# 【手把手】JavaWeb 入门级项目实战 -- 文章发布系统 （第四节）

首先，更正一下上一章中的一个小错误，就是在index.jsp中，banner部分没有添加结束的标签,加上去就OK了，我也是完善页面的时候发现的。

另外，index.jsp中引入的jQuery也需要换成本地的。

```
<script src="${basePath}/static/js/jQuery.js"></script>
```

今天我把页面重构了一下，添加了内容区和底栏（footer），我会把目前的代码上传的，有需要的自己去看就行了，我们就不在前台页面花费太多的时间了。div＋css，布局等等，这些东西以后有时间的话，我单独开贴分享吧。

都已经写了三篇文章了，还没有写Java代码，实在有些说不过去。

## 1. 登陆页面

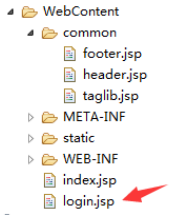
登陆页我已经写好了，现在看看效果，简单说明一下。


点击登陆按钮，可以跳转到登陆页面。

登陆按钮就是一个超链接。

```
<div class='login'>
    <span><a href="login.jsp">登陆</a></span>
    <span>|</span>
    <span><a href="javascript:void(0)">注册</a></span>
</div>
```

JSP页面就是一个servlet，但是省去了很多写Servlet的麻烦，login.jsp已经写好了，就放在WebContent目录下。




昨天憋了一上午，总算写好了登陆页面。我不是专业做前端的，所以只做了一个大概样子。用了很多css3的属性，所以IE678上浏览的效果是不好的。

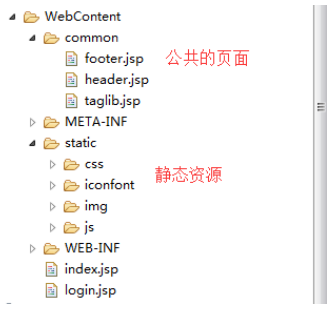
关于这个页面，今天我调整了一下，不想搞得那么复杂了，我去掉了所有的图标和飘动白云，关于css特效，h5的话呢，以后单独拿出来说明吧，毕竟好多人都反应说太花哨了，因为是入门级的小项目，我也不想弄得那么复杂了。

虽然页面单调了些，不过对于初学者来说，相对来说比较好理解。之前的页面的确有点太花哨了，还弄了几朵云飘来飘去的，说不定还影响性能，所以我把这些都去掉了。


## 2. 新的目录结构

之前的代码有很多冗余的地方，比如标题栏，每个页面都需要写一遍。而且js和css都是写在本页面的。实际开发一般都不会这么做。所以，我把这些东西都分离出来了，放在各自的目录里。

以下是新的目录结构：




header.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<div class="header">
    <div class='logo'>原创文字</div>
    <ul>
        <li class='first'><a href="index.jsp">首页</a></li>
        <li class='item'><a href="javascript:void(0)">原创故事</a></li>
        <li  class='item'><a href="javascript:void(0)">热门专题</li>
        <li  class='item'><a href="javascript:void(0)">欣赏美文</li>
        <li  class='item'><a href="javascript:void(0)">赞助</a></li>
    </ul>

    <div class='login'>
        <span><a href="login.jsp">登陆</a></span>
        <span>|</span>
        <span><a href="javascript:void(0)">注册</a></span>
    </div>
</div>
```

这就是标题栏，以后新增的jsp页面，只需要把这个header.jsp引入就可以了。注意，这种引入就相当于把里面的代码原封不动地拷贝进去，所以如果用相对路径引用资源文件，就还是以原本的页面为准。

引入方式：

```
<!-- 头部页面 -->
<%@include file="common/header.jsp" %>
```

footer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<div class='footer'>
    免责声明：本站所有素材均来自网络，仅供学习交流。如果侵犯了您的权益，请联系我，我会第一时间删除侵权内容。
</div>
```

taglib.jsp (一些公共的配置等)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%
    String path = request.getContextPath();
    int port = request.getServerPort();
    String basePath  = null;
    if(port==80){
        basePath = request.getScheme()+"://"+request.getServerName()+path;
    }else{
        basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path;
    }
    request.setAttribute("basePath", basePath);
%>
```

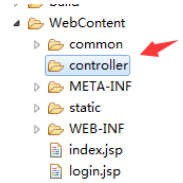
basePath 就是项目的根路径。这样做的好处就是，使得JSP看起来很干净，没有那么多冗余的代码了。

大概就是这个样子，接下来，我们开始写业务。

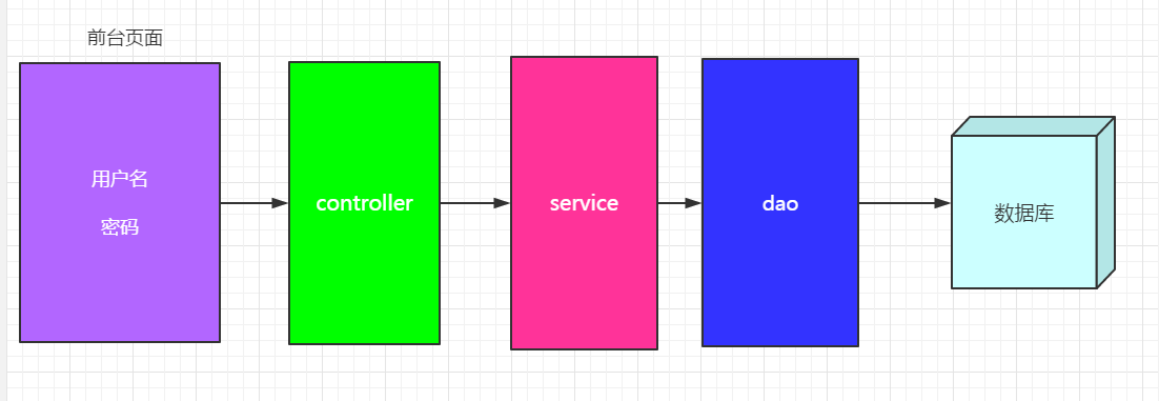
### 3. 登陆功能的MVC流程

登陆框中，目前只有用户名和密码这两个选项。

我们首先要做的就是将这两个值传递到后台。所谓的后台，其实就是Java代码。为了看起来比较清晰，我们在WebContent目录下新建一个controller包。




这是一个MVC分层的示意图




意思就是说，用户登录之后，我们需要验证它的用户名和密码是否正确，那么就需要将数据拿到数据库里面去匹配。总体的流程大概是这样：我先在前台获取用户名和密码，然后到controller层（控制层），这一层需要接受你传过来的用户名和密码，进行一些基本的控制。

然后继续将数据传递到service层，也就是业务层，这一层会根据具体的业务对你的数据进行判断和分析，最后，才传递到dao层，这一层原则上就是和数据库进行交互的。多半是写sql语句然后操作数据库。

就比如说用户登录这个功能，我需要判断的就是

1. 你这个用户是否存在？
2. 用户名和密码是否正确？

最终，还需要将登录的标志返回给前台。

dao -> servide -> controller -> JSP

这样就是一个完整的流程。

### 4. 从JSP到controller层

让我们打开login.jsp页面，引入jQuery

```
<script src="${basePath}/static/js/jQuery.js"></script>
```

登录框的HTML代码：

```
<!-- 登陆框 -->
<div class='content'>
    <div class='logo'><i style='font-size:90px;' class="iconfont icon-denglu"></i></div>
    <div class='inputBox mt50 ml32'>
        <input type="text" id="username" autofocus="autofocus" autocomplete="off" maxlength="60" placeholder="请输入账号/邮箱/手机号">
        <i style='font-size:20px;margin-left:-32px;opacity:0.8;' class='iconfont icon-yonghuming'></i>
    </div>
    <div class='inputBox mt50 ml32'>
        <input type="password" id="password" autofocus="autofocus" autocomplete="off" maxlength="60" placeholder="请输入密码">
        <i style='font-size:20px;margin-left:-32px;opacity:0.8;' class='iconfont icon-mima'></i>
    </div>

    <div class='mt50 ml32'>
        <button class="login_btn" onclick="#">登陆</button>
    </div>
</div>
```

我们在下面写一个script块，js代码就全部写在这里。

```
<script>
                在这里写JS代码

</script>
```

给登陆按钮绑定一个点击事件：

```
<div class='mt50 ml32'>
        <button class="login_btn" onclick="login()">登陆</button>
</div>
```

登陆方法

```
function login(){
    var username = $('#username').val();
    var password = $('#password').val();
    alert(username + "," + password);
}
```

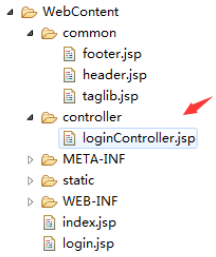
当成功alert出来数据后，说明到此为止的代码是正确的。

接下来，利用jQuery的ajax方法，将数据提交到controller层。

```
function login(){
    var username = $('#username').val();
    var password = $('#password').val();
    $.ajax({
        type:"post",//请求方式
        url:"${basePath}/controller/loginController.jsp",//请求地址
        data:{"username":username,"password":password},//传递给controller的json数据
        error:function(){
            alert("登陆出错！");
        },
        success:function(data){ //返回成功执行回调函数。

        }
    });

}
```

我已经都写好注释了，ajax方法在web开发过程中，是被普遍使用的。

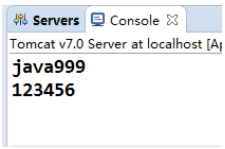
新建一个loginController.jsp，这就是所谓的服务器端。




```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%

    //设置请求的编码
    //request.setCharacterEncoding("UTF-8");
    //获取客户端传递过来参数
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    System.out.println(username);
    System.out.println(password);
    //如果用户名和密码不为空

%>
```

从JSP页面到controller，用户名和密码是被装在一个叫做request的变量中，它其实也就相当于一个json，一个map，都是差不多的东西，这里就不详细说明了。当然了，他也是JSP九大隐式对象中的一员。


我们来测试一下，点击登陆按钮。

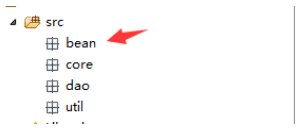



成功了！可以看到数据已经成功传递到controller层了。

因为我们还没有数据表和JavaBean，所以我们先不急着写service层，先开始编写JavaBean吧。

### 5. 从JavaBean到数据库表。

我们在src目录下新建一个存放JavaBean的包




关于JavaBean，如果不是很了解的话，可以看看这篇文章：

http://www.cnblogs.com/skyblue-li/p/5900216.html

一个记录用户信息的JavaBean，我想了以下这些属性：

```
private String id;    //主键，采用UUID
private String username;  //用户名
private String password;  //密码
private String headerPic; //头像
private String email;     //电子邮箱
private Integer male;     //性别 0男 1女 3保密
private String createTime;//创建时间
private String updateTime;//最后更新时间
private Integer isDelete; // 删除状态0未删除1删除
private String address;   //地址
private String telephone; //电话
```

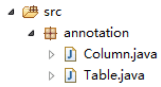
当你的JavaBean设计好了，差不多对应的数据库表也就出来了。

之前写过一篇关于注解的文章：

http://www.cnblogs.com/skyblue-li/p/5900228.html

现在可以用这个知识点做点有趣的事情了，比如将一个JavaBean转换成建表语句。

新建一个注解包，里面添加两个注解




column.java

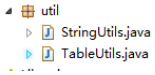
```
package annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.FIELD)       //注解的目标
@Retention(RetentionPolicy.RUNTIME)
public @interface Column {

    public String field() ; //字段名称
    public boolean primaryKey() default false;//是否为主键
    public String type() default "VARCHAR(80)";//字段类型
    public boolean defaultNull() default true; //是否允许为空

}
```

Table.java

```
package annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.TYPE)        //注解的目标是类
@Retention(RetentionPolicy.RUNTIME)
public @interface Table {

    public String tableName();
}
```

我们创建了两个注解。

接下来，在util包（也就是工具包）中新建两个工具类




StringUtils 字符串工具类

```
package util;


public class StringUtils {

    public static boolean isEmpty(String str) {
        return null == str  || str.equals("")
                || str.matches("\\s*");
    }

    public static String defaultValue(String content,String defaultValue){
        if(isEmpty(content)){
            return defaultValue;
        }
        return content;
    }
}
```

isEmpty的作用是判断字符串是否为空。
defaultValue表示给字符串设置默认值，有点类似于oracle数据库中的nvl语法。

TableUtils 数据表工具类

```
package util;

import java.lang.reflect.Field;

import annotation.Column;
import annotation.Table;

public class TableUtils {


    public static String getCreateTableSQl(Class<?> clazz){
        StringBuilder sb = new StringBuilder();
        sb.append("create table ");
        //获取表名
        Table table = (Table) clazz.getAnnotation(Table.class);
        String tableName = table.tableName();
        sb.append(tableName).append("(\n");

        Field[] fields = clazz.getDeclaredFields();
        String primaryKey = "";
        //遍历所有字段
        for (int i = 0; i < fields.length; i++) {
            Column column = (Column) fields[i].getAnnotations()[0];
            String field = column.field();
            String type = column.type();
            boolean defaultNull = column.defaultNull();

            sb.append("\t" + field).append(" ").append(type);
            if(defaultNull){
                if(type.toUpperCase().equals("TIMESTAMP")){
                    sb.append(",\n");
                }else{
                    sb.append(" DEFAULT NULL,\n");
                }
            }else{
                sb.append(" NOT NULL,\n");
                if(column.primaryKey()){
                    primaryKey = "PRIMARY KEY ("+field+")";
                }
            }
        }

        if(!StringUtils.isEmpty(primaryKey)){
            sb.append("\t").append(primaryKey);
        }
```

```
        sb.append("\n) DEFAULT CHARSET=utf8");

        return sb.toString();
    }

}
```

getCreateTableSQl方法是利用反射和注解有关的知识，给一个JavaBean自动生成建表语句，目前只支持MySQL，因为这方面的知识我也是刚开始学，写得不好的地方还请各位多多包涵。

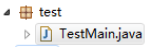
接下来，给JavaBean添加注解。

```
@Table(tableName = "t_user")
public class User{
   //属性
}
```

属性如下：

```
@Column(type = "varchar(30)" ,field = "id" ,primaryKey = true ,defaultNull = false)
private String id;         //主键，采用UUID

@Column(type = "VARCHAR(20)", field = "username")
private String username;  //用户名

@Column(type = "VARCHAR(20)", field = "password")
private String password;  //密码

@Column(type = "VARCHAR(60)", field = "headerPic")
private String headerPic; //头像

@Column(type = "VARCHAR(60)", field = "email")
private String email;     //电子邮箱

@Column(type = "VARCHAR(2)", field = "sex")
private Integer sex;      //性别 0男 1女 3保密

@Column(type = "datetime", field = "create_time")
private String createTime;//创建时间

@Column(type = "timestamp", field = "update_time")
private String updateTime;//最后更新时间

@Column(type = "int(1)", field = "is_delete")
private Integer isDelete; // 删除状态  0未删除  1删除

@Column(type = "VARCHAR(200)", field = "address")
private String address;   //地址

@Column(type = "VARCHAR(15)", field = "telephone")
private String telephone; //电话
```

创建一个测试包和测试类：




```
package test;

import bean.User;
import util.TableUtils;

public class TestMain {
    public static void main(String[] args) {
        String sql = TableUtils.getCreateTableSQl(User.class);
        System.out.println(sql);
    }
}
```

运行

```
create table t_user(
        id varchar(30) NOT NULL,
        username VARCHAR(20) DEFAULT NULL,
        password VARCHAR(20) DEFAULT NULL,
        headerPic VARCHAR(60) DEFAULT NULL,
        email VARCHAR(60) DEFAULT NULL,
        sex VARCHAR(2) DEFAULT NULL,
        create_time datetime DEFAULT NULL,
        update_time timestamp,
        is_delete int(1) DEFAULT NULL,
        address VARCHAR(200) DEFAULT NULL,
        telephone VARCHAR(15) DEFAULT NULL,
        PRIMARY KEY (id)
) DEFAULT CHARSET=utf8
```

OK，拿到sql语句了。

我已经安装了mysql，用root用户登陆后，新建一个database

```
mysql> create database article;
Query OK, 1 row affected (0.00 sec)

mysql>
```

使用这个database

```
mysql> use article;
Database changed
```

将刚才得到的sql语句复制进去，加分号，回车。

```
mysql> create table t_user(
    ->  id varchar(30) NOT NULL,
    ->  username VARCHAR(20) DEFAULT NULL,
    ->  password VARCHAR(20) DEFAULT NULL,
    ->  headerPic VARCHAR(60) DEFAULT NULL,
    ->  email VARCHAR(60) DEFAULT NULL,
    ->  sex VARCHAR(2) DEFAULT NULL,
    ->  create_time datetime DEFAULT NULL,
    ->  update_time timestamp,
    ->  is_delete int(1) DEFAULT NULL,
    ->  address VARCHAR(200) DEFAULT NULL,
    ->  telephone VARCHAR(15) DEFAULT NULL,
    ->  PRIMARY KEY (id)
    -> ) DEFAULT CHARSET=utf8
    -> ;
Query OK, 0 rows affected (0.19 sec)
```

这就表明数据库表已经建好了，默认编码是UTF-8。

本文结束。

最后说一下更新的问题，最近事情比较多，无奈改为周更了，正常情况下每周日定时更新。至于源码，我还没有放到github上，打算以后写得差不多了再发布吧。