【Java框架型项目从入门到装逼】第五节 - 在Servlet中接收和返回数据

在上一节的程序中，我们可以看到HttpServletRequest, HttpServletResponse这两个对象。可以说，这是JavaWeb中至关重要的两个对象。接下来，我们来做一个简短的说明：

1、HttpServletRequest

request对象（HttpServletRequest）代表客户端的请求，当客户端通过HTTP协议访问服务器
时，HTTP请求头中的所有信息都封装在这个对象中，通过这个对象提供的方法，可以获得客户端请求的所有信息。

让我们回顾刚才的过程，我们在浏览器的地址栏中输入http://localhost/wzry/login.do，那么我们就是给服务器发起了一个请求login.do。就是web.xml中配置的url-pattern，随便你写什么，不是非得要“xxx.do”。

其中，请求头就是Request Headers. 我们还可以看到请求的方式是Get方式，通过浏览器地址栏的方式就是GET方式。现在，我们改变在请求的同时加入一点信息：
http://localhost/wzry/login.do?username=admin&password=123&type=weixin
在请求地址后面加一个？，开始拼接数据，每一个数据都是key=value 的形式，不同数据之间用 & 连接。再次回车。我们可以看到信息发生了变化：

不论你是什么请求，你往服务器传递的数据只能是 字符串！

现在，我们可以在Servlet中接收这些参数！

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String username = req.getParameter("username");
    String password = req.getParameter("password");
    String type = req.getParameter("type");

    System.out.println("用户登录...");

    System.out.println(username);
    System.out.println(password);
    System.out.println(type);

}
```

运行结果：

```
进入com.wzry.web.LoginServlet
用户登录...
admin
123
weixin
```

正常情况下，为了保存这些数据，我们都会各自建立一个Java类，比如用户类。我们为了方便起见，可以采用一种公用的数据结构来保存，那就是Map。从道理上也能明白吧，客户端传递数据到我们的服务器，我们是不是首先得想办法把它存起来？好像给你一筐鸡蛋，然后他说，鸡蛋给你，框子我得拿走，那么你是不是得找一个容器，把鸡蛋装起来呢？不就是这个道理嘛。

Map就是这么一个容器。
修改后的代码：

```
String username = req.getParameter("username");
String password = req.getParameter("password");
String type = req.getParameter("type");

System.out.println("用户登录...");

System.out.println(username);
System.out.println(password);
System.out.println(type);

System.out.println("开始存入Map...");
Map<String,Object> user = new HashMap<String,Object>();

user.put("username", username);
user.put("password", password);
user.put("type", type);

System.out.println("存入Map成功！");

System.out.println(user);
```

在实际的开发中，传进来的数据肯定是不一样的，如果我们太依赖于getParameter这个方法，就无法做到灵活变通。那么有没有一种通用的方法，让request对象中附带的数据自动转换为Map呢？

我已经封装好了一个工具类，里面就有这样的方法。

```
public static Map<String,Object> getParameters(HttpServletRequest request){
    Map<String,Object> map = new HashMap<String,Object>();
    Enumeration<String> names = request.getParameterNames();

    while(names.hasMoreElements()){
        String key = names.nextElement();           //获取key值
        String value = request.getParameter(key);   //获取对应的value
        map.put(key, value);
    }

    return map;
}
```

这里用到了枚举，实现细节我们不去讨论，现在用这个代码来进行一把骚操作。
静态导入这个工具类：

import static com.wzry.util.WebUtil.*;

直接调用转换的方法：

```
Map<String,Object> user = getParameters(req);
System.out.println(user);
```

爽不？

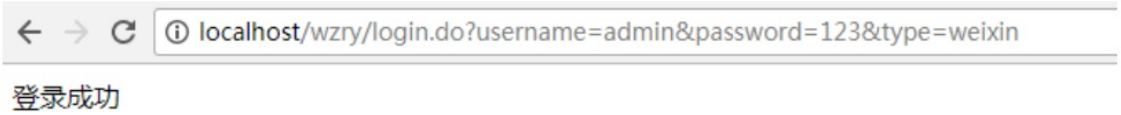
2、HttpServletResponse

Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的request对象（HttpServletRequest）、和代表响应的response对象（HttpServletResponse）。
request和response对象即代表请求和响应，那我们要获取客户机提交过来的数据，只需要找request对象就行了。要向客户机输出数据，只需要找response对象就行了。

在刚才的例子中，我们添加以下代码：

```
resp.setContentType("text/html;charset=utf-8");
PrintWriter out = resp.getWriter();
out.println("登录成功");
```

页面效果：



我们通过这种方式，就可以往客户端发送一个数据。

刚才讲了GET方式提交可以直接在浏览器地址栏操作，GET方式提交的缺点就是会暴露自己的数据信息，还有一种POST提交的方式。相比GET方式要安全一点，它不会直接暴露数据。现在我们通过form表单来做一个讲解。
在WebContent目录下新建一个index.jsp。

编写form表单：

```
<!-- 采用post表单提交 -->
<form style="margin-left:200px;" id="myform" name="myform" method="post" onsubmit="return sumbitTest();"
      action="login.do">
    <table>
        <tr>
            <td>用户名:</td>
            <td> <input type="text" name="username" /> </td>
        </tr>
        <tr>
            <td>密码:</td>
            <td> <input type="password" name="password" /> </td>
        </tr>
        <tr>
            <td colspan="2">
                <input style="display:none" id="login_btn" type="submit" value="提交">
            </td>
        </tr>


        <input id="loginType" type="text" name="loginType" hidden='true'/> </td>

    </table>
</form>
```

用户名和密码都有对应的id：

```
<tr>
    <td>用户名:</td>
    <td> <input type="text" name="username" id="username"/> </td>
</tr>
<tr>
    <td>密码:</td>
    <td> <input type="password" name="password" id="password"/> </td>
</tr>
```

```
//验证登录信息
function sumbitTest(){
    //在这个方法中可以对登录信息进行校验 （作业，用户名和密码都不能为空）
    if(!$("#username").val()){
        alert("用户名不能为空！");
        return false;
    }

    if(!$("#password").val()){
        alert("密码不能为空！");
        return false;
    }

}
```

为了项目的严谨性，防止用户通过抓包的方式手动提交，从而绕过JS验证，我们一般还需要在后台也进行一个验证。

```
if(user.get("username") == null){
    System.out.println("用户名不能为空！");
}

if(user.get("password") == null){
    System.out.println("密码不能为空！");
}
```

为了方便起见，我们先把js验证给去掉。

```
//验证登录信息
function sumbitTest(){

    return true;

    //在这个方法中可以对登录信息进行校验 （作业，用户名和密码都不能为空）
    if(!$("#username").val()){
        alert("用户名不能为空! ");
        return false;
    }

    if(!$("#password").val()){
        alert("密码不能为空! ");
        return false;
    }

}
```

我们故意不填写用户名和密码，点击登录按钮，结果并没有什么卵用。因为其实传递到后台是有值的，只是为""，这一点和js不同，在Java中，""不等于假，它只是代表一个空字符串。所以我们需要修改一下验证条件。还有，为了不让代码继续往下执行，我们需要及时return。

```
if(user.get("username") != null && user.get("username").toString().equals("") ){
    System.out.println("用户名不能为空! ");
    return;
}

if(user.get("password") != null && user.get("password").toString().equals("") ){
    System.out.println("密码不能为空! ");
    return;
}
```

为了给用户返回错误信息，我们得把信息抛到页面上。

```
if(user.get("username") != null && user.get("username").toString().equals("") ){
    System.out.println("用户名不能为空! ");
    resp.setContentType("text/html;charset=utf-8");
    PrintWriter out = resp.getWriter();
    out.println("用户名不能为空! ");
    return;
}

if(user.get("password") != null && user.get("password").toString().equals("") ){
    System.out.println("密码不能为空! ");
    resp.setContentType("text/html;charset=utf-8");
    PrintWriter out = resp.getWriter();
    out.println("密码不能为空! ");
    return;
}
```

关注一下，这里有两个重复点，于是考虑封装。

```
public class StringUtil {

    public static boolean isEmpty(Object o){
        if(o == null) return true;
        if("".equals(o.toString())){
            return true;
        }
        return false;
    }

    public static boolean isNotEmpty(Object o){
        return !isEmpty(o);
    }

}
```

```
    if(StringUtil.isEmpty(user.get("username"))){
        System.out.println("用户名不能为空! ");
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out = resp.getWriter();
        out.println("用户名不能为空! ");
        return;
    }

    if(StringUtil.isEmpty(user.get("password")) ){
        System.out.println("密码不能为空! ");
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out = resp.getWriter();
        out.println("密码不能为空! ");
        return;
    }
```

再来一个通用的把数据返回给前台的方法：

```
public static void writeObject(HttpServletResponse response,Object o){
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = null;
    try {
        out = response.getWriter();
        out.println(o);
    } catch (IOException e) {
        e.printStackTrace();
    }finally{
        out.flush();
        out.close();
    }
}


    if(StringUtil.isEmpty(user.get("username"))){
        System.out.println("用户名不能为空! ");
        WebUtil.writeObject(resp, "用户名不能为空! ");
        return;
    }

    if(StringUtil.isEmpty(user.get("password")) ){
        System.out.println("密码不能为空! ");
        WebUtil.writeObject(resp, "密码不能为空! ");
        return;
    }
```

我要下载源码