

Spring Boot实践，开发社区登录模块

牛客Java高级工程师 第二章

► 1. 发送邮件

- 邮箱设置
 - 启用客户端SMTP服务
- Spring Email
 - 导入 jar 包
 - 邮箱参数配置
 - 使用 JavaMailSender 发送邮件
- 模板引擎
 - 使用 Thymeleaf 发送 HTML 邮件

Table of Contents

[◀ Back to index](#)

1. Remoting and Web Services with Spring
2. Enterprise JavaBeans (EJB) Integration
3. JMS (Java Message Service)
4. JMX
5. JCA CCI
- 6. Email**
 - 6.1. Usage
 - 6.2. Using the JavaMail MimeMessageHelper
7. Task Execution and Scheduling
8. Cache Abstraction
9. Appendix

► 2. 开发注册功能

- 访问注册页面
 - 点击顶部区域内的链接，打开注册页面。
- 提交注册数据
 - 通过表单提交数据。
 - 服务端验证账号是否已存在、邮箱是否已注册。
 - 服务端发送激活邮件。
- 激活注册账号
 - 点击邮件中的链接，访问服务端的激活服务。



注册

账号: 请输入您的账号!

密码: 请输入您的密码!

确认密码: 请再次输入密码!

邮箱: 请输入您的邮箱!

立即注册

▶ 3. 会话管理

- HTTP的基本性质

- HTTP是简单的
- HTTP是可扩展的
- HTTP是无状态的，有会话的

- Cookie

- 是服务器发送到浏览器，并保存在浏览器端的一小块数据。
- 浏览器下次访问该服务器时，会自动携带该数据，将其发送给服务器。

- Session

- 是JavaEE的标准，用于在服务端记录客户端信息。
- 数据存放在服务端更加安全，但是也会增加服务端的内存压力。

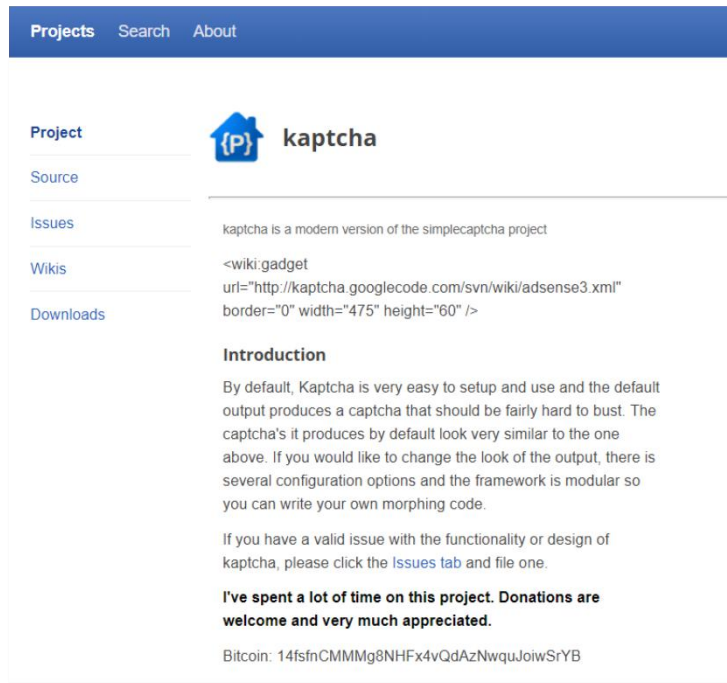
HTTP 是无状态，有会话的

HTTP是无状态的：在同一个连接中，两个执行成功的请求之间是没有关系的。这就带来了一个问题，用户没有办法在同一个网站中进行连续的交互，比如在一个电商网站里，用户把某个商品加入到购物车，切换一个页面后再次添加了商品，这两次添加商品的请求之间没有关联，浏览器无法知道用户最终选择了哪些商品。而使用HTTP的头部扩展，HTTP Cookies就可以解决这个问题。把Cookies添加到头部中，创建一个会话让每次请求都能共享相同的上下文信息，达成相同的状态。

► 4. 生成验证码

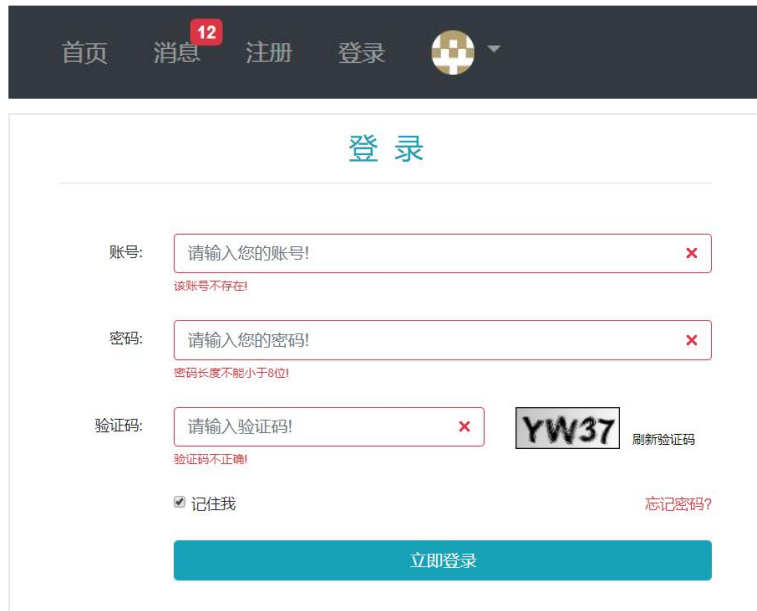
- Kaptcha
 - 导入 jar 包
 - 编写 Kaptcha 配置类
 - 生成随机字符、生成图片

<https://code.google.com/archive/p/kaptcha>



► 5. 开发登录、退出功能

- 访问登录页面
 - 点击顶部区域内的链接，打开登录页面。
- 登录
 - 验证账号、密码、验证码。
 - 成功时，生成登录凭证，发放给客户端。
 - 失败时，跳转回登录页。
- 退出
 - 将登录凭证修改为失效状态。
 - 跳转至网站首页。



The screenshot shows a web application's login interface. At the top is a dark navigation bar with links for '首页' (Home), '消息' (Messages) with a red badge showing '12', '注册' (Register), and '登录' (Login). A user profile icon is on the right. The main content area has a title '登 录' (Login) in blue. Below it are three input fields, each with a red border and a red 'x' icon on the right. The first field is labeled '账号:' (Account) and has the error message '该账号不存在!' (This account does not exist!). The second field is labeled '密码:' (Password) and has the error message '密码长度不能小于8位!' (Password length cannot be less than 8 characters!). The third field is labeled '验证码:' (Captcha) and has the error message '验证码不正确!' (Captcha is incorrect!). To the right of the captcha field is a box showing 'YW37' and a link '刷新验证码' (Refresh captcha). Below the fields is a checkbox labeled '记住我' (Remember me) and a link '忘记密码?' (Forgot password?). At the bottom is a large blue button labeled '立即登录' (Login immediately).

► 6. 显示登录信息

- 拦截器示例
 - 定义拦截器，实现HandlerInterceptor
 - 配置拦截器，为它指定拦截、排除的路径
- 拦截器应用
 - 在请求开始时查询登录用户
 - 在本次请求中持有用户数据
 - 在模板视图上显示用户数据
 - 在请求结束时清理用户数据



► 7. 账号设置

- 上传文件
 - 请求：必须是POST请求
 - 表单：enctype= "multipart/form-data"
 - Spring MVC：通过 MultipartFile 处理上传文件
- 开发步骤
 - 访问账号设置页面
 - 上传头像
 - 获取头像

The screenshot displays a web interface for account settings. It is divided into two main sections: '上传头像' (Upload Avatar) and '修改密码' (Change Password). The '上传头像' section includes a label '选择头像:', a text input field with the placeholder '选择一张图片', a '文件' (File) button, and a large teal '立即上传' (Upload Immediately) button. The '修改密码' section contains three input fields: '原密码:' (Original Password) with placeholder '请输入原始密码!', '新密码:' (New Password) with placeholder '请输入新的密码!', and '确认密码:' (Confirm Password) with placeholder '再次输入新密码!'. A large teal '立即保存' (Save Immediately) button is at the bottom of this section.

► 8. 检查登录状态

- 使用拦截器
 - 在方法前标注自定义注解
 - 拦截所有请求，只处理带有该注解的方法
- 自定义注解
 - 常用的元注解：
`@Target`、`@Retention`、`@Document`、`@Inherited`
 - 如何读取注解：
`Method.getDeclaredAnnotations()`
`Method.getAnnotation(Class<T> annotationClass)`

Thanks

