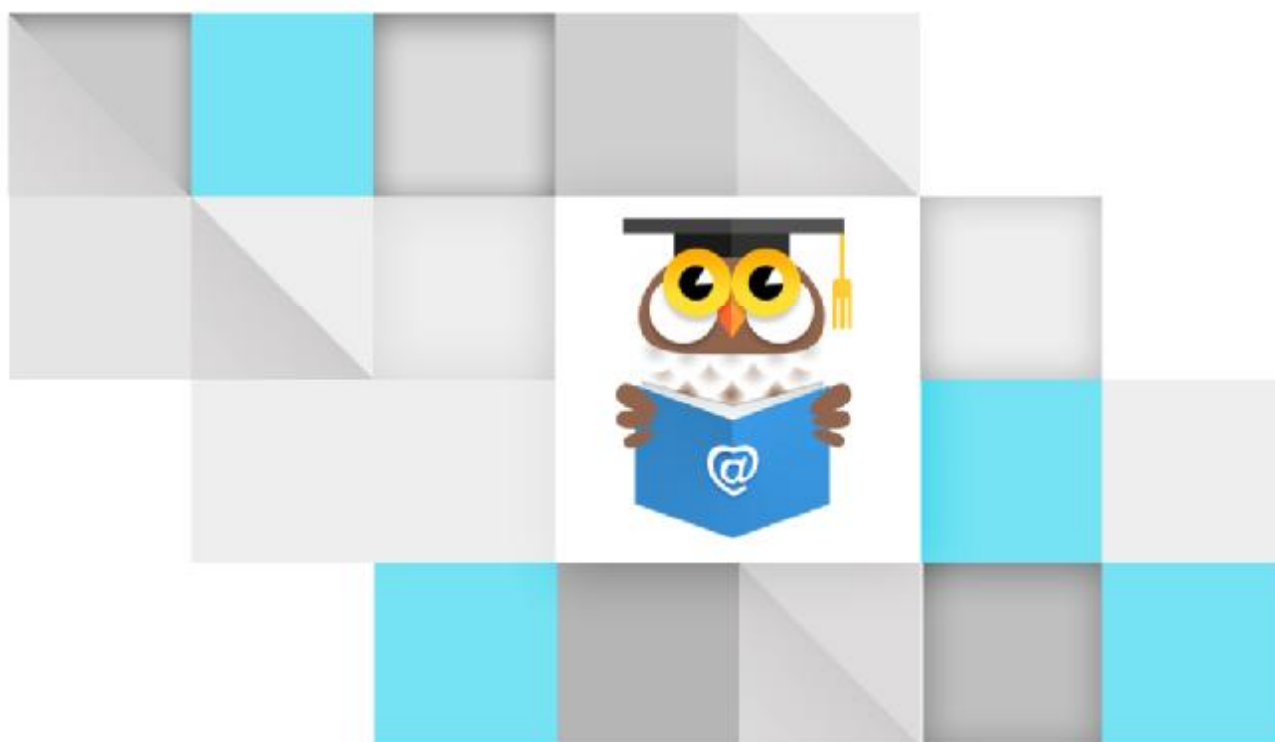


去哪玩旅游网（动态 WEB）

教学指导手册

PRJ-WTP-JEE-004 创建验证码（EX）



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 | 3 |
| 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、获取响应图片 | 7 |
| 任务 3、生成随机颜色 | 10 |
| 任务 4、绘制随机字符 | 13 |
| 任务 5、组建验证码字符串 | 16 |
| 场景总结 | 18 |

一、场景说明

1、完成效果

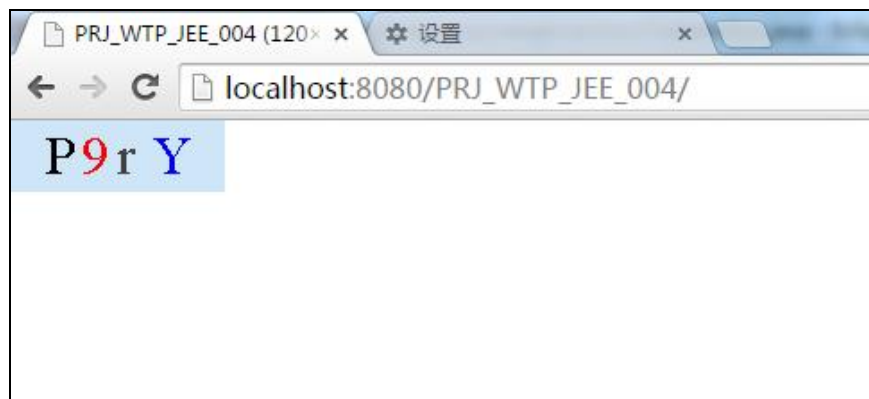


图 1-1-1

2、业务描述

本场景主要用于实现在客户端浏览器网页中，**动态验证码图片**的显示。

2-1. **验证码**一般由动态图片负责呈现，当前场景要求其具有以下特点：

2-1.1. 该动态图片的**背景色随机**。

2-1.2. 该动态图片需在“A-Z”，“a-z”，“0-9”之间随机显示**4个字符**。

2-1.3. 该动态图片能够在客户端显示。

3、实现思路

3-1. 本场景建议将“创建验证码图片”分为五个任务依次实现：

3-1.1. 任务1. 创建Servlet组件，该Servlet负责创建**动态验证码图片**。

3-1.2. 任务2. 在浏览器中呈现一张黑色背景，120 * 40大小的二进制图片。

1) 任务2用于体验二进制图片响应到客户端的方法与流程。

2) 任务2是绘制验证码的前置任务。

3-1.3. 任务3. 基于任务2，将验证码图片的背景色设置为随机色。

3-1.4. 任务4. 基于任务3，为验证码图片添加4个随机字符。

3-1.5. 任务5. 确保随机生成的4位随机字符，能够以字符串形式保留在服务端。

1) 任务5需要创建一个与图片字符对应的验证码字符串。

2) 该字符串用于记录完整的验证码字符，用于后续的登录验证。

4、核心组件

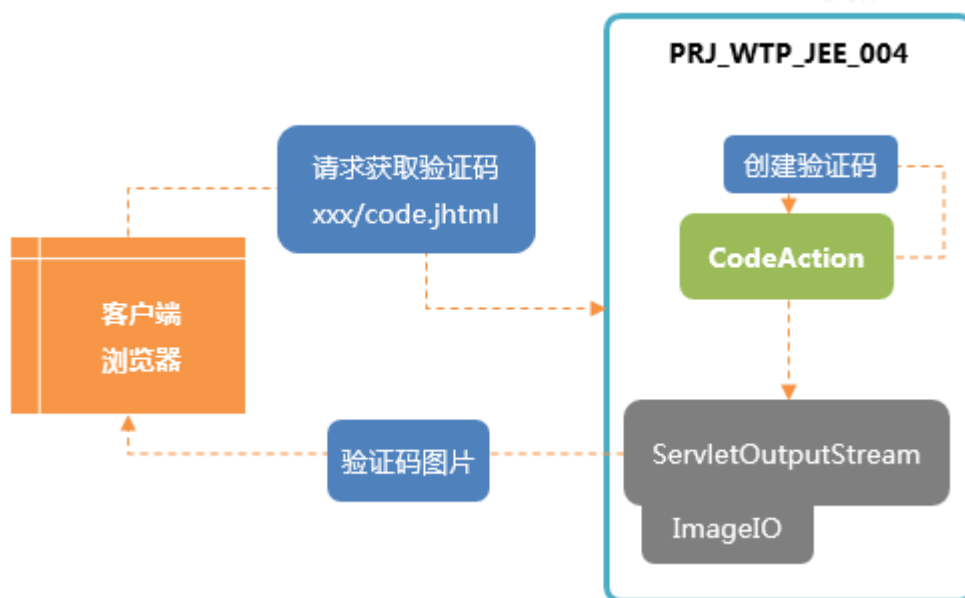


图 1-4-1

4-1. CodeAction (需实现) :

4-1.1. 用于绘制验证码的Servlet。

4-1.2. 验证码通过二进制流响应回客户端。

4-2. ServletOutputStream (需调用) :

4-2.1. Servlet用于响应二进制文件的流对象。

4-2.2. 图片、PDF、Excel、Word等非文本类数据均由该组件负责响应客户端。

4-3. ImageIO (需调用) :

4-3.1. 基于ServletOutputStream , 负责向客户端发送图片数据。

二、实训技能

1、重点演练

1-1. Servlet响应流对象 : ServletOutputStream。

1-2. 了解Java 2D编程技术 (可选)。

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 HttpServletResponse

I 获取响应消息流

I Java 2D 编程 (*)

1-3.

4、前置条件

4-1. 前置场景 : PRJ-WTP-JEE-003 – 模拟用户登录

4-2. 已学技能：

4-2.1. Java开发工具（Eclipse）。

4-2.2. Tomcat服务器。

4-2.3. Servlet应用技术。

5、搭建环境

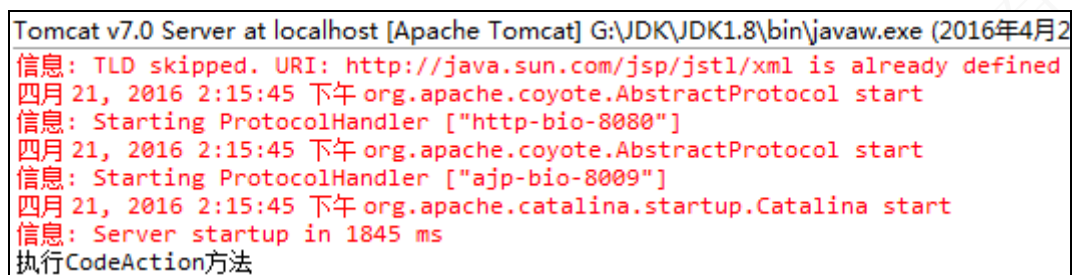


三、场景任务

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)
信息: TLD skipped. URI: http://java.sun.com/jsp/jstl/xml is already defined
四月 21, 2016 2:15:45 下午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
四月 21, 2016 2:15:45 下午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
四月 21, 2016 2:15:45 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 1845 ms
执行CodeAction方法
```

图 3-1-1

1-2. 任务目标：

1-2.1. 创建Servlet组件，该Servlet负责创建验证码。

1-3. 任务要求：

1-3.1. 创建用于绘制验证码图片的Servlet组件：

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

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

2. 实现思路：

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

2-2. 使CodeAction继承HttpServlet组件。

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

2-4. 为了便于测试，将code.jhtml设置为站点首页。

2-5. 在CodeAction的doGet中，编写控制台输出测试语句。

2-6. 实现流程如下图：



图 3-1-2

3. 推荐步骤：

3-1. 创建名为CodeAction的Servlet。

3-2. 配置CodeAction。

3-3. 将code.jhtml设置为首页。

4. 验证与测试：

4-1. 运行项目工程。

4-2. 在浏览器中输入 http://localhost:8080/PRJ_WTP_JEE_004/ 并敲击回车访问。

4-3. 控制台应显示“执行CodeAction方法”的字样。

任务 2、获取响应图片

1. 任务说明：

1-1. 完成效果：

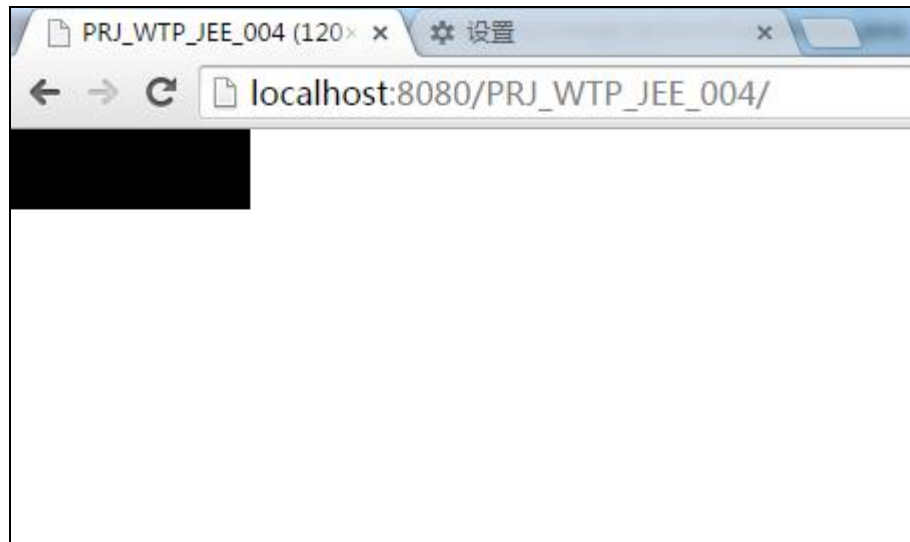


图 3-2-1

1-2. 任务目标：

1-2.1. 在浏览器中呈现一张黑色背景，120 * 40大小的二进制图片。

1-2.2. 任务2用于体验二进制图片响应到客户端的方法与流程。

1-2.3. 任务2是绘制验证码的前置任务。

1-3. 任务要求：

1-3.1. 通过CodeAction向客户端响应一张颜色固定、大小固定的图片。

1) 图片宽度：120px（像素）

2) 图片高度：40px（像素）

3) 图片背景：黑色（默认）。

1-3.2. 当客户端通过以code.jhtml结尾的URL访问站点时，即可看到该图片。

2. 实现思路：

- 2-1. 利用BufferedImage对象动态创建一张二进制图片。
- 2-2. 利用ServletOutputStream、ImageIO向客户端输出图片。
- 2-3. 实现流程如下图所示：

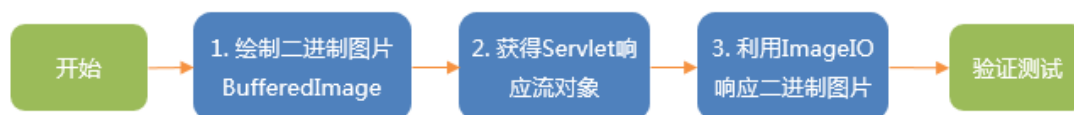


图 3-2-2

提示：任务2涉及部分Java2D GUI技术，如果用户没有学习过相关技术，建议按代码临摹步骤完成本任务。

3. 推荐步骤：

- 3-1. 创建一张颜色固定、大小固定的图片（代码临摹）
 - 3-1.1. 在CodeAction的doGet方法中编写二进制图片绘制代码。
 - 3-1.2. 通过BufferedImage对象，创建一张120 * 40大小的二进制图片。
- 3-2. 获取Servlet的二进制输出流ServletOutputStream的对象（代码临摹）。
- 3-3. 把图片对象通过二进制响应流传输给客户端（代码临摹）。
- 3-4. 关闭二进制输出流ServletOutputStream对象。（代码临摹）

注意：本处使用了Java的2D绘图技术，请按以下代码实现任务需求：

+ 代码临摹：

```
// 创建一张120 * 40大小的二进制图片对象
// 参数1：图片宽度； 参数2：图片高度； 参数3：图片类型（RGB模式）
BufferedImage image = new BufferedImage(Constants.IMAGE_WIDTH, Constants.IMAGE_HEIGHT, BufferedImage.TYPE_INT_RGB);

// 获取Servlet的二进制输出流ServletOutputStream的对象
ServletOutputStream sos = response.getOutputStream();

// 把图片对象通过二进制输出流响应给客户端
// 参数1：二进制图片对象； 参数2：图片格式； 参数3：输出到客户端的流对象。
```

```
ImageIO.write(image, "GIF", sos);
```

```
//关闭二进制输出流ServletOutputStream对象
```

```
sos.close();
```

+ 业务说明：

1) 验证码的一些基本信息已经封装在Constants类中，包括：

1-1. 验证码高度（整型）：Constants.IMAGE_HEIGHT

1-2. 验证码宽度（整型）：Constants.IMAGE_WIDTH

1-3. 字符颜色集（整型数组）：Constants.color

1-4. 字符字体集（整型数组）：Constants.codeFont

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:8080/PRJ_WTP_JEE_004/

4-3. 浏览器左上角出现黑色框（见任务完成效果图）。

任务 3、生成随机颜色

1、任务说明：

1-1. 完成效果：

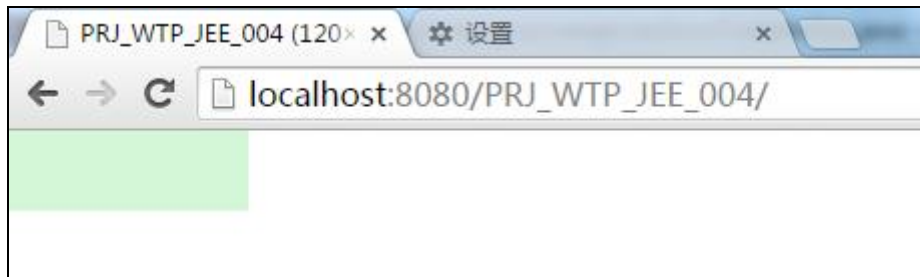


图 3-3-1

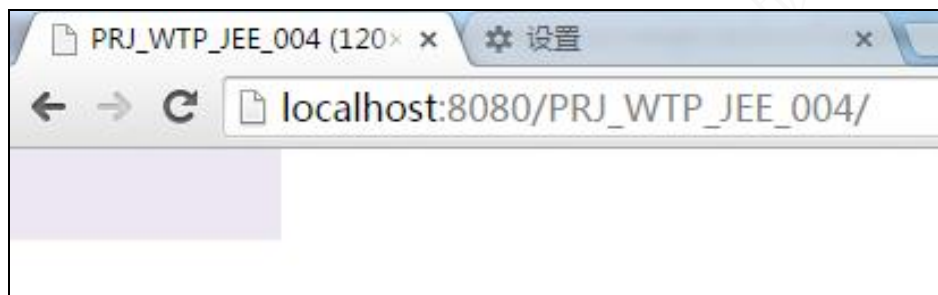


图 3-3-2

1-2. 任务目标：

1-2.1. 基于任务2，将验证码图片的背景色设置为随机色。

1-3. 任务要求：

1-3.1. 随机产生颜色值，并将其设置成验证码图片的背景色。

- 1) 限制1. 使用RGB色调值作为验证码的背景颜色。
- 2) 限制2. 必须创建随机色生成函数，命名：**getRandColor**

2、实现思路：

2-1. 在CodeAction中，创建RGB随机颜色函数：**getRandColor**。

2-1.1. getRandColor函数形式如下：

```
private Color getRandColor() {
```

```
}

```

2-1.2. 利用Random对象，分别产生3个0~255之间的随机数。

2-1.3. 3个随机数分别表示：蓝色（B）、红色（R）、绿色（G）。

2-1.4. 通过java.awt.Color将三个随机色组成Color对象，并返回。

2-2. CodeAction的doGet方法调用getRandColor，获取随机产生的颜色。

2-3. 通过Java 2D编程技术中的Graphics对象为验证码设置背景色。

2-4. 实现流程如下图所示：



图 3-3-3

+ 小贴士：

1) 所谓RGB颜色是指由蓝色（B）、红色（R）、绿色（G）三个数值叠加而成的颜色。

2) RGB色调的取值范围为0~255，数值越大颜色越淡，反之越深，例如：

2-1) 白色的RGB取值：255 255 255

2-2) 黑色的RGB取值：0 0 0

提示：任务3涉及部分Java2D GUI技术，如果用户没有学习过相关技术，建议按代码临摹步骤完成本任务。

3、推荐步骤：

3-1. 创建随机背景色函数：getRandColor

3-1.1. 在CodeAction类中，定义getRandColor函数。

3-2. 获取随机的RGB色调值：

3-2.1. 创建Random类的实例。

3-2.2. 连续随机获取3个0~255之间的随机数，作为红、绿、蓝色的色值；

3-2.3. 利用三个随机数实例化一个Color对象，并返回（利用Color的构造函数）。

3-3. 为验证码图片绘制随机背景色（代码临摹）：

3-3.1. 获得验证码的绘图对象：Graphics。

3-3.2. 为绘图对象Graphics上色。

3-3.3. 设置验证码的背景颜色。

注意：本处使用了Java的2D绘图技术，请按以下代码实现任务需求：

+ 代码临摹：

```
// 获取图形绘制对象：Graphics
// Graphics对象就是BufferedImage的画笔。
// 通过使用画笔就可以在BufferedImage上绘制图形、文字、颜色。
Graphics g = image.getGraphics();

// 调用getRandColor，为画笔上色
g.setColor(getRandColor());

// 使用画笔，填充验证码的背景色。
// 参数1-2：颜色填充的起始位置：图片的（0,0）坐标（即左上角）；
// 参数3-4：颜色填充范围（宽度，高度）。
g.fillRect(0, 0, Constants.IMAGE_WIDTH, Constants.IMAGE_HEIGHT);
```

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_004/ 并敲击回车进行访问。

4-3. 在浏览器的左上角显示验证码图片，通过刷新页面会显示其它不同颜色。

任务 4、绘制随机字符

1. 任务说明：

1-1. 完成效果：

1-1.1. 在页面中显示的验证码效果：

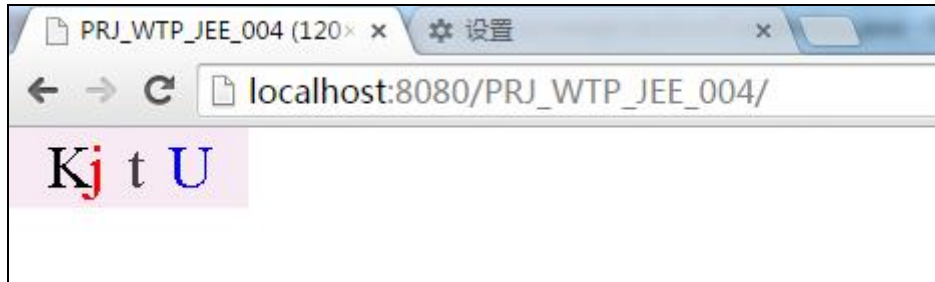


图 3-4-1

1-2. 任务目标：

1-2.1. 基于任务3，为验证码图片添加4个随机字符。

1-3. 任务要求：

1-3.1. 创建4个随机字符，取值范围：0~9 、 a~z 、 A~Z：

- 1) 将4个随机字符从**左向右**依次绘制在验证码上。
- 2) 限制1. 必须创建随机字符生成函数，命名：**drawCode**。
- 3) 限制2. 绘制字符时，字体颜色固定使用常量：Constants.**color**
- 4) 限制3. 绘制字符时，字体类型固定使用常量：Constants.**codeFont**
- 5) 限制4. 随机字符的取值范围，固定使用常量：Constants.**IMAGE_CHAR**

2. 实现思路：

2-1. 在CodeAction中，创建随机字符函数：**drawCode**。

2-1.1. **drawCode**函数形式如下：

```
// 参数1：验证码的字符绘制组件（任务2中已经创建）  
// 参数2：标识创建第几个随机字符
```

```
private void drawCode(Graphics graphics, int i) {  
  
}
```

2-1.2. drawCode函数负责创建随机字符。

2-1.3. drawCode函数负责将随机字符绘制于验证码图片中。

2-2. 在CodeAction的doGet方法中，连续四次调用getRandColor，创建随机字符。

2-3. 实现流程如下图所示：

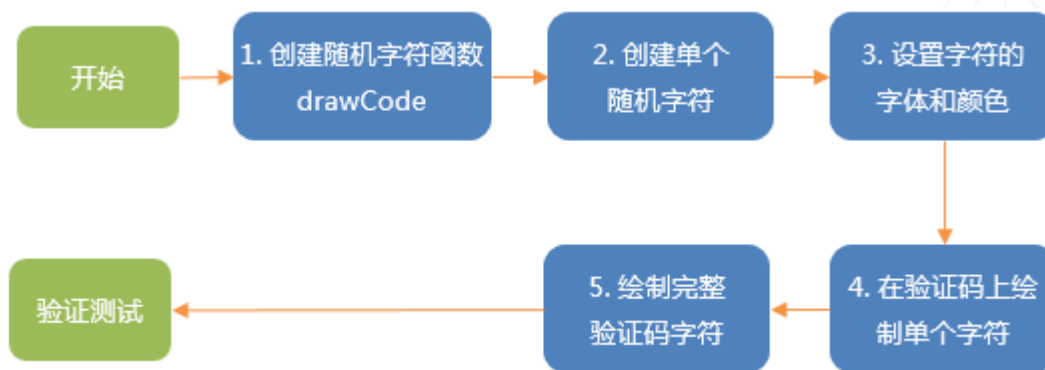


图 3-4-2

提示：任务4涉及部分Java2D GUI技术，如果用户没有学习过相关技术，建议按代码临摹步骤完成本任务。

3. 推荐步骤：

3-1. 创建随机字符函数：drawCode

3-1.1. 在CodeAction类中，定义drawCode函数。

3-2. 创建单个随机字符：

3-2.1. 创建Random类的实例。

3-2.2. 在0~9、a~z、A~Z之间随机获取一个字符。

1) 使用Constants.IMAGE_CHAR作为随机取值范围。

2) 从Constants.IMAGE_CHAR中随机截取1位字符。

3-3. 设置字符的字体类型和字体颜色（代码临摹）。

3-4. 在图片上绘制单个字符（代码临摹）。

注意：本处使用了Java的2D绘图技术，请按以下代码实现任务需求：

+ 代码临摹：

```
// 设置验证码字体
graphics.setFont(Constants. codeFont[i]);

// 设置验证码颜色
graphics.setColor(Constants. color[i]);

// 在验证码图片上绘制随机字符
// 参数1：待写的字符（随机获取的字符）；
// 参数2：写入到验证码的横坐标（X）位置；
// 参数3：写入到验证码的纵坐标（Y）位置；
// 说明1：为确保写入图片的字符不发生重叠，因此所有字符的横坐标（X）位置是需要变化的。
// 说明2：本任务要求字符与字符之间的间隔为20个px，因此横坐标的计算公式：20 + i * 20。
graphics.drawString(number, 20 + i * 20, 30);
```

3-5. 绘制完整验证码字符：

3-5.1. 定位到doGet方法，在设置背景颜色代码（以下语句）之后。

```
g.fillRect(0, 0, Constants. IMAGE_WIDTH, Constants. IMAGE_HEIGHT);
```

3-5.2. 调用drawCode4次，依次创建4个随机字符。

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_004/；

4-3. 在浏览器左上角显示生成的验证码图片；

任务 5、组建验证码字符串

1、任务说明：

1-1. 完成效果：

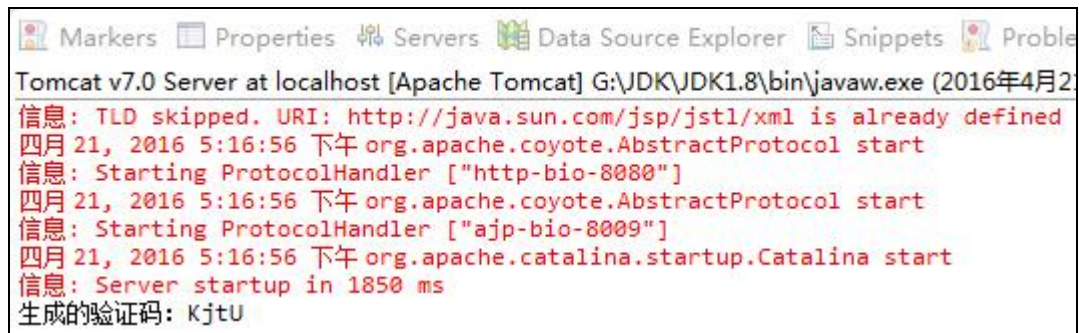


图 3-5-1

1-2. 任务目标：

1-2.1. 确保随机生成的4位随机字符，能够以字符串形式保留在服务端中。

1) 任务5，需要创建一个与图片字符对应的验证码字符串。

2) 该字符串用于记录完整的验证码字符，用于后续的登录验证。

1-3. 任务要求：

1-3.1. 创建一个用于记录验证码字符串的变量：

1) 限制1. 变量为字符类型，取名：`codeNumbers`。

1-3.2. 基于任务4，每一个随机字符均需要合并入`codeNumbers`。

1-3.3. 4个随机字符创建完毕后，在控制台输入验证码字符串，格式如下：

生成的验证码：验证码字符串内容

2、实现思路：

2-1. 在CodeAction中，创建用于记录4位随机字符的成员变量：`codeNumbers`。

2-2. 在drawCode函数中，将每次创建的随机字符合并入`codeNumbers`变量中。

2-3. 随机字符创建完毕，向控制台输出完整的验证码字符串。

2-3.1. 清空`codeNumbers` , 防止客户端多次请求验证码时随机字符被重复记录。

2-4. 实现流程如下图所示：

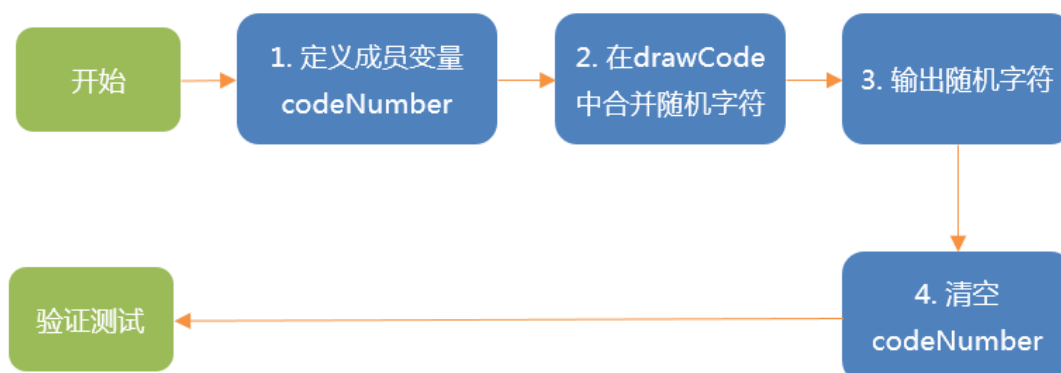


图 3-5-2

3、推荐步骤：

3-1. 组建验证码字符串：

3-1.1. 在CodeAction类中定义一个成员变量：`codeNumbers`：

3-1.2. 定位drawCode，把生成的随机字符number合并入`codeNumbers`中。

3-2. 定位doGet方法，在4次调用drawCode函数的代码后，向控制台输出验证码字符串。

3-3. 最后，将记录验证码字符串变量codeNumbers置空；

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_004/；

4-3. 控制台显示验证码的字符串内容。

场景总结

Q1. 请您谈谈Servlet响应流：ServletOutputStream的作用与使用场景。

1. Servlet是J2EE中的核心组件，它负责接收请求，寻找业务处理组件，响应结果到客户端。
2. Servlet具有多种响应模式，常见的有网页字符响应形式和纯二进制数据的响应形式：
 - 2-1. 响应网页模式，可以使用PrintWriter组件。
 - 2-2. 响应二进制数据，可以使用ServletOutputStream组件。
3. 二进制数据响应客户端极为常见，例如：显示一张动态图片（验证码）、一个PDF文件、一个Word文档、一份Excel报表等等，因此ServletOutputStream组件非常常用。
4. 值得注意的是：二进制数据响应方式依赖于响应头中的content-type属性，它用于告诉客户端响应体的数据类型。该属性需要以MIME类型方式呈现，常见的MIME类型有：
 - 4-1. text/html: 纯网页形式。
 - 4-2. image/gif: 图片形式。
 - 4-3. text/xml: 常见于Ajax响应形式。
 - 4-4. application/vnd.ms-excel: 文件下载以EXCEL格式打开
 - 4-5. application/octet-Stream: 文件下载并保存搭配客户端
 - 4-6. video/mpeg: 视频传输以MPEG编码格式打开

作者：Roger.Huang