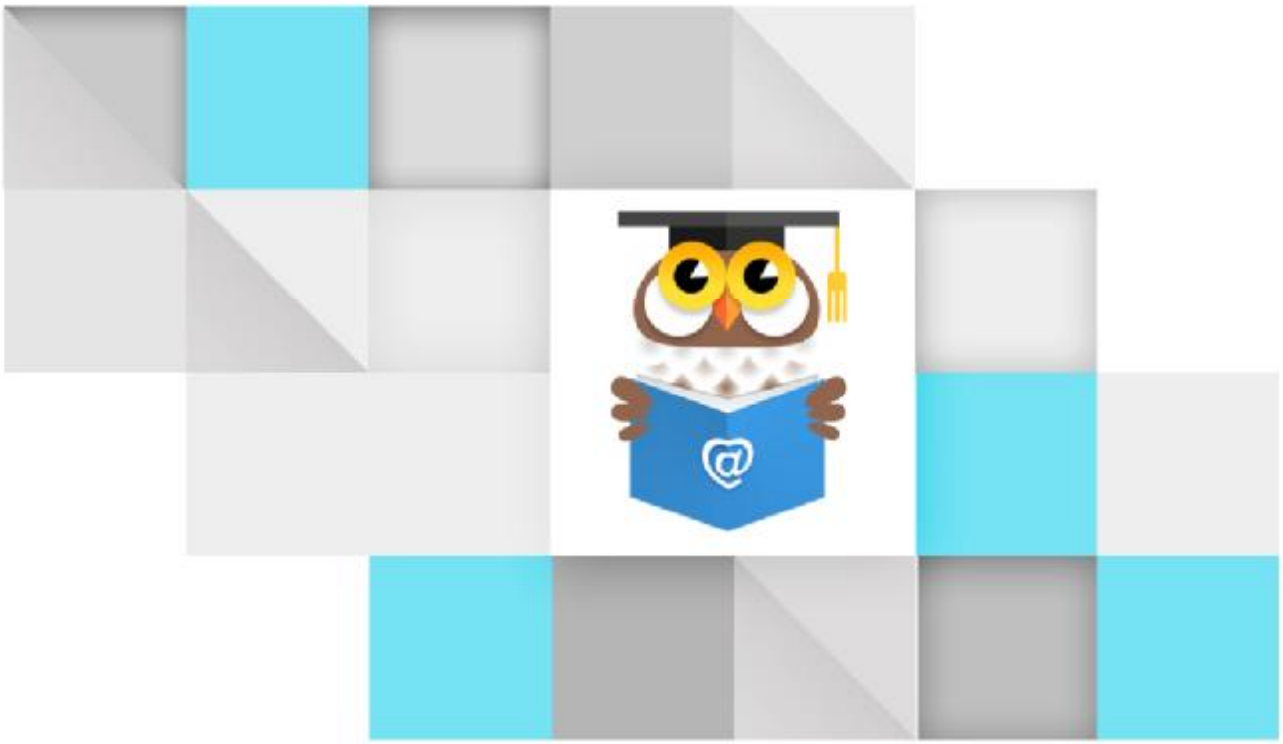
去哪玩旅游网（动态 WEB）

教学指导手册

PRJ-WTP-JEE-005 – 校验验证码



Campus Solution Group

目 录

一、场景说明

1、完成效果

图 **1-1-1**

2、业务描述

本场景用于整合：场景PRJ-WTP-JEE-003、场景PRJ-WTP-JEE-004的业务，实现先验证“验证码的有效性”，后验证用户名与密码的业务逻辑。

2-1. 在登录页**login.html**的验证码框中显示验证码图片。

2-1.1. 要求，刷新登录页时，可以重新显示新的验证码图片。

2-1.2. 要求，点击验证码时，可以重新显示新的验证码图片。

2-2. 在登录过程中优先验证验证码的有效性，再验证用户名和密码是否正确。

2-2.1. 验证码正确、用户名密码输入正确，显示站点首页（**index.html**）。

2-2.2. 验证码不正确，显示登录页（**login.html**）。

3、实现思路

3-1. 本场景建议将验证“验证码的有效性”分为二个任务依次实现：

3-1.1. 任务**1.** 在《去哪玩旅游网》的登录页面显示验证码图片。

3-1.2. 任务**2.** 在验证用户名和密码之前，优先验证验证码的有效性。

注意：本场景只考虑单个用户登录，多用户登录将在场景**015**中利用**Session**实现。

4、核心组件

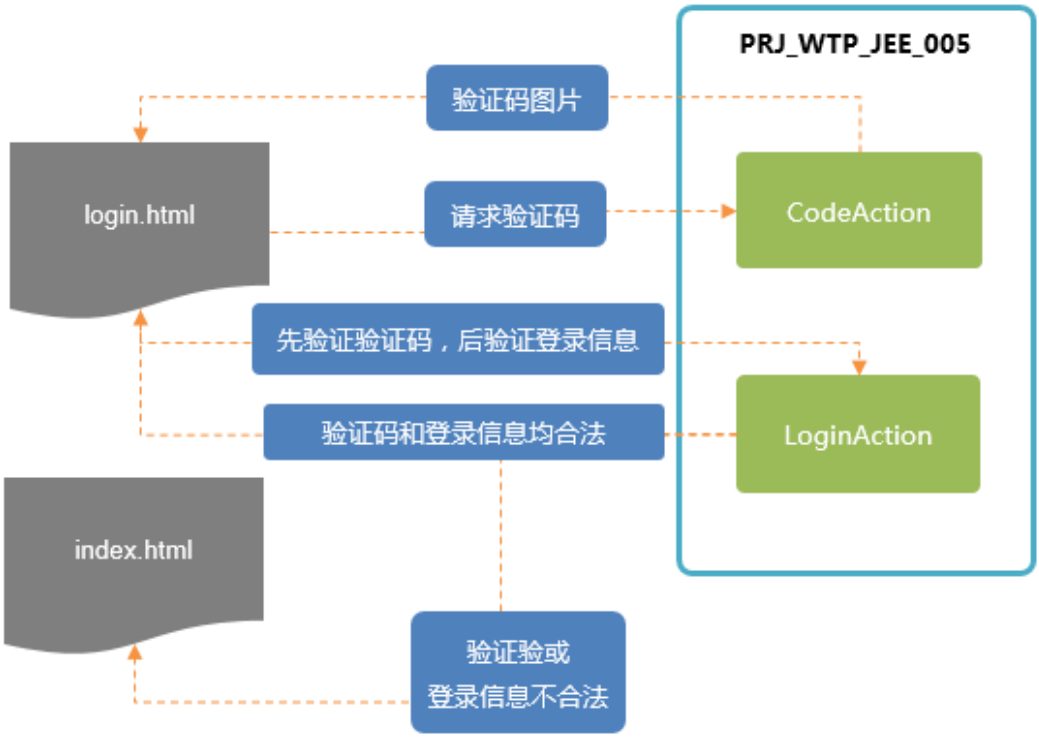


图 **1-4-1**

二、实训技能

1、重点演练

1-1. 理解完整的登录业务流程。

1-2. 理解验证码的使用、验证与存储逻辑。

2、相关技能

- **Servlet** 技术应用

3、相关知识点

- 开发 **Servlet** 组件
- 配置 **Servlet** 组件
- 接收客户端请求
- 获取表单数据

4、前置条件

4-1. 前置场景：**PRJ-WTP-JEE-004** 创建验证码

4-2. 必备知识与技能：

4-2.1. Servlet应用技术。

5、搭建环境

三、场景任务

任务 1、显示验证码图片

1. 任务说明：

1-1. 完成效果：

图 **3-1-1**

1-2. 任务目标：

1-2.1. 在《去哪玩旅游网》的登录页面显示验证码图片。

1-3. 任务要求：

1-3.1. 在《去哪玩旅游网》的登录页面显示验证码：

1） 确保每次加载登录页，验证码均会重新刷新并显示。

2） 确保每次点击验证码，验证码均会重新刷新并显示。

2. 实现思路：

2-1. 在登录页面：**login.html**中的**IMG**标签内，设置为获取验证码的**URL**。

2-1.1. IMG标签代码：**<img width="120px" height="40px">**

2-1.2. 将**IMG**标签的**src**属性，设置为获取验证码的**URL**。

说明：由场景**PRJ-WTP-JEE-004**可知，访问验证码的**URL**为：**/项目名/code.jhtml**

2-2. 利用**JS**代码实现：点击切换验证码：

2-2.1. 通过**IMG**标签的**onclick**事件，实现验证码切换。

2-2.2. **onclick**事件确保，**IMG**标签的**src**属性可以重新获取验证码图片。

小心：为确保每次获取新的验证码图片有效，在**onclick**事件中，应考虑在获取验证码图片的**URL**尾部增加一个客户端时间，以此保证每次**URL**的格式不重复。

2-3. 实现流程如下图所示：

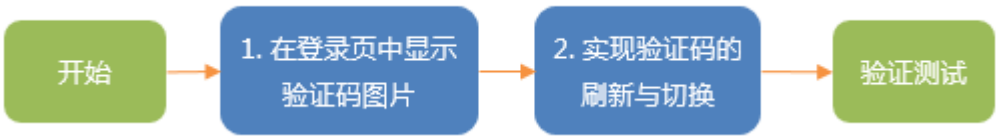


图 **3-1-2**

3. 推荐步骤：

3-1. 在**login.html**页中，设置获取验证码图片的**URL**，并显示。

3-2. 实现验证码的刷新与切换（代码临摹）：

3-2.1. 为**img**标签添加**onclick**事件，通过**JS**重新设置**img**标签的图片源（**src**属性）。

+ 代码临摹：

1) **onclick**事件内容：

```
javascript:this.src = '/PRJ_WTP_JEE_005/code.jhtml?id=' + new Date().getMil
liseconds()
```

+ 小贴士：

1）浏览器拥有缓存功能，对于相同的**URL**，浏览器会优先从缓存中获取，而不是向服务端获取。为了达到刷新验证码图片的目的，就要在**URL**后追加一个参数，确保每次参数值都不同，从而保证每次都能从服务端获取不同的验证码图片。

2） 本场景使用系统时间作为参数，确保每次**URL**都不同（**new Date().getMilliseconds()**）。

4. 验证与测试：

4-1. 运行项目工程。

4-2. 在**Chrome**浏览器中输入 **http://localhost/PRJ_WTP_JEE_005/** 并敲击回车。

4-3. 在登录页中看到验证码图片：

4-4. 点击验证码图片，确保验证码刷新。

任务 2、验证码存储与验证

1. 任务说明：

1-1. 完成效果：

1-1.1. 验证码正确、用户名密码输入正确，需显示的页面效果（**index.html**）：

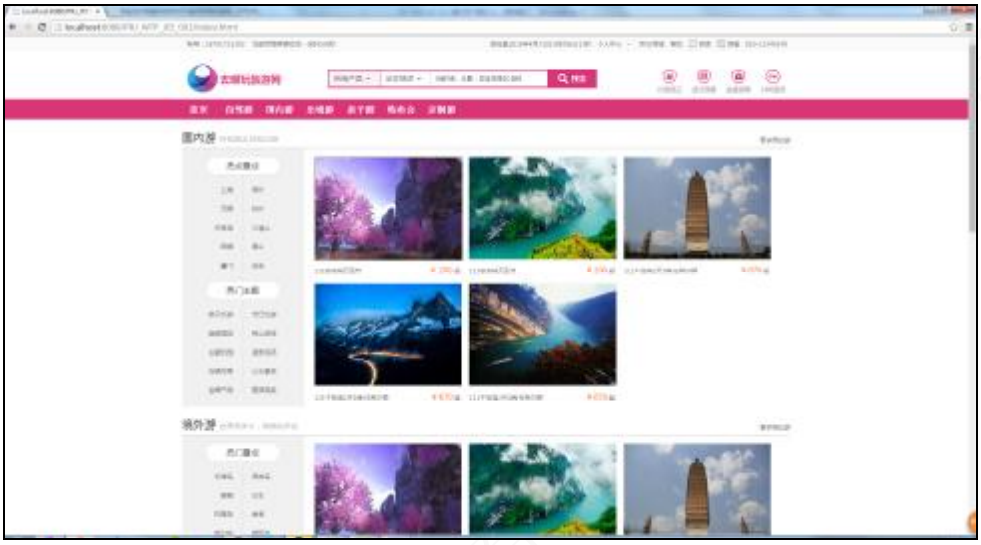


图 **3-2-1**

1-1.2. 验证码错误，需显示的页面效果（**login.html**）：

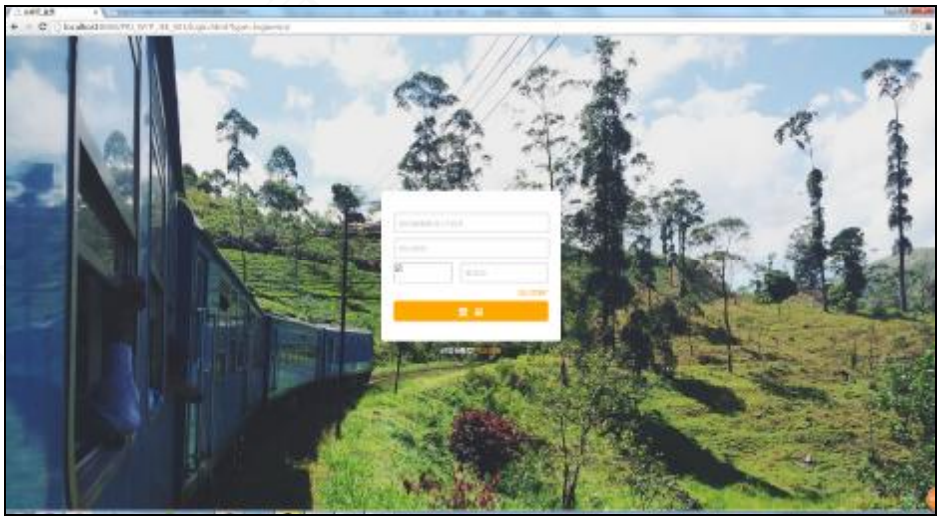


图 **3-2-2**

1-2. 任务目标：

1-2.1. 在验证用户名和密码之前，优先验证验证码的有效性。

1-3. 任务要求：

1-3.1. 在场景**PRJ-WTP-JEE-004**的基础之上，确保**CodeAction**创建的验证码字符串可以长时间被保留在服务端。

1-3.2. 在场景**PRJ-WTP-JEE-003**的基础之上，确保**LoginAction**在验证完验证码有效性后，再验证用户名和密码的有效性。

1） 如果验证码验证失败，则直接跳转到：**login.html**

2） 如果验证码验证正确，则进入用户名和密码的验证流程。

3） 限制**1.** 验证验证码时，应该忽略大小写问题。

2. 实现思路：

2-1. 将**CodeAction**生成的验证码字符保存起来，为验证验证码的有效性做准备。

2-1.1. 利用**Globle**组件保存**CodeAction**生成的验证码字符。

说明：**Globle**是当前系统自定义的一个临时容器，负责存放服务器端生成的验证码。

2-2. 在登录**Servlet**（**LoginAction**）中验证验证码输入有效性。

2-2.1. 从**Globle**中获取服务端保存的验证码字符。

2-2.2. 从请求对象中获取用户输入的验证码字符。

2-2.3. 比对后，验证码输入错误：直接跳转到：**login.html**。

2-2.4. 比对后，验证码输入正确：进入用户名和密码的验证流程。

2-3. 实现流程如下图所示：

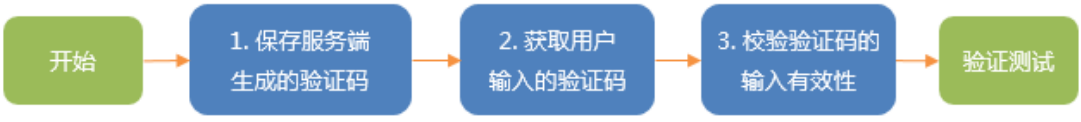


图 **3-2-3**

+ 小贴士：

1) 为何需要保存服务端生成的验证码字符串：

1-1. **HTTP**为无状态协议，当**Servlet**处理完请求后，业务流程计算所得的数据会丢失。

1-2. 每次业务计算所得的数据，需要通过**Web**容器中的作用域组件存储。

1-3. **Globle**就是当前系统自定义的临时作用域，负责存放服务器端生成的验证码。

3. 推荐步骤：

3-1. 保存服务端生成的验证码：

3-1.1. 在**CodeAction**的**doGet**方法中保存验证码字符串。

3-1.2. 在清空**codeNumbers**的代码之前，通过**Globle**保存验证码字符串。

提示：**Globle**的**setCode**静态函数，用于保存验证码字符串。

3-2. 获取用户输入的验证码。

3-3. 获取**Globle**中保存的验证码（服务器端生成的验证码）。

3-4. 校验验证码的输入有效性：

3-4.1. 判断用户输入的验证码和服务端保存的验证码是否一致：

1） 验证码错误：重定向到登录页面。

2） 验证码正确：进入到用户名和密码的校验流程。

4. 验证与测试：

4-1. 运行项目工程。

4-2. 在浏览器地址栏输入 **http://localhost/PRJ_WTP_JEE_005/** 并敲击回车进行访问

4-3. 登录成功跳转：

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**1**】：。

4-4. 登录失败跳转：

4-4.1. 在登录页输入正确的用户名和密码，可见【完成效果图**2**】：。

场景总结

Q1. 什么是浏览器缓存？它的作用是什么？

1. 浏览器缓存是为了加速浏览，浏览器在用户磁盘上对请求过的文档进行存储，当访问者再次请求这个页面时，浏览器就可以从本地磁盘显示文档，这样就可以提速。

2. 缓存对于静态媒体（例如：图片）有极好的加速浏览效果（无需每次都从服务器上下载大图）。

3. 但是对于会发生变动的媒体（例如：**js**、**css**、动态图片），缓存反而约束了其即时性。

4. 一个案例：

4-1. 网页中存在链接：**<javascript src=“/common/main.js” />**

浏览器加载该文件后会在本地缓存，再次访问页面时该文件不会再次下载，而如果网站更新了文件中的内容，浏览器同样不会响应修改，这样反而影响了网站的正常更新。

4-2. 如果修改成：**<javascript src=“/common/main.js？2016010115470001” />**

那么只需替换问号后的内容即能“骗过“浏览器，实现更新文件的操作。

5. 因此，浏览器缓存是默认行为，用途在于加速访问，如果业务需要不依赖于缓存，则必须保证网页中的**URL**能够实时更新，本场景就是利用这一特性实现了验证码的即时刷新。

作者：**Roger.Huang**