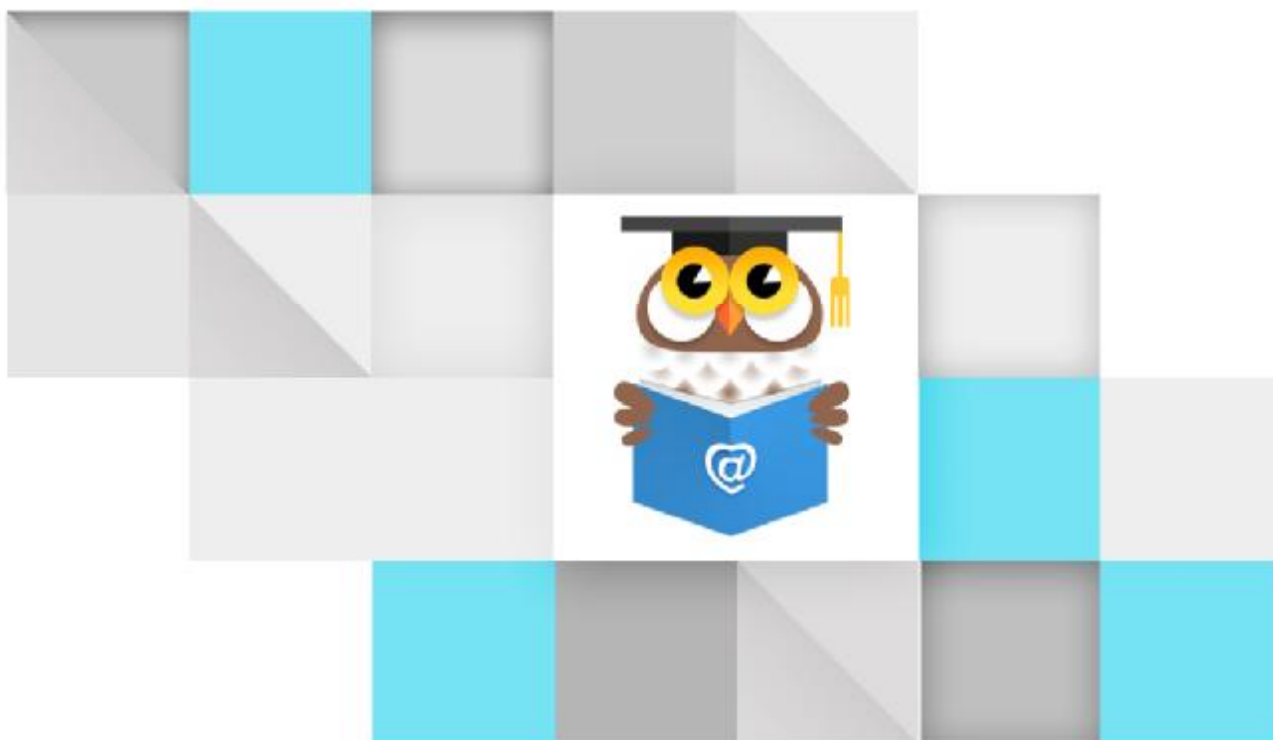


# 去哪玩旅游网（动态 WEB）

## 教学指导手册

### PRJ-WTP-JEE-003 – 模拟用户登录



**Campus** Solution Group

# 目 录

一、场景说明 .....	1
1、完成效果 .....	1
2、业务描述 .....	1
3、实现思路 .....	1
4、核心组件 .....	2
二、实训技能 .....	3
1、重点演练 .....	3
2、相关技能 .....	3
3、相关知识点 .....	3
4、前置条件 .....	4
5、搭建环境 .....	4
三、场景任务 .....	5
任务 1、创建并配置 Servlet 组件 .....	5
任务 2、获取用户提交数据 .....	7
任务 3、页面定向跳转 .....	9
任务 4、可维护的跳转连接配置 .....	11
场景总结 .....	15

# 一、场景说明

## 1、完成效果

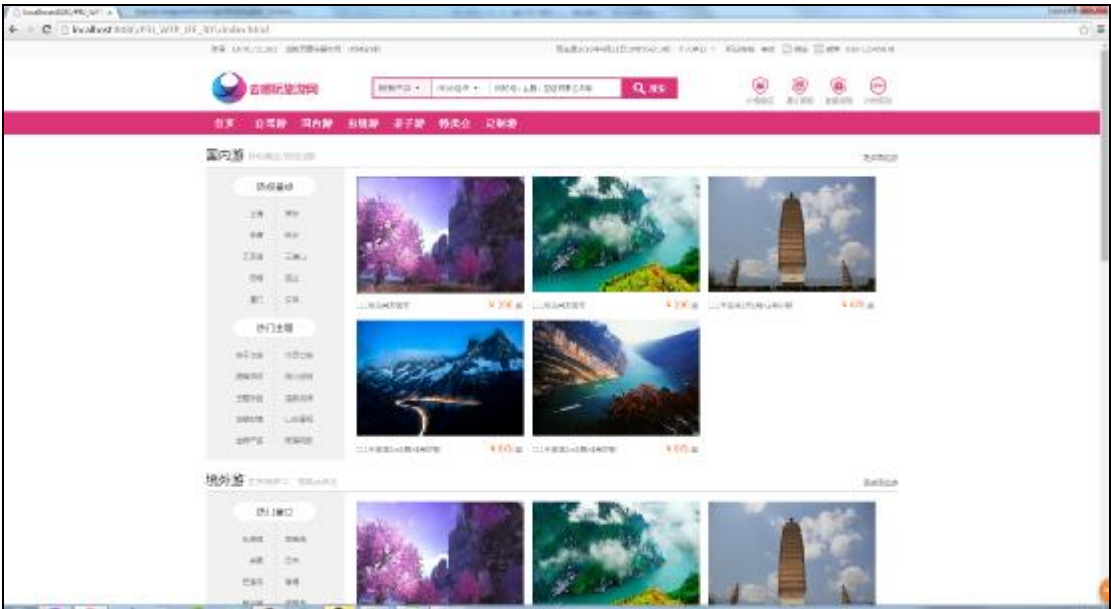


图 1-1-1

## 2、业务描述

本场景主要用于“模拟”实现《去哪儿旅游网》的用户登录功能（**固定用户名和密码**）。

2-1. 在《去哪儿旅游网》的登录页面验证登录有效性（用户名与密码是否正确）。

2-1.1. 正确情况：用户名 = **18701721202**，密码 = **123456**。

2-1.2. 登录信息有效，则跳转到首页：index.html。

2-1.3. 登录信息无效，则跳转回登录页：login.html。

2-2. 确保登录正确的跳转页面（index.html）与登录失败的跳转页面（login.html）是易于维护的，即跳转页面存储于文件（有代码负责读取）。

## 3、实现思路

3-1. 本场景建议将“模拟登录功能”分为四个任务依次实现：

3-1.1. 任务1. 创建登录Servlet组件，为验证登录有效性做准备。

3-1.2. 任务2. 由任务1创建的Servlet负责获取用户名和密码。

3-1.3. 任务3. 基于任务2,判断登录数据的有效性,根据判断结果实现不同页面的跳转。

3-1.4. 任务4. 基于任务3,增加页面跳转连接的可维护性。

## 4、核心组件

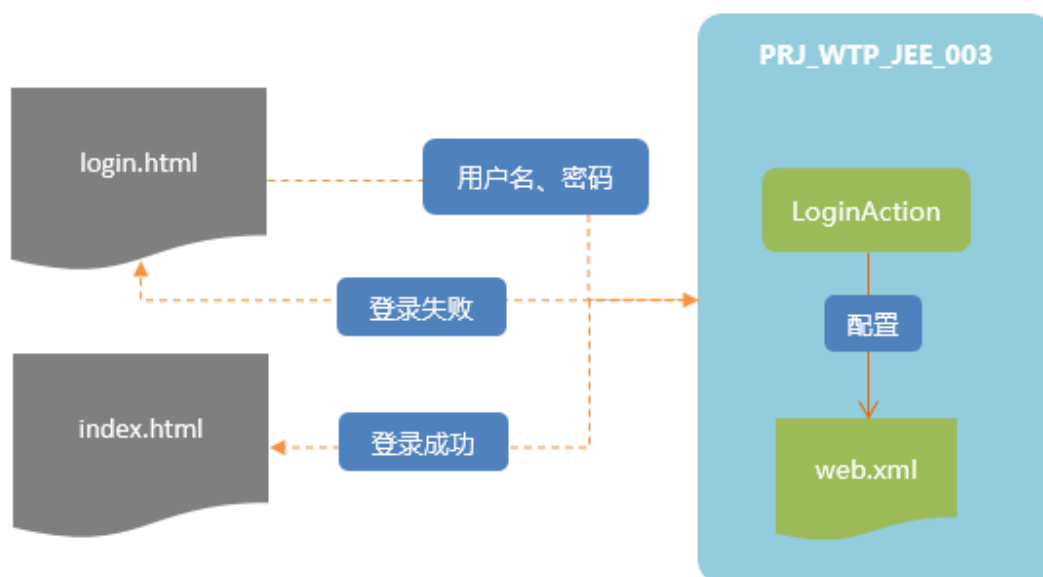


图 1-4-1

4-1. LoginAction (需实现)：

4-1.1. 负责处理《去哪玩旅游网》项目登录页面的请求。

4-1.2. 该组件将根据验证与判定结果，跳转到不同页面。

4-2. web.xml (需实现)：

4-2.1. 《去哪玩旅游网》站点的核心配置文件。

4-2.2. LoginAction只有配置于该文件中才能响应客户端的请求。

4-2.3. 登录正确与错误的不同跳转连接，存放于该文件中。

#### 4-3. login.html ( 已提供 )

《去哪玩旅游网》项目的登录页（静态页面），在本场景中的职责是：

- 1 ) 负责发送登录请求到Servlet。
- 2 ) 当登录失败后重新跳回到该界面。

#### 4-4. index.html ( 已提供 )

《去哪玩旅游网》项目的首页（静态页面），只有登录成功后才能显示该页面。

## 二、实训技能

### 1、重点演练

- 1-1. 创建、配置、使用Servlet组件。
- 1-2. 通过Servlet组件，获取客户端提交的请求数据。
- 1-3. 通过HttpServletRequest对象，读取表单数据。
- 1-4. 通过客户端重定向技术，实现页面（HTML）跳转。
- 1-5. 通过ServletConfig对象，读取web.xml文件的配置信息。

### 2、相关技能

- I Servlet技术应用

### 3、相关知识点

- I 开发Servlet组件
- I 配置Servlet组件
- I 接收客户端请求

- I 响应客户端请求
- I ServletConfig接口
- I HttpServletRequest
- I 获取表单数据
- I 客户端重定向

## 4、前置条件

4-1. 前置场景：PRJ-WTP-JEE-002 – 监控请求与响应

4-2. 前置技能：

4-2.1. Java开发工具（Eclipse）。

4-2.2. Web容器（Tomcat）。

4-2.3. Servlet应用技术。

## 5、搭建环境

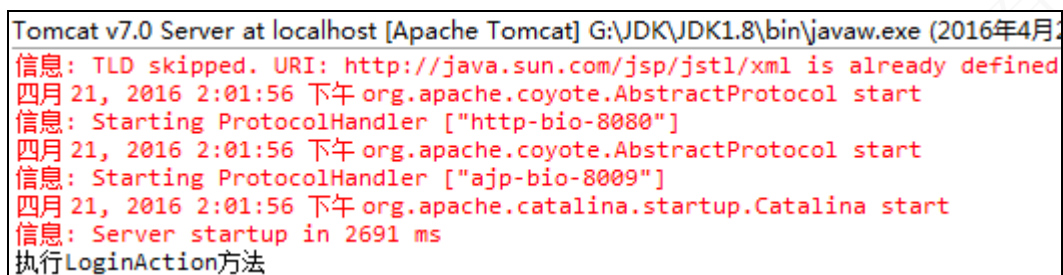


## 三、场景任务

### 任务 1、创建并配置 Servlet 组件

#### 1. 任务说明：

##### 1-1. 完成效果：



```
Tomcat v7.0 Server at localhost [Apache Tomcat] G:\JDK\JDK1.8\bin\javaw.exe (2016年4月21日 下午2:01:56)
信息: TLD skipped. URI: http://java.sun.com/jsp/jstl/xml is already defined
四月 21, 2016 2:01:56 下午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
四月 21, 2016 2:01:56 下午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
四月 21, 2016 2:01:56 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 2691 ms
执行LoginAction方法
```

图 3-1-1

##### 1-2. 任务目标：

1-2.1. 创建登录Servlet组件，为验证登录有效性做准备。

##### 1-3. 任务要求：

1-3.1. 创建用于处理登录请求的Servlet组件：

- 1) 限制1. Servlet取名：**LoginAction**。
- 2) 限制2. LoginAction创建于**campsg.qunawan.action**包中。
- 3) 限制3. 允许通过**login.jhtml**结尾的URL访问该Servlet。

1-3.2. 当LoginAction获取请求后，在控制台输出：“**执行LoginAction方法**”。

#### 2. 实现思路：

2-1. 在**campsg.qunawan.action**包中创建一个Class类，命名：**LoginAction**。

2-2. 使LoginAction继承HttpServlet组件。

2-3. 在**web.xml**中配置该Servlet，将url-pattern设置为：**/login.jhtml**

2-4. 在LoginAction的doGet或doPost方法中，编写控制台输出测试语句。

2-5. 实现流程如下图：

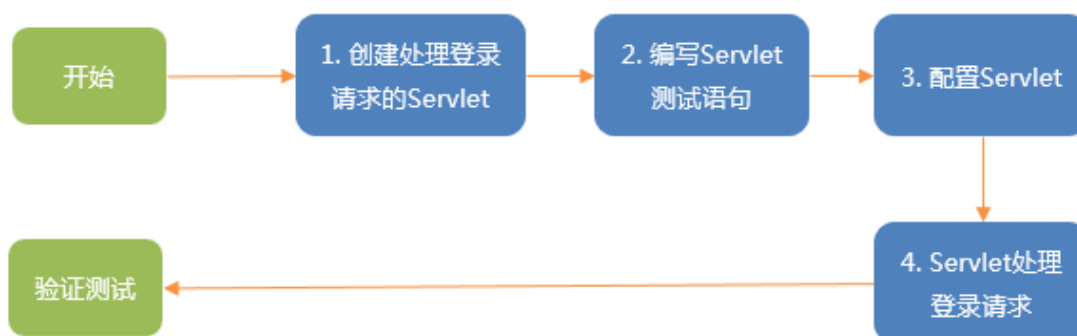


图 3-1-2

### 3. 推荐步骤：

3-1. 创建名为LoginAction的Servlet

3-2. 编写Servlet测试语句

3-2.1. 在doGet方法中，向控制台输出字符串：“*执行LoginAction方法*”。

3-2.2. 在doPost方法中，调用doGet方法。

3-3. 在web.xml中配置Servlet：

3-3.1. 配置Servlet的<url-pattern>标签值为：/login.jhtml。

3-4. Servlet处理登录请求：

3-4.1. 定位login.html：PRJ\_WTP\_JEE\_003/WebContext/login.html

3-4.2. 确保Form表单的action属性能够放到LoginAction中。

#### + 提示：

1) 访问LoginAction，Form标签的action属性应该设置为：

[/PRJ\\_WTP\\_JEE\\_003/login.jhtml](#)

属性说明：/项目名/Servlet的url-pattern

### 4. 验证与测试：



4-1. 运行项目工程

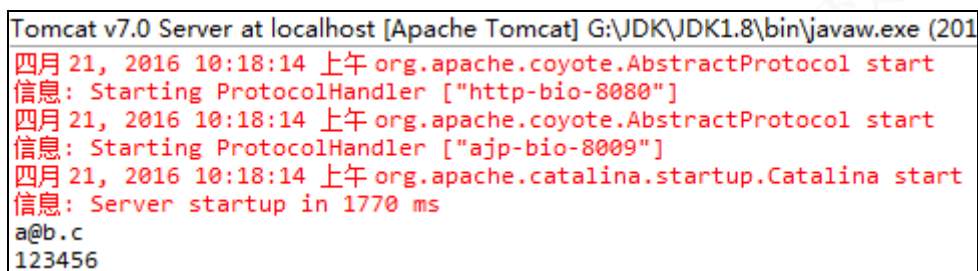
4-2. 在登录界面输入任务用户名和密码，【例如 账号：a@b.c 密码：123456】。

4-3. 并点击登录按钮，控制台应显示“执行LoginAction方法”的字样。

## 任务 2、获取用户提交数据

### 1. 任务说明：

1-1. 完成效果：



```
Tomcat v7.0 Server at localhost [Apache Tomcat] G:\JDK\JDK1.8\bin\javaw.exe (2016-04-21 10:18:14)
四月 21, 2016 10:18:14 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
四月 21, 2016 10:18:14 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
四月 21, 2016 10:18:14 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 1770 ms
a@b.c
123456
```

图 3-2-1

1-2. 任务目标：

1-2.1. 基于任务1，实现Servlet对用户名和密码的获取。

1-3. 任务要求：

1-3.1. 在LoginAction中获取客户端提交的**用户名和密码**。

1) 限制1. 用户名控件的**name属性**为：name。

2) 限制2. 密码控件的**name属性**为：password。

1-3.2. 在控制台中打印获取的用户名和密码。

### 2. 实现思路：

2-1. 在login.html中找到用户名和密码控件。

2-2. 将用户名的**name属性**修改为：name。

2-3. 将密码控件的**name属性**修改为：password。

2-4. 在LoginAction的doGet方法中，从HttpServletRequest对象中获取登录信息。

2-5. 在控制台中打印获取的用户名和密码。

2-6. 实现流程如下图：



图 3-2-2

### 3. 推荐步骤：

3-1. 设置网页控件的name属性：

3-1.1. 定位login.html：PRJ\_WTP\_JEE\_003/WebContext/login.html

3-1.2. 找到“id = tid”的input标签，修改name属性。

3-1.3. 找到“id = pwd”的input标签，修改name属性。

3-2. Servlet获取用户提交数据：

3-2.1. 在LoginAction的doGet方法中获取登录页面提交的用户名和密码。

3-2.2. 向控制台输出获取到的用户名和密码。

### 4. 验证与测试：

4-1. 运行项目工程

4-2. 在登录界面输入用户名和密码，【例如 账号：a@b.c 密码：123456】。

4-3. 点击登录按钮，控制台应显示客户端提交的用户名和密码。

## 任务 3、面重定向跳转

### 1. 任务说明：

#### 1-1. 完成效果：

##### 1-1.1. 登录信息正确，需显示的页面效果（index.html）：

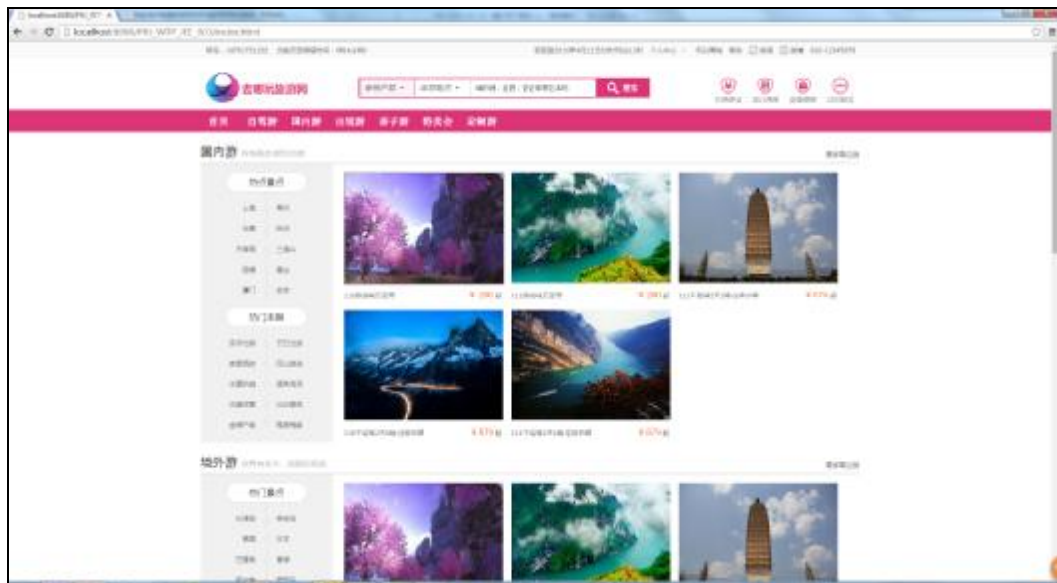


图 3-3-1

##### 1-1.2. 登录信息错误，需显示的页面效果（login.html）：

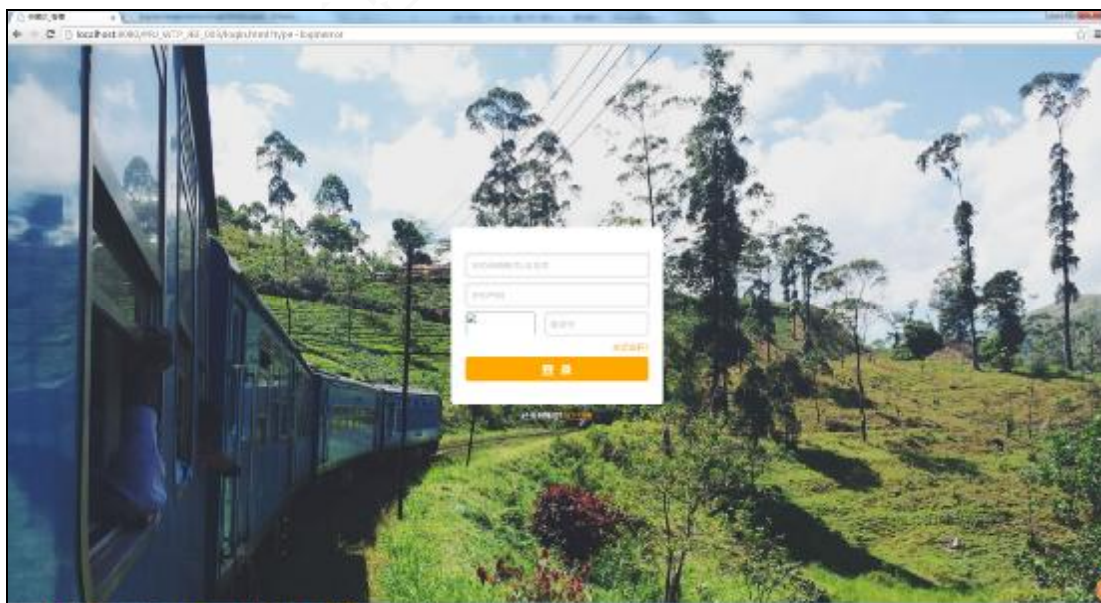


图 3-3-2

## 1-2. 任务目标：

1-2.1. 基于任务2，判断登录数据的有效性，根据判断结果实现不同页面的跳转。

## 1-3. 任务要求：

1-3.1. 在LoginAction中判断用户名与密码是否正确。

1-3.2. 如果用户名 = 18701721202，密码 = 123456，则跳转到首页：index.html。

1-3.3. 如果用户名和密码错误则跳转回登录页：login.html。

## 2. 实现思路：

2-1. 在LoginAction的doGet方法中判断登录数据有效性。

2-2. 正确，通过sendRedirect方法跳转到index.html。

2-3. 错误，通过sendRedirect方法跳转到login.html。

## 3. 推荐步骤：

2-1. 在LoginAction的doGet方法中判断用户名和密码的正确性。

2-2. 根据判断结果实现相关页面的跳转。

### + 提示：

1) 重定向跳转页面：使用response对象的sendRedirect方法。

注意：重定向跳转页面时，浏览器地址栏会切换成跳转后的URL。

## 4. 验证与测试：

4-1. 运行项目工程

4-2. 登录成功跳转：

4-2.1. 在登录页输入正确的用户名和密码，可见【完成效果图1】：。

4-3. 登录失败跳转：

4-3.1. 在登录页输入正确的用户名和密码，可见【完成效果图2】：。

## 任务 4、可维护的跳转连接配置

### 1. 任务说明：

#### 1-1. 完成效果：

##### 1-1.1. 登录信息正确，显示的页面效果（index.html）：

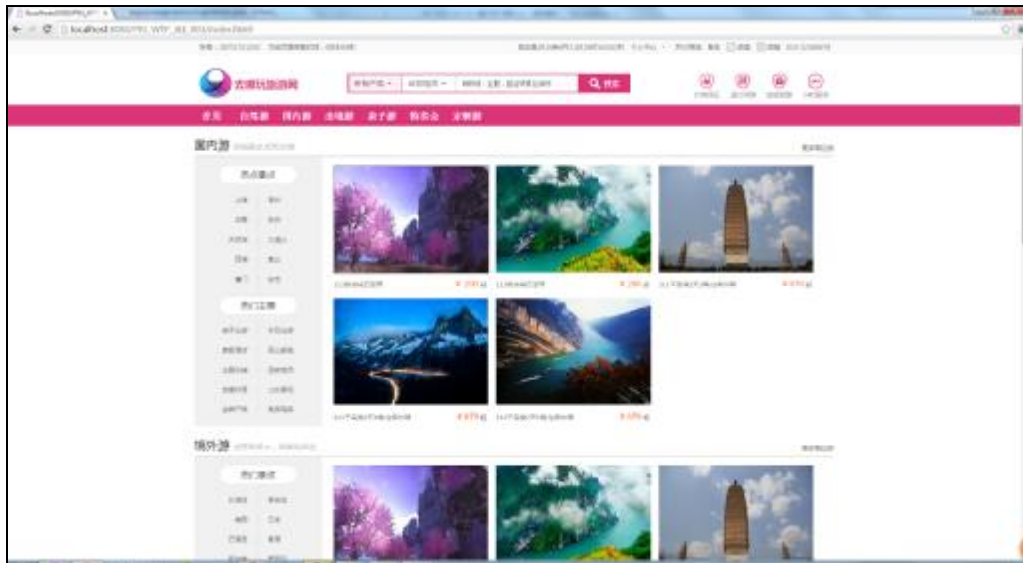


图 3-4-1

##### 1-1.2. 登录信息错误，显示的页面效果（login.html）：

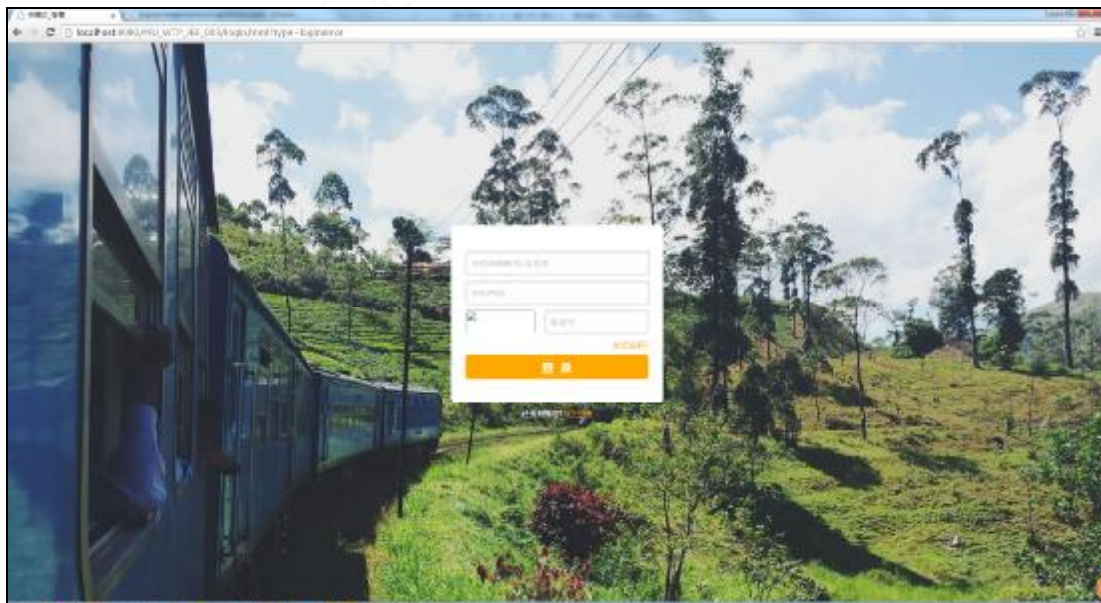


图 3-4-2

## 1-2. 任务目标：

1-2.1. 基于任务3，增加页面跳转连接的可维护性。

## 1-3. 任务要求：

1-3.1. 将登录成功的跳转页面URL：index.html，保存于web.xml文件中便于维护。

1-3.2. 将登录失败的跳转页面URL：login.html，保存于web.xml文件中便于维护。

1-3.3. 跳转连接前，优先从web.xml文件中读取连接，再实现页面跳转。

## 2. 实现思路：

2-1. 将登录业务涉及的跳转 URL 连接保存于 web.xml 文件中。

2-2. 正确与错误的跳转连接均可设置于<servlet>标签下的<init-param>标签中。

2-3. 在 LoginAction 的 init 方法中，利用 ServletConfig 组件读取跳转连接。

2-4. LoginAction 按读取的连接执行跳转操作。

2-5. 实现流程如下图：



图 3-4-3

## 3. 推荐步骤：

3-1. 在web.xml中设置跳转连接：

3-1.1. 在LoginAction的<servlet>标签下设置跳转页面的配置：<init-param>。

3-1.2. 登录正确的跳转连接可作如下配置：

1) <param-name>：index\_url；

2) <param-value>：index.html；

3-1.3. 登录失败的跳转连接可作如下配置：



1) <param-name> : error\_url ;

2) <param-value> : login.html ;

3-2. LoginAction读取跳转连接 :

3-2.1. 在LoginAction中重写父类的init方法 ;

3-2.2. 在init方法中, 可通过父类获取ServletConfig对象实例。

3-2.3. 通过ServletConfig对象获取正确跳转连接和错误跳转连接。

**+ 提示 :**

1) 重写init方法 :

1-1. init方法隶属于HttpServlet的父类GenericServlet。

1-2. 定位LoginAction类文件, 点击右键选择Source ➤ Override/Implement Methods

1-3. 在重写对话框中, 展开GenericServlet类结构, 选择init。

2) 获取web.xml文件中的跳转连接 :

2-1. ServletConfig对象可以获得web.xml文件中init-param中的数据。

2-2. GenericServlet类的getServletConfig()方法可以获取ServletConfig对象。

2-3. ServletConfig类的getInitParameter()方法可以获取web.xml文件中的数据。

3) init方法在Servlet的生命周期中永远只会被执行一次。

3-3. 重构页面跳转代码 :

3-3.1. 修改页面跳转语句, 语句可直接使用从web.xml文件中获取的URL。

**4. 验证与测试 :**

4-1. 运行项目工程

4-2. 登录成功跳转 :

4-2.1. 在登录页输入正确的用户名和密码, 可见【完成效果图1】 : 。

#### 4-3. 登录失败跳转：

4-3.1. 在登录页输入正确的用户名和密码，可见【完成效果图2】：。

##### + 小贴士：

##### 1) 为何需要增加连接跳转的维护性：

1-1. 跳转连接URL被直接编写在了LoginAction中，该方式对于系统的后期维护而言是非常不便捷的（代码会被编译，编译后的代码无法修改）。

1-2. 如果将跳转连接编写在文件中（例如：web.xml），那么文件中的连接可以随时修改（文件不会被编译）。

1-3. 系统发布后，如需维护跳转连接可以直接修改文件（例如：web.xml）。



## 场景总结

Q1. 请您谈谈Java Web应用程序开发过程中web.xml的作用。

- 1、web.xml是Java Web应用的核心文件，一个Web应用有且仅有一个web.xml文件。
- 2、web.xml是Java Web应用开发者与Web容器（例如：Tomcat）之间“沟通的”桥梁：
  - 1-1. Java Web应用开发者将客户端与服务端的通讯规则编写在web.xml中。
  - 1-2. Web容器则根据web.xml的配置规则将开发者的意图实现。

例如：开发者希望当用户输入URL：<http://10.31.44.158/Qunawan/index.html>时

服务器端组件：cn.campsg.qunawan.action.MainAction 能够负责处理本次用户的请求，并将处理结果响应给客户端，这样的意图需要被完整的配置在web.xml文件中告之Web容器。

Q2. 请您谈谈对于Servlet组件的理解，重点阐述一下Servlet的工作机制：

- 1、Servlet的作用与价值：
  - 1-1. Java Class要能够被称之为Servlet，必须继承：javax.servlet.http.HttpServlet
  - 1-2. Servlet是Java Web开发过程中最重要的组件，是唯一一个可以用来接收客户端请求，并对请求做出响应的组件。
- 2、Servlet的工作模式
  - 2-1. 创建Servlet：创建普通Java类，继承javax.servlet.http.HttpServlet
  - 2-2. 配置Servlet：在web.xml中设置，URL与Servlet之间的匹配关系。
  - 2-3. 寻找Servlet：客户端将请求发送到Web容器（Tomcat），容器从Web.xml中获取URL与Servlet的对应关系匹配当前URL，如果没有对应关系直接向客户端显示404错误。
  - 2-4. 实例化Servlet：Web容器（Tomcat）根据匹配到的Servlet类全名，搜索容器中是否存在该Servlet实例，如果没有新建一个，如果存在直接调用该Servlet中的回调函数。

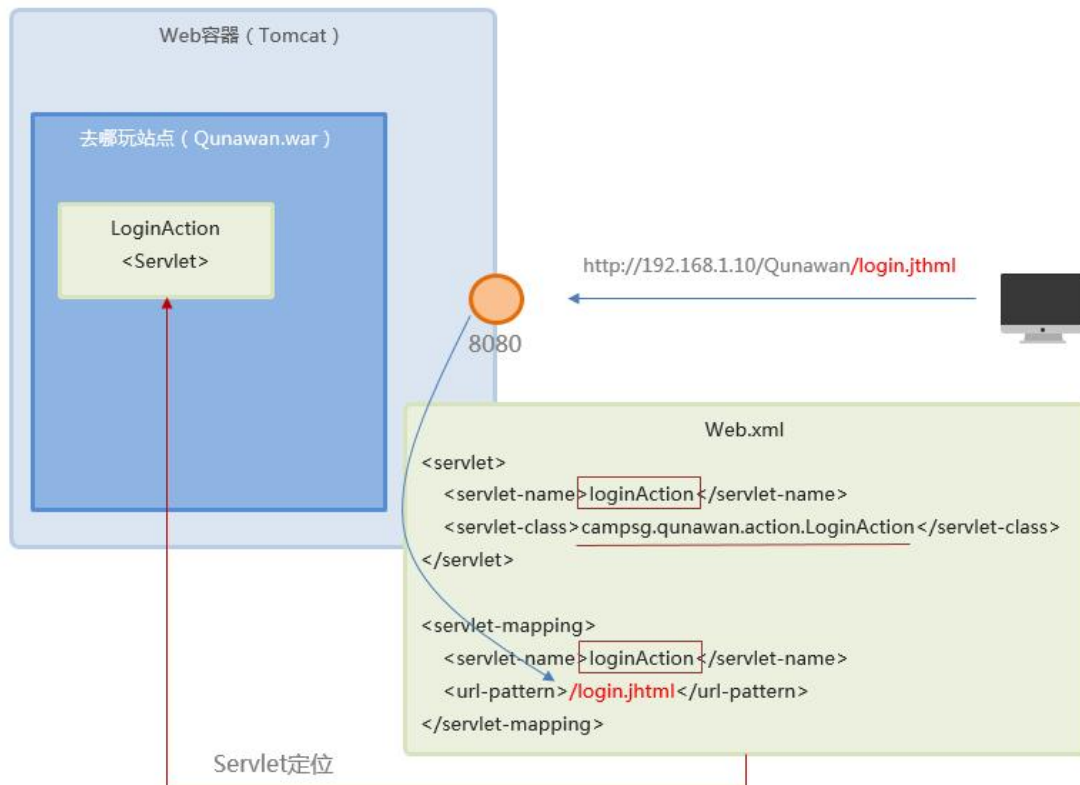


图 3-5-1

### 3、Servlet的运用场景和职责

#### 3-1. 实际生产环境中，Servlet主要的工作职责是：

- 3-1.1. 接收客户端请求。
- 3-1.2. 将请求的业务处理交给合适的业务组件处理与计算。
- 3-1.3. 将计算结果通过响应的方式发送到客户端。

#### 3-2. Servlet几乎可以处理所有类型的请求，例如：

- 3-2.1. 文件上传请求。
- 3-2.2. 文件下载请求。
- 3-2.3. 二进制文件处理请求（Excel、PDF文件生成与现实）。
- 3-2.4. 数据处理请求。
- 3-2.5. 其他类型的请求。

**Q3. 叙述您对于Servlet生命周期的理解，以及容器调用Servlet回调函数的流程：**

1、Servlet的生命周期：

1-1. 首次访问Servlet，Web容器负责创建Servlet并保存于容器中（防止被重复创建），新建Servlet时，Web容器负责调用Servlet的init回调函数。

所有Servlet初始化操作都应该写在init函数中，例如读取Web.xml中的init-param数据。

1-2. 实例化完，容器开始区分请求类型（Get/POST），并调用Servlet的service回调函数。

所有请求业务处理前的前置工作都应该写在service之中，例如：设置网站统一字符集。

1-3. Web容器根据类型区分结果，调用Servlet的doGet或者doPost回调函数处理请求。

doGet和doPost是处理请求的核心函数。

1-4. 当容器停止（Tomcat），容器会销毁所有Servlet，并调用Servlet的destroy回调函数。

如在Servlet中启动了线程，务必在destroy中销毁掉。

作者：Roger.Huang