

[18] 메일발송

1. 의존추가(pom.xml)

```
<!-- https://mvnrepository.com/artifact/javax.mail/mail -->
    <dependency>
        <groupId>javax.mail</groupId>
        <artifactId>mail</artifactId>
        <version>1.4.7</version>
    </dependency>
```

2. 한글 필터

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

```
<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

Property 생성



Gamil host , post

```
mail.host=smtp.gmail.com  
mail.port=587
```

- 포트 확인

```
mail.host=smtp.gmail.com  
mail.port=587  
mail.username=aa01096030024  
mail.password=pkwhaojwwidjybd
```

기본설정 라벨 받은편지함 계정 및 가져오기 필터 및 차단된 주소 전달 및 POP/IMAP 부가기능 채팅 및 Meet 고

전달:
[자세히 알아보기](#)

[전달 주소 추가](#)

도움말: [필터를 만들면](#) 메일 중 일부만 전달할 수도 있습니다.

POP 다운로드:
[자세히 알아보기](#)

1. 상태: 9:57 이후로 수신되는 메일에는 **POP**를 사용합니다.
 - ☐ 이미 다운로드 된 메일을 포함하여 모든 메일에 POP를 활성화 하기
 - ☐ 지금부터 수신되는 메일에만 POP를 사용하기
 - ☐ POP 사용 안함

2. POP로 메시지를 여는 경우 [Gmail 사본을 받은편지함에 보관하기](#)

3. 이메일 클라이언트 구성 (예: Outlook, Eudora, Netscape Mail)
[설정 방법](#)

발신 메일(SMTP) 서버

smtp.gmail.com

SSL 필요: 예

TLS 필요: 예(사용 가능한 경우)

인증 필요: 예

TLS/STARTTLS용 포트: 587

직장이나 학교 계정으로 Gmail을 사용하는 경우 올바른 SMTP 구성에 관한 정보는 [관리자](#)에게 문의하세요.

Root-context

```
private static final Logger logger = LoggerFactory.getLogger(
JavaMailSenderImpl
/**
 * Simply selects t
JavaMailSenderImpl - org.springframework.mail.javamail
JavaMailSenderImpl() : void - Method stub
```

```
<!-- mailSender 빈 생성 -->
<beans:bean id="mailSender"
class="org.springframework.mail.javamail.JavaMailSenderImpl">
```

- mailSender 빈 생성

```
28 <!-- mailSender 빈 생성 -->
29 <beans:bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl"
30 <beans:property name="host" value="${mail.host}"/>
31 <beans:property name="port" value="${mail.port}"/>
32 <beans:property name="username" value="${mail.username}"/>
33 <beans:property name="password" value="${mail.password}"/>
34 </beans:bean>
35 </beans:beans>
36
```

mail.properties

```
1 mail.host=smtp.gmail.com
2 mail.port=587
3 mail.username=aa01096030024
4 mail.password=pkwhaojwwidjybdv
```

```

<beans:property name="javaMailProperties">
<beans:props>
<beans:prop key="mail.smtp.auth">true</beans:prop>
<beans:prop key="mail.smtp.starttls.enable">true</beans:prop>
<beans:prop key="mail.debug">true</beans:prop>
<beans:prop key="mail.transport.protocol">smtp</beans:prop>
<beans:prop key="mail.smtp.ssl.trust">smtp.gmail.com</beans:prop>
<beans:prop key="mail.smtp.ssl.protocols">TLSv1.2</beans:prop>
</beans:props>
</beans:property>

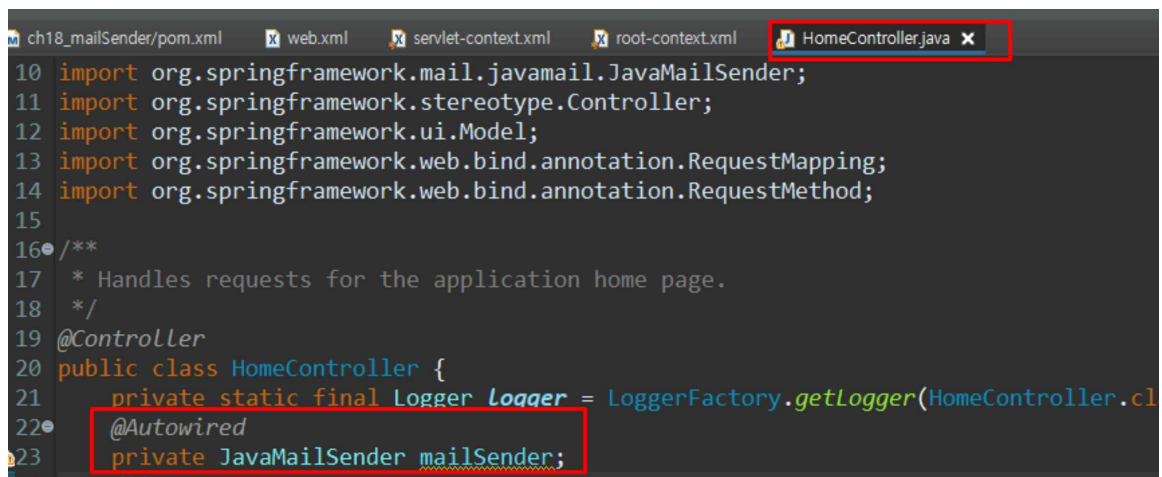
```

```

<beans:property name="javaMailProperties">
  <beans:props>
    <beans:prop key="mail.smtp.auth">true</beans:prop>
    <beans:prop key="mail.smtp.starttls.enable">true</beans:prop>
    <beans:prop key="mail.debug">true</beans:prop>
    <beans:prop key="mail.transport.protocol">smtp</beans:prop>
    <beans:prop key="mail.smtp.ssl.trust">smtp.gmail.com</beans:prop>
    <beans:prop key="mail.smtp.ssl.protocols">TLSv1.2</beans:prop>
  </beans:props>
</beans:property>

```

-컨트롤러 @Autowired 추가 후 실행 해보기

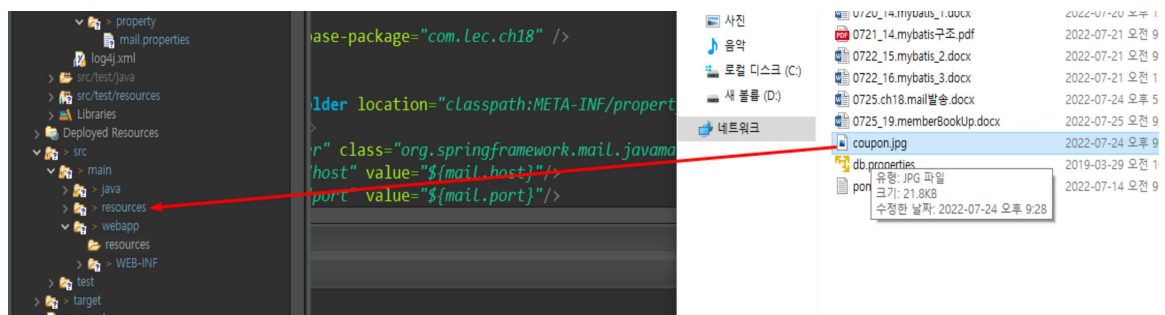


```

10 import org.springframework.mail.javamail.JavaMailSender;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.ui.Model;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15
16 /**
17  * Handles requests for the application home page.
18  */
19 @Controller
20 public class HomeController {
21     private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
22     @Autowired
23     private JavaMailSender mailSender;

```

-img 추가



-컨트롤러

```
*/
@Controller
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    @Autowired
    private JavaMailSender mailSender;
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Model model) {
        return "join";
    }
}
```

-join.jsp 생성

```
<body>
    <form action="textMail">
        <p>이름<input type="text" name="name" required="required"></p>
        <p>메일<input type="email" name="email" required="required"></p>
        <input type="submit" value="회원가입(TEXT 단순 메일 감)">
    </form>
    <hr color="green">
    <form action="htmlMail">
        <p>이름<input type="text" name="name" required="required"></p>
        <p>메일<input type="email" name="email" required="required"></p>
        <input type="submit" value="회원가입(html 메일 감)">
    </form>
</body>
```

- (textMail)처리 다시 컨트롤러 에서
- jsp 에서 입력 받은 name , email 받아오고

```
@RequestMapping(value = "textMail", method= RequestMethod.GET)
public String textMail(String name, String email)
```

join.jsp

```
</script>
<head>
<body>
    <form action="textMail">
        <p>이름<input type="text" name="name" required="required"></p>
        <p>메일<input type="email" name="email" required="required"></p>
```

```

@RequestMapping(value = "textMail", method= RequestMethod.GET)
public String textMail(String name, String email, Model model) {
    SimpleMailMessage message = new SimpleMailMessage(); // textMail에서 필요한 아이
    message.setFrom("aa01096030024@gmail.com"); // 관리자메일 보내는 사람메일
    message.setTo(email); //받는 사람 메일
    message.setSubject("[TEXT 가입인사]" + name + "님 회원가입 감사합니다"); // 보낼 메일에 제목
    String content = name + "님 회원가입 감사합니다\n <h1>태그 안먹음</h1>"; // 글내용 기니까 String으로 먼저 받아주고
    message.setText(content); // 메일 본문 내용

    mailSender.send(message); // 메일 보내기
    model.addAttribute("mailSendResult", "TEXT메일이 발송되었습니다");
    return "sendResult";
}

```

SimpleMailMessage : 바로 호출 가능 textMail에서 필요한 아이

setFrom : 관리자메일 (보내는사람 메일)

setTo(email) : jsp에서 입력한 받는사람 메일

setSubject : 보낼 메일에 제목

String content : 글내용은 길 수도있으니 String 으로 먼저 받아온다

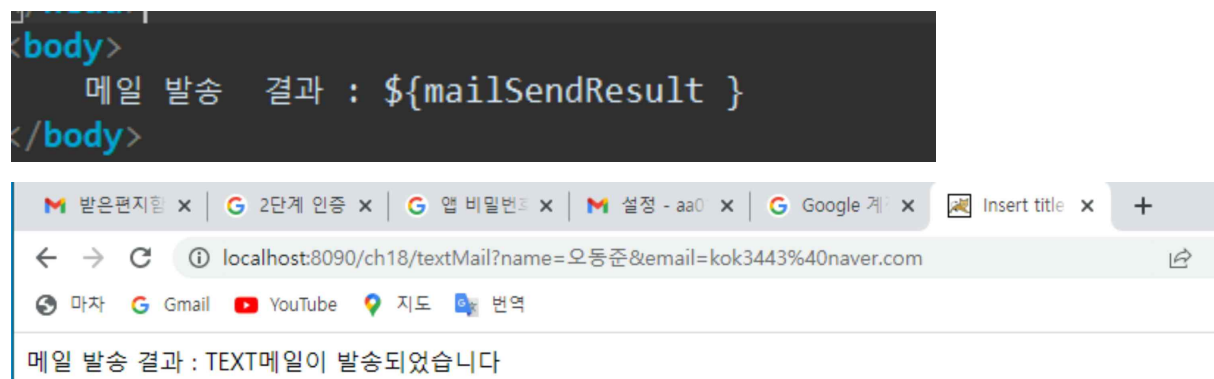
setText(content) : String 으로 받아온 content 넣어주기

mailSender(JavaMailSender변수).send : 메일보내기 처리

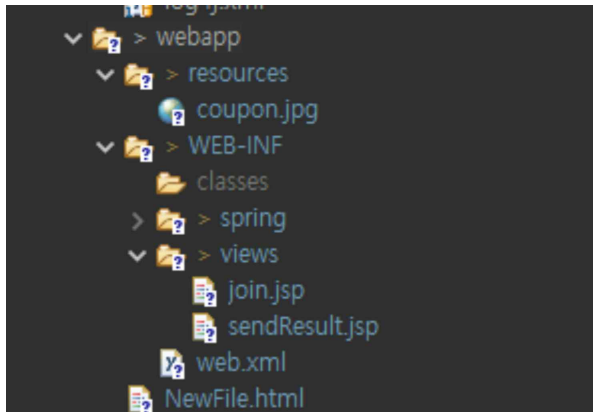
<h1>태그 안먹음</h1>

-sendResult.jsp

결과 확인

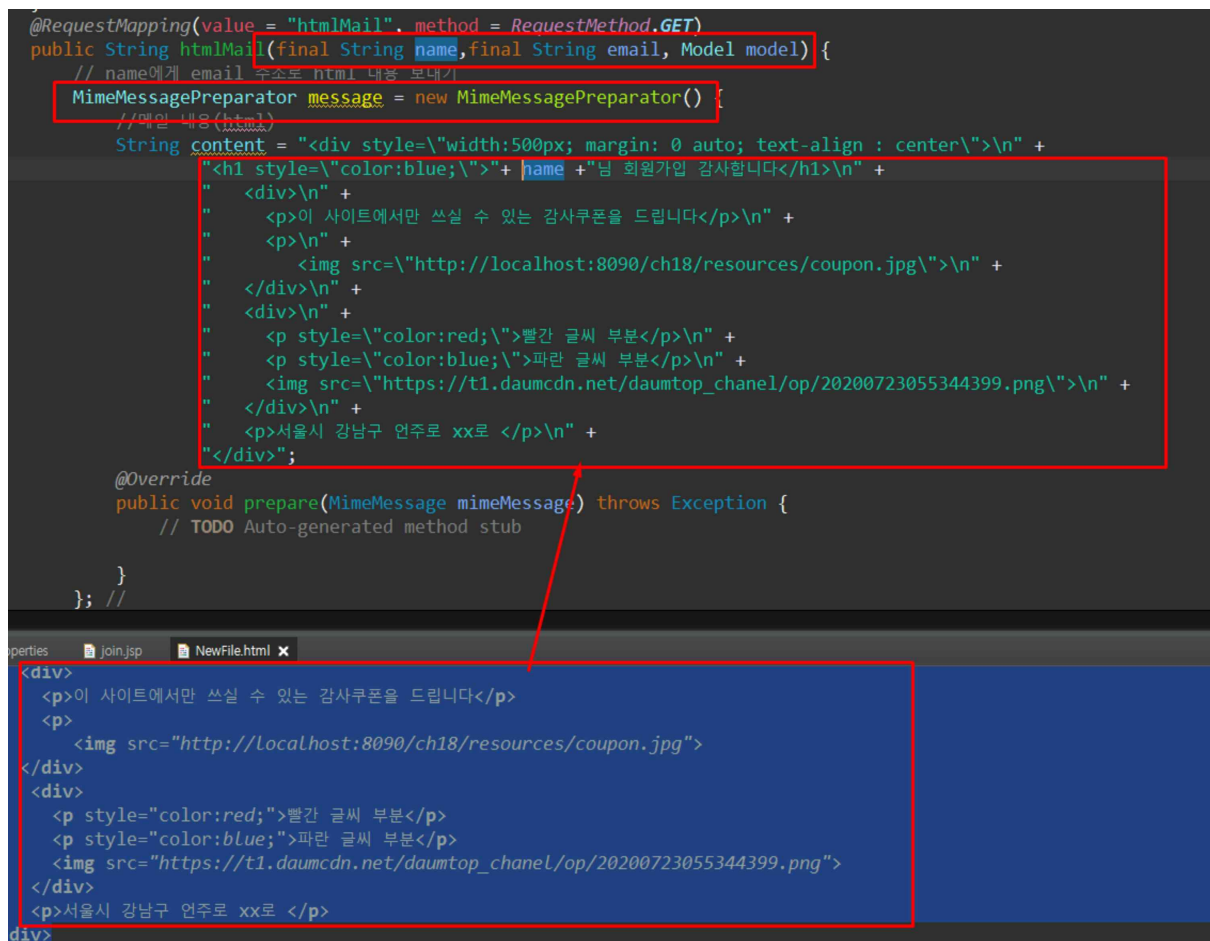


-htmlMail 처리



-webapp 밑에 html생성

-html(본문내용) 컨트롤러에서 받아오기



```

        <p>저를자랑함주인주보xx호</p>\n    +
        "</div>";
    @Override
    public void prepare(MimeMessage mimeMessage) throws Exception {
        // 보내는 메일, 받는 메일, 메일 제목, utf-8
        mimeMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(email));
        mimeMessage.setFrom(new InternetAddress("aa01096030024@gmail.com"));
        mimeMessage.setSubject("[HTML가입인사]" + name + "님 회원가입 감사합니다");
        mimeMessage.setText(content, "utf-8", "html");
    }
}; //
mailSender.send(message);
model.addAttribute("mailSendResult", "HTML 메일이 발송되었습니다");
return "sendResult";

```