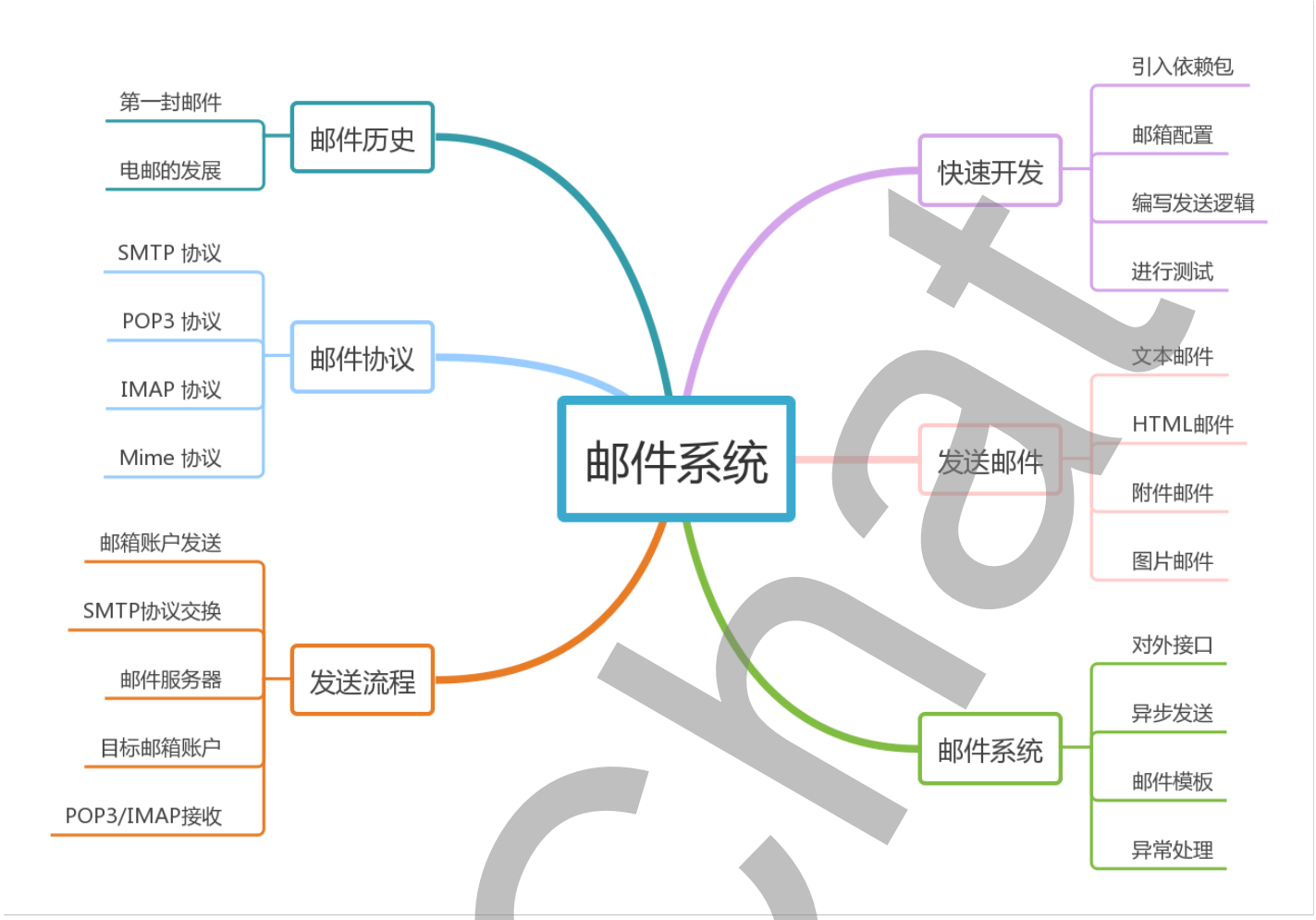
第 4-10 课：使用 Spring Boot 开发邮件系统

电子邮件是在因特网上使用的非常多的一种应用，它可以非常方便的让相隔很远的人进行通信，主要特点是操作简单、快捷。现在的电子邮件系统是以存储与转发的模型为基础，邮件服务器接收、转发、提交及存储邮件，寄信人、收信人及他们的计算机都不用同时在线，寄信人和收信人只需在寄信或收信时简短的连线到邮件服务器即可。

互联网发展到现在，邮件服务已经成为互联网企业中必备功能之一，应用场景非常广泛，比较常见的有：用户注册、忘记密码、监控提醒、企业营销等。大多数互联网企业都会将邮件发送抽取为一个独立的微服务，对外提供接口来支持各种类型的邮件发送。

本课内容将会从以下几部分来介绍如何开发一个邮件系统：

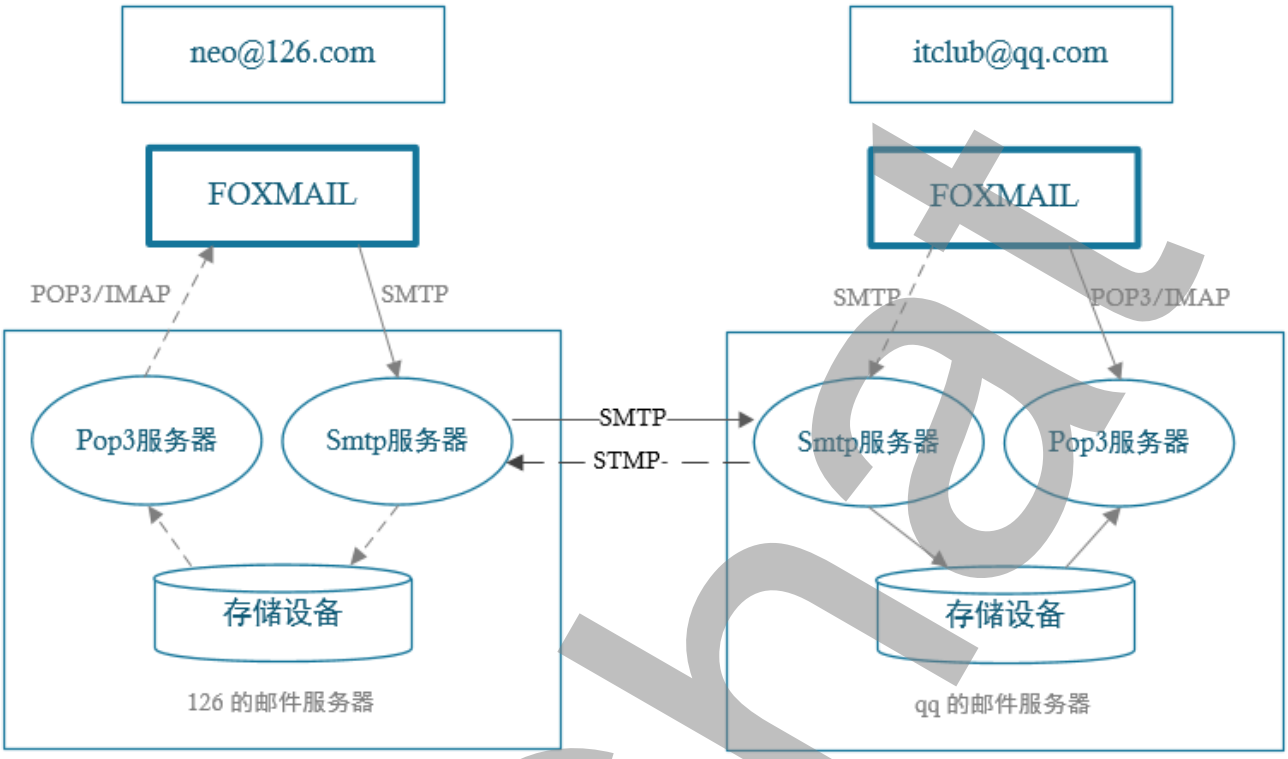
- 电子邮件的历史
- 发送邮件涉及到哪些协议
- 介绍一个完整的邮件发送流程
- 快速体验邮件发送流程
- 介绍如何开发文本、HTML、附件、图片的邮件
- 做一个邮件系统需要考虑的因素



邮件历史

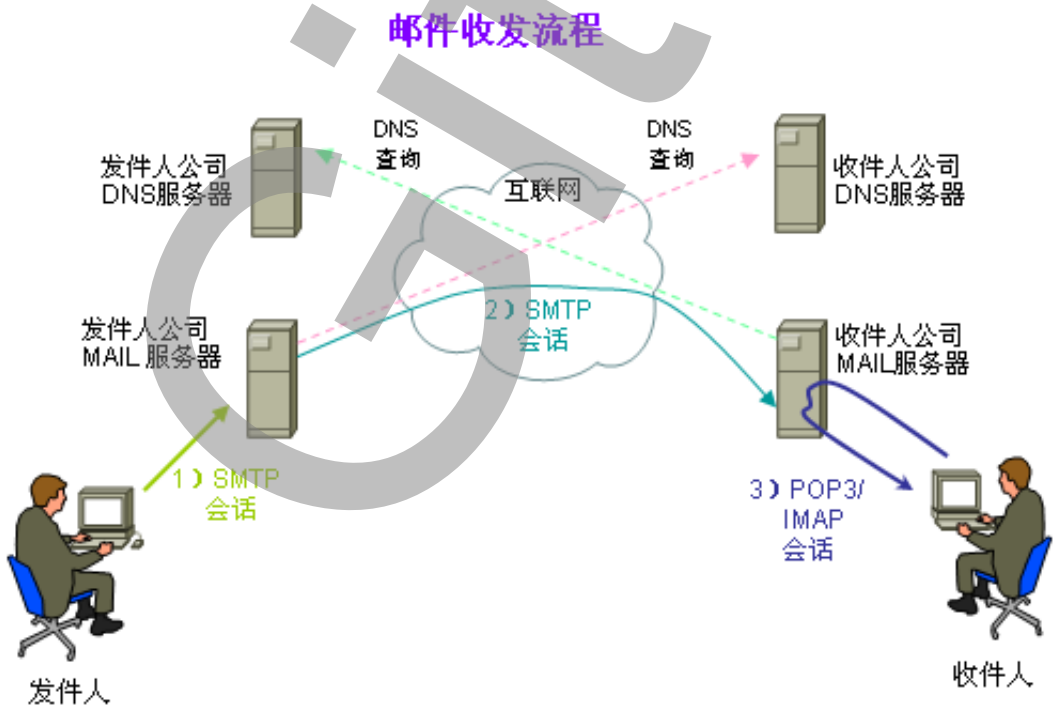
关于整个邮件的发展历史，比如“电子邮件的发展、世界的第一封电子邮件、中国的第一封电子邮件”的资料可点击链接查看，这里不作陈述。

用一张图来看看发送邮件过程中的协议使用：



> 实线代表 neo@126.com 发送邮件给 itclub@aa.com；虚线代表 itclub@aa.com 发送邮件给 neo@126.com。

邮件发送流程



- 发信人在用户代理上编辑邮件，并写清楚收件人的邮箱地址；
- 用户代理根据发信人编辑的信息，生成一封符合邮件格式的邮件；
- 用户代理把邮件发送到发信人的邮件服务器上，邮件服务器上面有一个缓冲队列，发送到邮件服务器上面的邮件都会加入到缓冲队列中，等待邮件服务器上的 SMTP 客户端进行发送；
- 发信人的邮件服务器使用 SMTP 协议把这封邮件发送到收件人的邮件服务器上；
- 收件人的邮件服务器收到邮件后，把这封邮件放到收件人在这个服务器上的信箱中；
- 收件人使用用户代理来收取邮件，首先用户代理使用 POP 3 协议来连接收件人所在的邮件服务器，身份验证成功后，用户代理就可以把邮件服务器上面的收件人邮箱里面的邮件读取出来，并展示给收件人。

这就是邮件发送的一个完整流程。

简单使用

最早期的时候使用 JavaMail 的相关 API 来开发，需要自己去封装消息体，代码量比较庞大；后来 Spring 推出了 JavaMailSender 来简化邮件发送过程，JavaMailSender 提供了强大的邮件发送功能，可支持各种类型的邮件发送。

现在 Spring Boot 在 JavaMailSender 的基础上又进行了封装，就有了现在的 spring-boot-starter-mail，让邮件发送流程更加简洁和完善。下面将介绍如何使用 Spring Boot 发送邮件。

pom 包配置

引入加 spring-boot-starter-mail 依赖包：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
    </dependency>
</dependencies>
```

配置文件

在 application.properties 中添加邮箱配置，不同的邮箱参数稍有不同，下面列举几个常用邮箱配置。

163 邮箱配置：

```
spring.mail.host=smtp.163.com //邮箱服务器地址
spring.mail.username=xxx@oo.com //用户名
spring.mail.password=xxyyooo    //密码
spring.mail.default-encoding=UTF-8

//超时时间，可选
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=3000
spring.mail.properties.mail.smtp.writetimeout=5000
```

126 邮箱配置：

```
spring.mail.host=smtp.126.com
spring.mail.username=yourEmail@126.com
spring.mail.password=yourPassword
spring.mail.default-encoding=UTF-8
```

QQ 邮箱配置如下：

```
spring.mail.host=smtp.qq.com
spring.mail.username=ityouknow@qq.com
spring.mail.password=yourPassword
spring.mail.default-encoding=UTF-8
```

> 注意：测试时需要将 spring.mail.username 和 spring.mail.password 改成自己邮箱对应的登录名和密码，这里的密码不是邮箱的登录密码，是开启 POP 3 之后设置的客户端授权密码。

这里以 126 邮件举例，有两个地方需要在邮箱中设置。

开启 POP 3 / SMTP 服务、IMAP / SMTP 服务

图片下方会有 SMTP 等相关信息的配置提示。

开通设置客户端授权密码

设置客户端授权密码一般需求手机验证码验证。

文本邮件发送

Spring 已经帮我们内置了 JavaMailSender，直接在项目中引用即可，封装一个 MailService 类来实现普通的邮件发送方法。

```
@Component
public class MailServiceImpl implements MailService{

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private JavaMailSender mailSender;

    @Value("${spring.mail.username}")
    private String from;

    @Override
    public void sendSimpleMail(String to, String subject, String content) {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom(from);
        message.setTo(to);
        message.setSubject(subject);
        message.setText(content);

        try {
            mailSender.send(message);
            logger.info("简单邮件已经发送。");
        } catch (Exception e) {
            logger.error("发送简单邮件时发生异常！", e);
        }

    }
}
```

> 文本邮件抄送使用：message.copyTo(copyTo) 来实现。

- from，即为邮件发送者，一般设置在配置文件中
- to，邮件接收者，此参数可以为数组，同时发送多人
- subject，邮件主题
- content，邮件的主体

邮件发送者 from 一般采用固定的形式写到配置文件中。

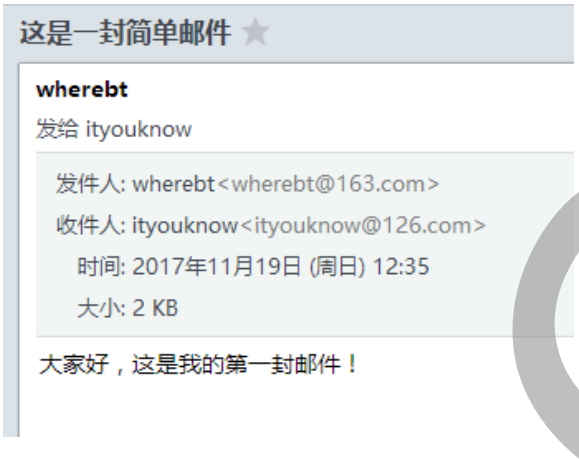
编写 test 类进行测试

```
@RunWith(SpringRunner.class)
@Spring BootTest
public class MailServiceTest {

    @Autowired
    private MailService MailService;

    @Test
    public void testSimpleMail() throws Exception {
        mailService.sendSimpleMail("ityouknow@126.com","这是一封简单邮件","大家好，这是
我的第一封邮件！");
    }
}
```

稍微等待几秒，就可以在邮箱中找到此邮件内容了，至此一个简单的文本邮件发送就完成了。



富文本邮件

在日常使用的过程中，我们通常在邮件中加入图片或者附件来丰富邮件的内容，下面将介绍如何使用 Spring Boot 来发送富文本邮件。

发送 HTML 格式邮件

邮件发送支持以 HTML 语法去构建自定义的邮件格式，Spring Boot 支持使用 HTML 发送邮件。

我们在 MailService 中添加支持 HTML 邮件发送的方法：

```
public void sendHtmlMail(String to, String subject, String content) {
    MimeMessage message = mailSender.createMimeMessage();

    try {
        //true 表示需要创建一个 multipart message
        MimeMessageHelper helper = new MimeMessageHelper(message, true);
        helper.setFrom(from);
        helper.setTo(to);
        helper.setSubject(subject);
        helper.setText(content, true);

        mailSender.send(message);
        logger.info("html邮件发送成功");
    } catch (MessagingException e) {
        logger.error("发送html邮件时发生异常! ", e);
    }
}
```

> 富文本邮件抄送使用：helper.addCc(cc) 来实现。

和文本邮件发送代码对比，富文本邮件发送使用 MimeMessageHelper 类，该类支持发送复杂邮件模板，支持文本、附件、HTML、图片等，接下来会一一使用到。

在测试类中构建 HTML 内容，测试发送：

```
@Test
public void testHtmlMail() throws Exception {
    String content="<html>\n" +
            "<body>\n" +
            "    <h3>hello world ! 这是一封html邮件!</h3>\n" +
            "</body>\n" +
            "</html>";
    mailService.sendHtmlMail("ityouknow@126.com","这是一封HTML邮件",content);
}
```

邮件内容写了一段话，下面为接收到的效果：

由此发现发送 HTML 邮件，是需要拼接一段 HTML 的 String 字符串交给 MimeMessageHelper 来处理，最后由邮件客户端负责渲染显示内容。

发送带附件的邮件

在 MailService 添加 sendAttachmentsMail 方法，发送带附件的邮件主要是使用 FileSystemResource 对文件进行封装，再添加到 MimeMessageHelper 中。

```
public void sendAttachmentsMail(String to, String subject, String content, String
filePath){
    MimeMessage message = mailSender.createMimeMessage();

    try {
        MimeMessageHelper helper = new MimeMessageHelper(message, true);
        helper.setFrom(from);
        helper.setTo(to);
        helper.setSubject(subject);
        helper.setText(content, true);

        FileSystemResource file = new FileSystemResource(new File(filePath));
        String fileName = file.getFilename();
        helper.addAttachment(fileName, file);
        //helper.addAttachment("test"+fileName, file);

        mailSender.send(message);
        logger.info("带附件的邮件已经发送。");
    } catch (MessagingException e) {
        logger.error("发送带附件的邮件时发生异常！", e);
    }
}
```

> 添加多个附件可以使用多条 helper.addAttachment(fileName, file)。

在测试类中添加测试方法：

```
@Test
public void sendAttachmentsMail() {
    String filePath="e:\\temp\\fastdfs-client-java-5.0.0.jar";
    mailService.sendAttachmentsMail("ityouknow@126.com", "主题：带附件的邮件", "有附件
, 请查收！", filePath);
}
```

> 附件可以是图片、压缩包、Word 等任何文件，但是邮件厂商一般都会对附件大小有限制，太大的附件建议使用网盘上传后，在邮件中给出链接。

效果图如下：

发送带静态资源的邮件

邮件中的静态资源一般指图片，在 MailService 中添加 sendInlineResourceMail 方法：

```
public void sendInlineResourceMail(String to, String subject, String content, Stri
ng rscPath, String rscId){
    MimeMessage message = mailSender.createMimeMessage();

    try {
        MimeMessageHelper helper = new MimeMessageHelper(message, true);
        helper.setFrom(from);
        helper.setTo(to);
        helper.setSubject(subject);
        helper.setText(content, true);

        FileSystemResource res = new FileSystemResource(new File(rscPath));
        helper.addInline(rscId, res);

        mailSender.send(message);
        logger.info("嵌入静态资源的邮件已经发送。");
    } catch (MessagingException e) {
        logger.error("发送嵌入静态资源的邮件时发生异常！", e);
    }
}
```

在测试类中添加测试方法：

```
@Test
public void sendInlineResourceMail() {
    String rscId = "neo006";
    String content="<html><body>这是有图片的邮件：<img src=\'cid:" + rscId + "\' ></b
ody></html>";
    String imgPath = "e:\\temp\\weixin.jpg";

    mailService.sendInlineResourceMail("ityouknow@126.com", "主题：这是有图片的邮件",
content, imgPath, rscId);
}
```

> 添加多个图片可以使用多条 <img src='cid:"+ rscId +"' > 和 helper.addInline(rscId, res) 来实现。

效果图如下：

以上是邮件发送的基础服务，已演示支持各种类型邮件。

邮件系统

如果只是想在系统中做一个邮件工具类的话，以上的内容基本就可以满足要求了。若要做成一个邮件系统的话还需要考虑以下几方面：

- 对外提供发送邮件的服务接口
- 固定格式邮件是否考虑使用模板
- 发送邮件时出现网络错误，是否考虑适当的重试机制
- 邮件系统是否考虑异步化，提升服务响应时间
- 是否开发邮件后台管理系统、开发出对应的管理软件、通过页面发送邮件、统计发送邮件成功率等数据。
- 常见异常处理措施

对外提供接口

作为一个独立的邮件系统，需要对外提供接口调用，我们以简单文本邮件为例做个演示。

首先需要定义个实例返回对象：

```
public class MailResult {
    private String rspCode;
    private String rspMsg;

    public MailResult() {
        this.rspCode = "00";
        this.rspMsg = "发送成功";
    }
    //省略 setter/getter
}
```

默认成功的返回码为：00，返回消息为：发送成功。

创建一个 MailController 类对外提供 HTTP 请求接口。

```
@RestController
public class MailController {
    private final Logger logger = LoggerFactory.getLogger(this.getClass());
    @Resource
    private MailService mailService;

    @RequestMapping("/sendSimpleMail")
    public MailResult sendSimpleMail(String to, String subject, String content) {
        MailResult result=new MailResult();
        if(StringUtils.isEmpty(to) || !to.contains("@")){
            result.setRspCode("01");
            result.setRspCode("手机人邮件格式不正确");
        }
        if(StringUtils.isEmpty(content) ){
            result.setRspCode("03");
            result.setRspCode("邮件正文不能为空");
        }
       try {
            mailService.sendSimpleMail(to,subject,content);
            logger.info("简单邮件已经发送。");
        } catch (Exception e) {
            result.setRspCode("04");
            result.setRspCode("邮件发送出现异常");
            logger.error("sendSimpleMail Exception ", e);
        }
        return result;
    }
}
```

当外部请求过来时首先进行参数校验，如果参数有误返回请求；发送邮件出现异常时返回错误，正常情况下返回 00；注意在 Service 层如果对异常信息进行了捕获的话，需要将异常信息抛到上层。

```
try {
    mailSender.send(message);
    logger.info("简单邮件已经发送。");
} catch (Exception e) {
    logger.error("发送简单邮件时发生异常！", e);
    throw e;
}
```

类似上述代码。

按照这个思路也可以提供发送带图片、带附件的邮件，同时也可以封装发送多人邮件、群发邮件等复杂情况。

邮件模板

通常我们使用邮件发送服务的时候，都会有一些固定的场景，如重置密码、注册确认等，给每个用户发送的内容可能只有小部分是变化的。因此，很多时候我们会使用模板引擎来为各类邮件设置成模板，这样只需要在发送时去替换变化部分的参数即可。

我们会经常收到这样的邮件：

```
尊敬的 neo 用户：

        恭喜您注册成为 xxx 网的用户，同时感谢您对 xxx 的关注与支持并欢迎您使用 xxx 的产品与服务。
        ……
```

邮件正文只有 neo 这个用户名在变化，邮件其他内容均不变，如果每次发送邮件都需拼接 HTML 代码，程序不够优雅，并且每次邮件正文有变化都需修改代码非常不方便。因此对于这类邮件，建议做成邮件模板来处理，模板的本质很简单，就是在模板中替换变化的参数，转换为 HTML 字符串即可，这里以 Thymeleaf 为例来演示。

Thymeleaf 是 Spring 官方推荐的前端模板引擎，类似 Velocity、FreeMarker 等模板引擎，相较与其他的模板引擎，Thymeleaf 具有开箱即用的特性。它提供标准和 Spring 标准两种方言，可以直接套用模板实现 JSTL、OGNL 表达式效果，避免每天套模板、改 JSTL、改标签的困扰。Thymeleaf 在有没有网络的环境下皆可运行，即它可以让美工在浏览器中查看页面的静态效果，也可以让程序员在服务器中查看带数据的动态页面效果。

下面来演示使用 Thymeleaf 制作邮件模板。

（1）添加依赖包

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

（2）在 resorces/templates 下创建 emailTemplate.html

emailTemplate.html 文件内容即为邮件的正文内容模板。

```
<!DOCTYPE html>
<html lang="zh" xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8"/>
        <title>邮件模板</title>
    </head>
    <body>
        您好，感谢您的注册，这是一封验证邮件，请点击下面的链接完成注册，感谢您的支持<br/>
        <a href="#" th:href="@{http://www.ityouknow.com/register/{id}(id=${id}) }"
>激活账号</a>
    </body>
</html>
```

我们发现上述的模板中只有 id 是一个动态的值，在发送过程中会根据传入的 id 值来替换链接中的 {id}。

（3）解析模板并发送

```
@Test
public void sendTemplateMail() {
    //创建邮件正文
    Context context = new Context();
    //设置模板需要替换的参数
    context.setVariable("id", "006");
    //使用 templateEngine 替换掉动态参数生产出最后的 HTML 内容
    String emailContent = templateEngine.process("emailTemplate", context);
    //最后调用 sendHtmlMail 发送邮件
    mailService.sendHtmlMail("ityouknow@126.com","主题：这是模板邮件",emailContent);
}
```

我们发现最后调用的还是 sendHtmlMail 的方法，邮件模板的作用只是处理 HTML 生成的部分，通过 Thymeleaf 模板引擎解析固定的模板，再根据参数来动态替换其中的变量，最后通过前面的 HTML 发送的方法发送邮件。

效果图如下：

单击“激活账号”跳转的链接：http://www.ityouknow.com/register/006。

发送失败

因为各种原因，总会有邮件发送失败的情况，如邮件发送过于频繁、网络异常等。在出现这种情况的时候，我们一般会考虑重新重试发送邮件，会分为以下几个步骤来实现：

- 接收到发送邮件请求，首先记录请求并且入库；
- 调用邮件发送接口发送邮件，并且将发送结果记录入库；
- 启动定时系统扫描时间段内，未发送成功并且重试次数小于 3 次的邮件，进行再次发送；
- 重新发送邮件的时间，建议以 2 的次方间隔时间，如 2、4、8、16...。

下面是一些常见的错误返回码。

- 421 HL:ICC 该 IP 同时并发连接数过大，超过了网易的限制，被临时禁止连接。
- 451 Requested mail action not taken: too much fail authentication 登录失败次数过多，被临时禁止登录，请检查密码与帐号验证设置。
- 553 authentication is required，密码配置不正确。
- 554 DT:SPM 发送的邮件内容包含了未被许可的信息，或被系统识别为垃圾邮件，请检查是否有用户发送病毒或者垃圾邮件。
- 550 Invalid User 请求的用户不存在。
- 554 MI:STC 发件人当天内累计邮件数量超过限制，当天不再接收该发件人的投信。

如果使用一个邮箱频繁发送相同内容邮件，也会被认定为垃圾邮件，报 554 DT:SPM 错误。

如果使用网易邮箱可以查看这里的提示：企业退信的常见问题？ 。

其他

异步发送

很多时候邮件发送并不是主业务必须关注的结果，比如通知类、提醒类的业务可以允许延时或者失败，这个时候可以采用异步的方式来发送邮件，加快主交易执行速度。在实际项目中可以采用消息中间件 MQ 发送邮

件，具体做法是创建一个邮件发送的消息队列，在业务中有需要用到邮件发送功能时，给对应消息队列按照规定参数发送一条消息，邮件系统监听此队列，当有消息过来时，处理邮件发送的逻辑。

管理后台

考虑做一个完善的邮件系统，可以设计一个独立的邮件管理后台，不但可以让系统之间调用时使用，也可以提供图形化界面让公司的运营、市场部的同事来发送邮件、查询邮件的发送进度、统计邮件发送成功率；也可以设置一些代码钩子，统计用户点击固定链接次数，方便公司营销人员监控邮件营销转化率。

一个非常完善的邮件系统需要考虑的因素非常多，比如是否设置白名单、黑名单来做邮件接收人的过滤机制，是否给用户提供邮件退订的接口等。因此，在初期邮件发送的基本功能完成之后，再结合公司业务，快速迭代逐步完善邮件系统，是一个推荐的做法。

总结

使用 Spring Boot 集成发送邮件的功能非常简单，只需要简单编码就可以实现发送普通文本邮件、带附件邮件、HTML 格式邮件、带图片邮件等。如果需要做成一个邮件系统还需要考虑很多因素，如邮箱发送失败重试机制、防止邮件被识别为垃圾邮件、固定时间内发送邮件的限制等。在微服务架构中，常常将一些基础功能下沉下来，作为独立的服务来使用，邮件系统作为平台的基础功能，特别适合作为独立的微服务来支持整个系统。

> 点击这里下载源码。