+++++++++++++++++++++++++

+Chapter 13 SPRING을 이용한 Mail 발송+

+++++++++++++++++++++++++

1. MailSender와 JavaMailSender를 이용한 Mail 발송

: SPRING Mail 발송 기능을 위한 MailSender Interface를 제공하고 있는데, 이 Interface는 다음과 같이 정의되어 있다.

package org.springframework.mail;

public interface MailSender{

void send(SimpleMailMessage simpleMailMessage) throws MailException;

void send(SimpleMailMessage[] simpleMailMessage) throws MailException;

}

MailSender Interface는 SimpleMailMessage를 전달받아 Mail을 발송하는 기능을 정의하고 있다. SimpleMailMessage는 Mail 제목과 단순 텍스트 내용으로

구성된 Mail을 발송할 때에 사용된다.

MailSender Interface를 상속받은 JavaMailSender는 Java Mail API의 MimeMessage를 이용해서 Mail을 발송하는 기능을 추가적으로 정의하고 있다.

JavaMailSender Interface는 다음과 같이 정의되어 있다.

package org.springframework.mail.javamail;

import java.io.InputStream;

import javax.mail.Internet.MimeMessage;

import org.springframework.mail.MailException;

import org.springframework.mail.MailSender;

public interface JavaMailSender extends MailSender{

MimeMessage createMimeMessage();

MimeMessage createMimeMessage(InputStream contentStream) throws MailException;

void send(MimeMessage mimeMessage) throws MailException;

void send(MimeMessage[] mimeMessage) throws MailException;

void send(MimeMessagePreparator mimeMessagePreparator) throws MailException;

void send(MimeMessagePreparator[] mimeMessagePreparator) throws MailException;

}

MailSender Interface와 JavaMailSender Interface의 메서드들이 발생하는 MailException은 RuntimeException이므로, 예외 처리가 필요한 경우에만

catch 블럭에서 처리해 주면 된다.

SPRING은 JavaMailSender Interface의 구현체로 JavaMailSenderImpl Class를 제공하고 있으므로, 이 Class를 이용해서 빈 설정을 하게 된다.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.1 JavaMailSender 빈 설정

: JavaMailSenderImpl Class는 Java Mail API를 이용해서 Mail을 발송하며 기본적으로 SMTP 프로토콜을 사용한다. SMTP 서버를 이용해서 Mail을 발송하므로

SMTP 서버 주소와 포트 번호를 필요로 한다. 이 두 정보는 각각 host 프로퍼티와 port 프로퍼티를 이용해서 입력 받는다.

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl"

p:host="somesmtp.host"

p:port="25"

p:defaultEncoding="euc-kr" />

<bean id="simpleNotifier" class="madvirus.spring.chap13.SimpleRegistrationNotifier" p:mailSender-ref="mailSender"/>

port 프로퍼티의 기본 값은 25이므로, 포트 번호가 25가 아닌 경우에만 port 속성을 설정하면 된다. defaultEncoding 프로퍼티는 발송될 Mail의 기본 인코딩을

설정한다. JavaMailSenderImpl은 내부적으로 Java Mail API의 MimeMessage를 이용하기 때문에, 인코딩을 지정하지 않는 SimpleMimeMessage를 이용할 경우에는

defaultEncoding 프로퍼티의 속성 값을 알맞게 입력해 주는 것이 좋다.

만약, SMTP 서버에서 인증을 필요로 한다면 다음과 같이 username 프로퍼티와 password 프로퍼티를 이용해서 인증에 사용되는 정보를 입력한다.

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl"

p:host="somesmtp.host" p:port="25" p:username="system"

p:password="system" p:defaultEncoding="euc-kr" />

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.2 SimpleMailMessage를 이용한 Mail 발송

: 단순히 텍스트만 구성된 Mail Message를 생성할 때에는 SimpleMailMessage를 이용한다. SimpleMailMessage는 아래의 메서드를 이용해서 Mail의 내용을

구성할 수 있도록 하고 있다.

메 서 드 설 명

=========================================

setForm(String form) 발신지 설정.

=========================================

setReplyTo(String replyTo) 응답 주소 설정.

=========================================

setTo(String to) 수신지 설정.

=========================================

setTo(String[] to) 수신지 목록 설정.

=========================================

setCc(String cc) 참조자 설정.

=========================================

setCc(String[] cc) 참조자 목록 설정.

=========================================

setBcc(String bcc) 숨은 참조자 설정.

=========================================

setBcc(String[] bcc) 숨은 참조자 목록 설정.

=========================================

setSentDate(Date sentDate) Mail 발송일 설정.

=========================================

setSubject(String subject) Mail 제목(주제) 설정.

=========================================

setText(String text) Mail 내용 설정.

=========================================

아래 Code는 SimpleMailMessage를 이용해서 Mail Message를 생성한 뒤 MailSender를 이용해서 Mail을 발송하는 예제 Code이다.

import org.springframework.mail.MailException;

import org.springframework.mail.MailSender;

import org.springframework.mail.SimpleMailMessage;

public class SimpleRegistrationNotifier implements RegistrationNotifier {

private MailSender mailSender;

public void setMailSender(MailSender mailSender) {

this.mailSender = mailSender;

}

@Override

public void sendEmail(Member member) {

SimpleMailMessage message = new SimpleMailMessage();

message.setSubject("[Simple] 회원 가입 안내");

message.setText("회원 가입을 환영합니다.");

message.setFrom("madvirus@wemade.com");

message.setTo(member.getEmail());

try {

mailSender.send(message);

} catch (MailException ex) {

// 예외 처리

ex.printStackTrace();

}

}

}

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.3 SimpleMailMessage의 재사용.

: SimpleMailMessage Class는 다른 SimpleMailMessage Class로부터 내용을 복사해서 사용하는 기능을 제공하고 있다. 아래 Code와 같이 생성자를 통해

SimpleMailMessage 객체를 전달하면, 설정 정보를 모두 복사한다.

SimpleMailMessage anotherMaiMessage = new SimpleMailMessage();

message.setFrom("madvirus@wemade.com");

...

SimpleMailMessage message = new SimpleMailMessage(anotherMailMessage);

이 기능을 이용하면 SimpleMailMessage를 SPRING 설정 파일에서 설정하고 필요로 하는 빈에서 재사용할 수 있게 된다. 아래 Code는 설정 예를 보여주고 있다.

<bean id="templateMailMessage" class="org.springframework.mail.SimpleMailMessage"

p:from="nals@lycos.co.kr"

p:subject="회원 가입을 환영합니다."/>

<bean id="simpleNotifier" class="madvirus.spring.chap13.SimpleRegistrationNotifier"

p:mailSender-ref="mailSender"

p:templateMailMessage-ref="templateMailMessage"/>

templateMailMessage 빈을 전달받은 Class에서는 templateMailMessage 객체를 이용해서 SimpleMailMessage 객체를 생성하면 추가적으로 필요한 정보만

설정하면 되므로 Mail Message를 생성하는 Code가 좀 더 간단해 진다. 아래 Code는 재사용의 예를 보여 주고 있다.

public class SimpleRegistrationNotifier implements RegistrationNotifier {

...

private SimpleMailMessage templateMailMessage;

public void setTemplateMailMessage(SimpleMailMessage templateMailMessage){

this.templateMailMessage = templateMailMessage;

}

...

@Override

public void send(Member member) {

SimpleMailMessage message = new SimpleMailMessage(templateMailMessage);

message.setText("회원 가입을 환영합니다.");

message.setTo(member.getEmail());

try {

mailSender.send(message);

} catch (MailException ex) {

// 예외 처리

ex.printStackTrace();

}

}

}

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.4 Java Mail API의 MimeMessage를 이용한 Message 생성

: SimpleMailMessage는 단순히 텍스트 기반의 Message를 발송하는 데에는 적합하지만, Mail 내용이 HTML로 구성되어 있다던가, 파일을 첨부해야 하는 경우에는

사용할 수 없다. 이런 기능이 필요할 경우에는 Java Mail API가 제공하는 MimeMessage를 직접 이용해서 Mail을 발송해 주어야 한다.

JavaMailSender Interface는 MimeMessage 객체를 생성해 주는 createMimeMessage() 메서드를 제공하고 있으며, 이 메서드가 리턴한 MimeMessage 객체를

이용해서 메서드를 구성한 뒤 Mail을 발송하면 된다. 아래는 MimeMessage를 이용해서 Message를 구성한 뒤 JavaMailSender를 이용해서 Mail을 발송하는 예제

Code이다.

import javax.mail.MessagingException;

import javax.mail.Message.RecipientType;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import org.springframework.mail.MailException;

import org.springframework.mail.javamail.JavaMailSender;

public class MimeRegistrationNotifier implements RegistrationNotifier {

private JavaMailSender mailSender;

public void setMailSender(JavaMailSender mailSender) {

this.mailSender = mailSender;

}

@Override

public void sendEmail(Member member) {

MimeMessage message = mailSender.createMimeMessage();

try {

message.setSubject("[Mime] 회원 가입 안내", "euc-kr");

String htmlContent = "<strong>안녕하세요</strong>, 반갑습니다.";

message.setText(htmlContent, "euc-kr", "html");

message.setFrom(new InternetAddress("madvirus@wemade.com"));

message.addRecipient(RecipientType.TO, new InternetAddress(member

.getEmail()));

} catch (MessagingException e) {

// 예외 처리

e.printStackTrace();

return;

}

try {

mailSender.send(message);

} catch (MailException e) {

// 예외 처리

e.printStackTrace();

}

}

}

===================================================================================================

2. MimeMessageHelper를 이용한 파일 첨부

: Java Mail API의 MimeMessage를 이용하면 파일을 첨부할 수 있긴 하지만, 작성해야 할 Code가 많기 때문에 꽤 성가신 데, SPRING이 제공하는 MimeMessageHelper

Class를 사용하면 파일 첨부를 매우 간단하게 처리할 수 있다.

MimeMessageHelper Class는 MimeMessage Class를 직접 사용하지 않고 MimeMessageHelper Class가 제공하는 기능을 이용해서 Mail Message를 생성할 수 있도록

돕는 Class이다. MimeMessageHelper Class는 setSubject(), setTo(), setFrom(), setText() 등의 메서드를 제공하고 있어 Mail 내용을 구성할 수 있도록 하며,

addAttachment() 메서드를 사용해서 파일을 첨부할 수 있도록 돕고 있다.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2.1 MimeMessageHelper를 이용한 첨부 파일 추가

: 다음 Code는 MimeMessageHelper Class를 이용하여 파일 첨부된 Mail을 발송하는 방법을 보여주고 있다.

import javax.activation.DataSource;

import javax.activation.FileDataSource;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeUtility;

import org.springframework.mail.MailException;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.MimeMessageHelper;

public class MimeHelperRegistrationNotifier implements RegistrationNotifier {

private JavaMailSender mailSender;

public void setMailSender(JavaMailSender mailSender) {

this.mailSender = mailSender;

}

@Override

public void sendEmail(Member member) {

MimeMessage message = mailSender.createMimeMessage();

try {

MimeMessageHelper messageHelper = new MimeMessageHelper(message,

true, "euc-kr");

messageHelper.setSubject("[Attachment] 회원 가입 안내");

String htmlContent = "<strong>안녕하세요</strong>, 반갑습니다.";

messageHelper.setText(htmlContent, true);

messageHelper.setFrom("madvirus@wemade.com", "최범균");

messageHelper.setTo(new InternetAddress(member.getEmail(), member

.getName(), "euc-kr"));

DataSource dataSource = new FileDataSource("c:\\안내문.docx");

messageHelper.addAttachment(MimeUtility.encodeText("안내문.docx",

"euc-kr", "B"), dataSource);

} catch (Throwable e) {

e.printStackTrace();

return;

}

try {

mailSender.send(message);

} catch (MailException e) {

e.printStackTrace();

}

}

}

MimeMessageHelper 객체를 생성할 때에는 Mail Message를 구성할 때 사용될 MimeMessage 객체를 전달한다. MimeMessageHelper 생성자의 두 번째

파라미터는 MultiPart 여부를 설정하며 true로 지정해야 첨부 파일을 추가할 수 있다.

MimeMessage message = mailSender.createMimeMessage();

MimeMessageHelper messageHelper = new MimeMessageHelper(message, true /\*MultiPart 여부\*/, "euc-kr");

파일 첨부를 하기 위해서는 먼저 파일 정보를 제공할 javax.annotation.DataSource를 생성해 주어야 한다. Activation API는 파일을 데이터로 사용하는

FileDataSource Class를 제공하고 있으므로 이 Class를 사용하면 파일을 Mail에 첨부할 수 있다.

FileDataSource를 만들었다면 MimeMessageHelper.addAttachment() 메서드를 이용해서 파일을 첨부하면 된다.

DataSource dataSource = new FileDataSource("c:\\안내문.docx");

messageHelper.addAttachment(MimeUtility.encodeText("안내문.docx", "euc-kr", "B"), dataSource);

MimeMessageHelper가 생성한 Mail Message는 생성자를 통해 전달받은 MimeMessage 객체에 전달되므로, MimeMessageHelper로 Mail 내용을 구성한

후에는 JavaMailSender의 send() 메서드에 MimeMessage 객체를 전달해서 Mail을 발송하면 된다.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2.2 MimeMessageHelper를 이용한 인라인 Resource 추가

: MimeMessageHelper.addInline() 메서드를 사용하면 Mail 내용에 이미지나 파일 등을 삽입할 수 있다. 아래 Code는 사용 예이다.

public class MimeInlineRegistrationNotifier implements RegistrationNotifier {

private JavaMailSender mailSender;

public void setMailSender(JavaMailSender mailSender) {

this.mailSender = mailSender;

}

@Override

public void sendEmail(Member member) {

MimeMessage message = mailSender.createMimeMessage();

try {

MimeMessageHelper messageHelper = new MimeMessageHelper(message,

true, "euc-kr");

messageHelper.setSubject("[Inline] 회원 가입 안내");

String htmlContent = "<strong>안녕하세요</strong>, 반갑습니다."

+ "<img src='cid:signature' />";

messageHelper.setText(htmlContent, true);

messageHelper.setFrom("madvirus@wemade.com", "최범균");

messageHelper.setTo(new InternetAddress(member.getEmail(), member

.getName(), "euc-kr"));

messageHelper.addInline("signature", new FileDataSource(

"sign.jpg"));

} catch (Throwable e) {

e.printStackTrace();

return;

}

try {

mailSender.send(message);

} catch (MailException e) {

e.printStackTrace();

}

}

}

인라인 Resource을 삽입할 때 주의할 점은 setText() 메서드를 호출한 이후에 addInline() 메서드를 호출해야 한다는 점이다. 그렇지 않을 경우 인라인 Resource이

올바르게 삽입되지 않는다.