用 XMLHttpRequest 对象的某个属性来指定响应函数，这个属性是（　　）。

A．onreadystatechange　　B．readyState

C．status　　D．responseText

4．使用 XMLHttpRequest 对象的某个方法来设置提交方式及地址，这个方法是（　　）。

A．send　　B．open　　C．post　　D．get

5．使用 XMLHttpRequest 对象的某些属性来判断整个 AJAX 调用中的执行状态，这些属性有（　　）。

A．readyState 和 status　　B．readyState 和 responseText

C．onreadystatechange 和 status　　D．responseText 和 onreadystatechange

6．DWR 的配置文件默认的放置路径是（　　）。

A．WEB-INF\js　　B．src　　C．WEB-INF\lib　　D．WEB-INF

7．DWR 框架的核心 JavaScript 函数库是（　　）。

A．tools.js　　B．自定义接口函数库

C．base.js　　D．engine.js

8．基于 DWR 框架的 AJAX 应用，需要依赖的 JavaScript 函数库有（　　）。

A．tools.js 和 engine.js　　B．tools.js 和自定义接口函数库

C．engine.js 和自定义接口函数库　　D．tools.js 和 base.js

二、填空题

1．AJAX 技术的实现原理是使用_____对象来实现的。

2．异步提交表现在于页面并不是_____提交的，只是_____提交。

3．XMLHttpRequest 对象的 open 方法的参数有_____和_____。

4．在 AJAX 应用中用来完成 AJAX 处理完毕后的善后工作的是_____。

5．在响应函数中使用 XMLHttpRequest 对象的_____属性判断后台是否执行（取值_____为表示执行完毕），使用 XMLHttpRequest 对象的_____属性判断执行是否成功（取值为_____表示执行成功），使用 XMLHttpRequest 对象的_____属性获得后台返回的结果。

6．表单中文本框的失去焦点的触发事件是_____。

7．DWR 框架是一个用于改善_____与_____交互的远程服务器端 AJAX 开源框架，在使用时需导入它所提供的一些 JavaScript_____，其中最基本的是 engine.js。

8．DWR 框架需要配置_____文件，将后台 Java 代码登记到 DWR 框架中。

三、操作题

1．改造 2.2 任务，在页面中密码框获得焦点时触发 AJAX 代码来验证用户名是否唯一，其他要求不变。