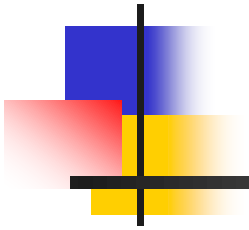


spring 10강-이메일 발송

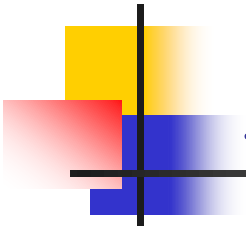
양 명 숙
[now4ever7@gmail.com]

스프링에서 이메일 발송기능 구현하기

- jar 파일 추가
  - [1]org.springframework.mail.javamail.JavaMailSenderImpl 사용하기 위해서는
    - spring-context-support-3.1.0.RELEASE.jar 추가
  - [2] javax.mail.internet.MimeMessage 등을 사용하기 위해서는
    - JavaMail API : mail-1.4.7.jar 추가
    - http://www.oracle.com/technetwork/java/index-138643.html
    - http://www.mvnrepository.com/artifact/javax.mail/mail/1.4.7

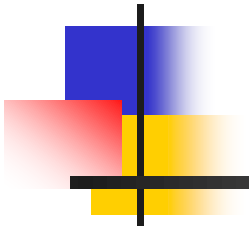
```
<dependency>
<groupId>javax.mail</groupId>
<artifactId>mail</artifactId>
<version>1.4.7</version>
</dependency>
```

pom.xml 에 추가

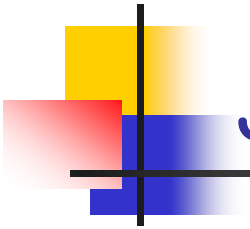


스프링에서 이메일 발송기능 구현하기

- iis와 같은 윈도우기반 웹서버는 이메일 발송기능을 제공하지만, Tomcat에서는 지원하지 않아, 별도의 이메일서버가 필요함
- 하지만 gmail을 사용하면 이메일서버 없이도 구현가능
- 단, gmail의 계정이 필요함



James 를 이용한 메일 발송

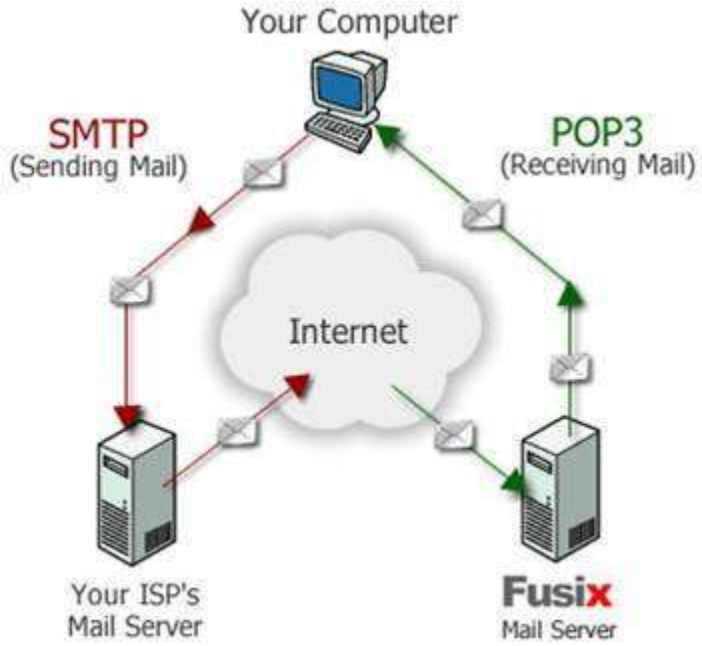


Java Mail API

- JAMES를 이용해 이메일 전송하기
  - JAVA MAIL API – 이메일을 전송하기위해 사용되는 API
  - E-Mail 서버로 사용될 수 있는 JAMES를 기반으로 이메일 전송하기
- JAMES
  - Apache에서 제공하는 Java Apache Mail Enterprise Server

JAMES

- 클라이언트에서 자바로 이메일을 작성하여 전송하게 되면 SMTP프로토콜을 이용하여 전달되어지는데 이 SMTP프로토콜을 해석 할 수 있는 중간 서버가 필요함
- 중간서버로 JAMES를 사용할 수 있고, Google 의 SMTP서버나 포탈 사이트에서 제공하는 메일서버를 사용할 수 있음



james

JAMES 설치 및 환경구축

- [1] 메일 서버 다운로드
  - 자바의 메일서버는 아파치에서 제공하는 james(Java Apache Mail Enterprise Server) 서버를 이용할 수 있다
  - 서버 다운로드 http://james.apache.org
    - apache-james-2.3.2.zip
- [2] 메일 서버 설치
  - 다운로드 받은 파일의 압축을 풀면 됨
  - 바탕화면에 압축을 풀면 안됨(한글 경로가 아닌 곳에 압축을 풀 것)
- [3] 메일 서버 환경 설정
  - james 서버의 환경설정 - config.xml 파일에서.
    - d:₩james-2.3.2₩apps 들어가서 james.sar 압축풀기
    - d:₩james-2.3.2₩apps₩james₩SAR-INF 들어가서 config.xml 파일 수정하기

config.xml

- config.xml 다음과 같이 수정

- 1) 사용자의 암호 지정 (설정이 끝나면 이 암호로 서버에 접속할 수 있다.)
  <account login="root" password="암호" />

- 2) 메일서버 지정하기
  servername 태그안에 자신의 아이피 혹은 호스트명을 적는다.
  <servername>127.0.0.1</servername>

- 3) DNS서버 지정하기
  ipconfig/all 로 자신의 DNS서버를 찾아서 기재한다.
  <dnsserver>
  <servers>
  <server>DNS서버 주소</server>
  </servers>
  .... 중략
  </dnsserver>

JAMES 설치 및 환경구축

- [4] JAMES 실행
  - /bin/ 디렉토리의 run.bat 파일을 실행시켜서 인스턴스를 띄운다.
  - 실행결과는 아래와 같다.
  - 창을 닫지말고, 켜 둔다.

```
Using PHOENIX_HOME:   C:□james-2.3.2
Using PHOENIX_TMPDIR: C:□james-2.3.2□temp
Using JAVA_HOME:
Phoenix 4.2

James Mail Server 2.3.2
Remote Manager Service started plain:4555
POP3 Service started plain:110
SMTP Service started plain:25
NNTP Service started plain:119
FetchMail Disabled
```

james 서버 접속 및 관리

- [1] 커맨드창에서 텔넷으로 서버에 접속.

  C:₩>telnet localhost 4555

- [2] root계정으로 로그인.
  - 아이디 : root
  - 패스워드 : config.xml 에서 지정한 비밀번호

- [3] help 라고 입력해보면 james 서버를 제어할 명령어가 출력됨

context-common.xml 에 추가

- spring설정파일(context-common.xml)에 추가하기

gmail 메일서버 이용시

```
<bean id="mailSender"
        class ="org.springframework.mail.javamail.JavaMailSenderImpl" >
        <property name="host" value="smtp.gmail.com" />
        <property name="port" value="587" />
        <property name="username" value="아이디@gmail.com" />
        <property name="password" value="비밀번호" />
        <property name="javaMailProperties">
                <props>
                        <prop key="mail.smtp.starttls.enable">true</prop>
                        <prop key="mail.smtp.auth">true</prop>
                        <prop key="mail.transport.protocol">smtp</prop>
                        <prop key="mail.debug">true</prop>
                </props>
        </property>
</bean>
```

mw-002.cafe24.com

아이디@cafe24.com

https://www.google.com/settings/u/0/security/lesssecureapps

보안 수준이 낮은 앱의 액세스 => 사용으로 변경

context-common.xml 에 추가

- spring설정파일(context-common.xml)에 추가하기 **james 메일서버 이용시**

```
<bean id="mailSender"
        class ="org.springframework.mail.javamail.JavaMailSenderImpl" >
        <property name="host" value="localhost" />
        <property name="port" value="25" />
        <property name="username" value="root" />
        <property name="password" value="root" />
        <property name="javaMailProperties">
                <props>
                        <prop key="mail.smtp.starttls.enable">true</prop>
                        <prop key="mail.smtp.auth">true</prop>
                        <prop key="mail.transport.protocol">smtp</prop>
                        <prop key="mail.debug">true</prop>
                </props>
        </property>
</bean>
```

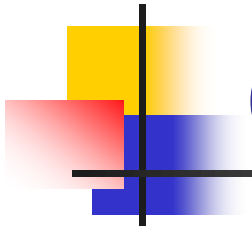
스프링 부트 – 이메일

- 1. email 전송을 지원하는 스프링 모듈을 import

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

- 2. 스프링의 mail starter는 MailSender interface와 MailSenderImpl을 제공한다.
  - 1. 실제로 사용하는 인터페이스와 클래스는 편의기능을 더 추가한 JavaMailSender, JavaMailSenderImpl이 된다.
  - 2. 어차피 @Autowired로 주입하기 때문에 구현 클래스에 대해서 알 필요가 없다.

```
@Component
public class EmailSender {
                @Autowired
                private JavaMailSender mailSender;
```



이메일

- 3. application.properties에 아래 내용 넣기

```
#email
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=이메일 주소
spring.mail.password=비밀번호
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

EmailSender

```
package com.herb.app.email;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMessage.RecipientType;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Component;
@Component
public class EmailSender  {
    @Autowired  protected JavaMailSender  mailSender;
    public void SendEmail(String subject, String content,
                          String receiver, String sender) throws Exception {
        MimeMessage msg = mailSender.createMimeMessage();
        msg.setSubject(subject);
        msg.setText(content);
        msg.setRecipient(RecipientType.TO , new InternetAddress(receiver));
        msg.setFrom(new InternetAddress(sender));
        mailSender.send(msg);
    }
}
```

EmailController

```
package com.herb.app.email.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import com.herb.app.email.EmailSender;

@Controller
@RequestMapping("/email")
public class EmailController {
    @Autowired
    private EmailSender emailSender;

    private static Logger logger = LoggerFactory.getLogger(EmailController.class);
```

```
<a href="<c:url value='/email/send.do' />">이메일 보내기</a>
```

EmailController

```
@RequestMapping("/send.do")
public String sendEmail (){
        String receiver = "now4ever7@gmail.com"; //받는 사람의 이메일 주소
        String subject = "문의에 대한 답변입니다!!. 안녕하세요~~";
        String content = "이메일 내용입니다. 감사합니다";
        String sender = "admin@herbmall.com"; //보내는 사람의 이메일 주소

        try{
                emailSender.SendEmail(subject, content, receiver, sender);
                logger.debug("메일발송 성공!");
        }catch(Exception e){
                e.printStackTrace();
                logger.debug("메일 발송 실패 : e.message=" + e.getMessage());
        }

        return "redirect:/index.do";
    }
}//class
```

```
DEBUG: JavaMail version 1.4.5
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name:
    {com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,
    Sun Microsystems, Inc],
    com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun
    Microsystems, Inc],
    com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun
    Microsystems, Inc],
    com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun
    Microsystems, Inc], com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun
    Microsystems, Inc],
    com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems, Inc]}
DEBUG: Providers Listed By Protocol: {imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun
    Microsystems, Inc], imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun Microsystems, Inc],
    smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems, Inc],
    pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems, Inc],
    pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems, Inc],
    smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems, Inc]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
DEBUG: getProvider() returning javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems,
    Inc]
DEBUG SMTP: useEhlo true, useAuth true
DEBUG SMTP: trying to connect to host "localhost", port 25, isSSL false
220 yang-hp SMTP Server (JAMES SMTP Server 2.3.2) ready Tue, 9 Jun 2015 21:51:11 +0900 (KST)
DEBUG SMTP: connected to host "localhost", port: 25

EHLO 125.133.193.3
250-yang-hp Hello 125.133.193.3 (127.0.0.1 [127.0.0.1])
250-PIPELINING
250 ENHANCEDSTATUSCODES
DEBUG SMTP: Found extension "PIPELINING", arg ""
```

DEBUG SMTP: Found extension "ENHANCEDSTATUSCODES", arg ""
DEBUG SMTP: use8bit false
MAIL FROM:<admin@herbmall.com>
250 2.1.0 Sender <admin@herbmall.com> OK
RCPT TO:<now4ever7@gmail.com>
250 2.1.5 Recipient <now4ever7@gmail.com> OK
DEBUG SMTP: Verified Addresses
DEBUG SMTP: now4ever7@gmail.com
DATA
354 Ok Send data ending with <CRLF>.<CRLF>
Date: Tue, 9 Jun 2015 21:51:11 +0900 (KST)
From: admin@herbmall.com
To: now4ever7@gmail.com
Message-ID: <93536588.0.1433854271184.JavaMail.yang-hp1@yang-hp>
Subject: =?UTF-8?B?66y47J2Y7JeQIOuMgO2VnCDri7Xrs4DsnoXri4g=?=
 =?UTF-8?B?64uk5SEuIOyViOuFle2VmOyEuOyalD8=?=
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: base64

7J2066mU7J28IOuCtOyaqeyeheuLiOuLpC4g6rCQ7IKs7ZWp64ul64uk
.
250 2.6.0 Message received
QUIT
221 2.0.0 yang-hp Service closing transmission channel

gmail 메일서버 이용

```
DEBUG: JavaMail version 1.4.7
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name:
    {com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Oracle],
    com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Oracle],
    com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Oracle],
    com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Oracle],
    com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Oracle],
    com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Oracle]}
DEBUG: Providers Listed By Protocol:
    {imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Oracle],
    imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Oracle],
    smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Oracle],
    pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Oracle],
    pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Oracle],
    smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Oracle]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
DEBUG: getProvider() returning
    javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Oracle]
DEBUG SMTP: useEhlo true, useAuth true
DEBUG SMTP: trying to connect to host "smtp.gmail.com", port 587, isSSL false
220 mx.google.com ESMTP kl10sm550529pbd.20 - gsmtp
DEBUG SMTP: connected to host "smtp.gmail.com", port: 587
```

EHLO 118.37.179.53
250-mx.google.com at your service, [118.37.179.53]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250 CHUNKING
DEBUG SMTP: Found extension "SIZE", arg "35882577"
DEBUG SMTP: Found extension "8BITMIME", arg ""
DEBUG SMTP: Found extension "STARTTLS", arg ""
DEBUG SMTP: Found extension "ENHANCEDSTATUSCODES", arg ""
DEBUG SMTP: Found extension "CHUNKING", arg ""
STARTTLS
220 2.0.0 Ready to start TLS
EHLO 118.37.179.53
250-mx.google.com at your service, [118.37.179.53]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2 PLAIN-CLIENTTOKEN
250-ENHANCEDSTATUSCODES
250 CHUNKING
DEBUG SMTP: Found extension "SIZE", arg "35882577"
DEBUG SMTP: Found extension "8BITMIME", arg ""
DEBUG SMTP: Found extension "AUTH", arg "LOGIN PLAIN XOAUTH XOAUTH2 PLAIN-CLIENTTOKEN"
DEBUG SMTP: Found extension "ENHANCEDSTATUSCODES", arg ""
DEBUG SMTP: Found extension "CHUNKING", arg “”

```
DEBUG SMTP: Attempt to authenticate using mechanisms: LOGIN PLAIN DIGEST-MD5 NTLM
DEBUG SMTP: AUTH LOGIN command trace suppressed
DEBUG SMTP: AUTH LOGIN succeeded
DEBUG SMTP: use8bit false
MAIL FROM:<admin@sunlight.pe.kr>
250 2.1.0 OK kl10sm550529pbd.20 - gsmtp
RCPT TO:<now4ever7@cyworld.com>
250 2.1.5 OK kl10sm550529pbd.20 - gsmtp
DEBUG SMTP: Verified Addresses
DEBUG SMTP:   now4ever7@cyworld.com
DATA
354  Go ahead kl10sm550529pbd.20 - gsmtp
Date: Tue, 24 Jun 2014 22:41:48 +0900 (KST)
From: admin@sunlight.pe.kr
To: now4ever7@cyworld.com
Message-ID: <51989910.0.1403617308154.JavaMail.yang@yang-hp>
Subject: =?MS949?B?ua7Ax7+hILTrx9EgtOS6r8DUtM+02SEhLiC+yLPnx8+8vL/kfn4=?=
MIME-Version: 1.0
Content-Type: text/plain; charset=MS949
Content-Transfer-Encoding: base64

wMy43sDPILO7v+vA1LTPtNkulLCou+fH1bTPtNk=
.
250 2.0.0 OK 1403617311 kl10sm550529pbd.20 - gsmtp
QUIT
221 2.0.0 closing connection kl10sm550529pbd.20 - gsmtp
```

전자 메일 발송 예제

받는이

보내는 이

제목

내용

전자 메일 발송

이메일 내용에 태그 적용하기

• 이메일 내용에 태그 적용하기
msg.setText(content, "utf-8", "html");

```
@Component
public class EmailSender {

    @Autowired private JavaMailSender mailSender;

    public void sendMail(String subject, String content,
                         String receiver, String sender) throws MessagingException {
        /*
        MimeMessage mimeMsg=mailSender.createMimeMessage();
        mimeMsg.setSubject(subject);
        mimeMsg.setText(content, “utf-8”, “html”);
        mimeMsg.setRecipient(RecipientType.TO,
                             new InternetAddress(receiver));
        mimeMsg.setFrom(new InternetAddress(sender));

        mailSender.send(mimeMsg);
        */
```

이메일 내용에 태그 적용하기

```
                MimeMessage message = mailSender.createMimeMessage();
                MimeMessageHelper messageHelper
                        = new MimeMessageHelper(message, true, "UTF-8");
        //MimeMessageHelper(MimeMessage mimeMessage, boolean multipart, String encoding)

                messageHelper.setSubject(subject);
                messageHelper.setText(content, true);
                //setText(String text, boolean html)

                messageHelper.setFrom(new InternetAddress(sender));
                messageHelper.setTo(new InternetAddress(receiver));

                mailSender.send(message);
        }

}
```

이메일 내용에 태그 적용하기

```
@Controller
public class EmailController {
        private static final Logger logger =LoggerFactory.getLogger(EmailController.class);

        @Autowired EmailSender emailSender;

        @RequestMapping("/email/send.do")
        public String send_email() {
                String subject="문의에 대한 답변입니다. 안녕하세요";
                String content="<h1>이메일 내용입니다.</h1> <strong>감사합니다.</strong>";
                String receiver="now4ever7@gmail.com";
                String sender="admin@herbmall.com";

                try {
                        emailSender.sendMail(subject, content, receiver, sender);
                        logger.info("이메일 발송 성공");
                } catch (MessagingException e) {
                        logger.info("이메일 발송 실패!!");
                        e.printStackTrace();
                }
                return "redirect:/index.do";
        }
```