

28장 스프링에서 지원하는 여러 가지 기능

28.1 다중 파일 업로드하기

28.2 썸네일 이미지 사용하기

28.3 HTML 형식 메일 보내기

28.4 스프링 인터셉터 사용하기

28.5 인터셉터 이용해 요청명에서 뷰이름 가져오기

28.1 다중 파일 업로드하기

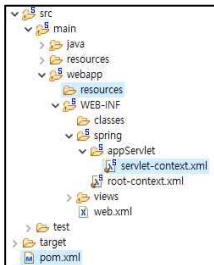
- 스프링의 CommonsMultipartResolver 클래스를 이용하면 여러 개의 파일을 한꺼번에 업로드할 수 있음

CommonsMultipartResolver 클래스 속성

속성	설명
maxUploadSize	최대로 업로드가 가능한 파일의 크기를 설정합니다.
maxInMemorySize	디스크에 임시 파일을 생성하기 전 메모리에 보관할 수 있는 최대 바이트 크기를 설정합니다.
defaultEncoding	전달되는 매개변수의 인코딩을 설정합니다.

28.1 다중 파일 업로드하기

1. 다음과 같이 파일을 준비합니다.



28.1 다중 파일 업로드하기

2. 파일 업로드에 필요한 라이브러리를 설치하도록 pom.xml을 작성합니다.

코드 28-1 pro28/pom.xml

```
...  
<dependency>  
  <groupId>commons-fileupload</groupId>  
  <artifactId>commons-fileupload</artifactId>  
  <version>1.2.1</version>  
</dependency>  
<dependency>  
  <groupId>commons-io</groupId>  
  <artifactId>commons-io</artifactId>  
  <version>1.4</version>  
</dependency>  
<dependencies>  
...  
...
```

28.1 다중 파일 업로드하기

3. servlet-context.xml 파일에서 CommonsMultipartResolver 클래스를 multipartResolver 빈으로 설정합니다.

코드 28-2 pro28/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml

...

```
<beans:bean id="multipartResolver"
            class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <beans:property name="maxUploadSize" value="52428800" />
    <beans:property name="maxInMemorySize" value="1000000" />
    <beans:property name="defaultEncoding" value="utf-8" />
</beans:bean>
```

...

28.1 다중 파일 업로드하기

4. 파일 업로드 및 다운로드 기능 컨트롤러를 구현하기 위한 자바 파일들을 준비합니다.



28.1 다중 파일 업로드하기

5. 먼저 파일 다운로드 컨트롤러인 `FileDownloadController` 클래스를 다음과 같이 작성합니다.
버퍼 기능을 이용해 빠르게 브라우저로 이미지 파일을 전송합니다.

코드 28-3 `pro28/src/main/java/com/myspring/pro28/ex01/FileDownloadController.java`

```
package com.myspring.pro28.ex01;

...

@Controller
public class FileDownloadController {
    private static String CURR_IMAGE_REPO_PATH = "c:\\spring\\image_repo";

    @RequestMapping("/download")
    protected void download(@RequestParam("imageFileName") String imageFileName,
        HttpServletResponse response) throws Exception {
        OutputStream out = response.getOutputStream();
        String downFile = CURR_IMAGE_REPO_PATH + "\\\" + imageFileName;
        File file = new File(downFile);
        response.setHeader("Cache-Control", "no-cache");
        response.addHeader("Content-disposition", "attachment; fileName=\"" + imageFileName);
        FileInputStream in = new FileInputStream(file);
        byte[] buffer = new byte[1024 * 8];
        while (true) {
            int count = in.read(buffer);
            if (count == -1) break;
            out.write(buffer, 0, count);
        }
        in.close();
        out.close();
    }
}
```

파일 저장 위치를 지정합니다.

다운로드할 이미지 파일 이름을 전달합니다.

다운로드할 ~~다운로드할~~ 파일 객체를 생성합니다.

헤더에 파일 이름을 설정합니다.

버퍼를 이용해 한 번에 8K(byte)씩 브라우저로 전송합니다.

코드 28-6 `pro28/src/main/webapp/WEB-INF/views/result.jsp`

```
<div class="result-images">
<c:forEach var="imageFileName" items="${map.fileList}" >
    
    <br><br>
</c:forEach>
</div>
```

업로드한 파일들을 `forEach`문을 이용해 `` 태

28.1 다중 파일 업로드하기

코드 28-4 pro28/src/main/java/com/myspring/pro28/ex01/FileUploadController.java

```
package com.myspring.pro28.ex01;

...
@Controller
public class FileUploadController {
    private static final String CURR_IMAGE_REPO_PATH = "c:\\spring\\image_repo";
    @RequestMapping(value="/form")

    public String form() {
        return "uploadForm";
    }

    @RequestMapping(value="/upload",method = RequestMethod.POST)
    public ModelAndView upload(MultipartHttpServletRequest multipartRequest,
        HttpServletResponse response) throws Exception{
        multipartRequest.setCharacterEncoding("utf-8");
        Map map = new HashMap();
        Enumeration enu=multipartRequest.getParameterNames();
        while(enu.hasMoreElements()){
            String name=(String)enu.nextElement();
            String value=multipartRequest.getParameter(name);
            map.put(name,value);
        }
        List fileList= fileProcess(multipartRequest);
        map.put("fileList", fileList);
        ModelAndView mav = new ModelAndView();
        mav.addObject("map", map);
        mav.setViewName("result");
        return mav;
    }
}
```

업로드창인 uploadForm.jsp를 반환합니다.

매개변수 정보와 파일 정보를 저장할 Map을 생성합니다.

전송된 매개변수 값을 key/value로 map에 저장합니다.

파일을 업로드한 후 반환된 파일 이름이 저장된 fileList를 다시 map에 저장합니다.

map을 결과창으로 포워드합니다.

```
16 var cnt=1;
17 function fn_addFile(){
18     $("#d_file").append("<br>"+<input type='file' name='file'+cnt+" />")
19     cnt++;
20 }
21 </script>
22 </head>
23 <body>
24 <h1>파일 업로드 하기</h1>
25 <form method="post" action="{contextPath}/upload" enctype="multipart/form
26     <label>아이디:</label>
27     <input type="text" name="id"><br>
28     <label>이름:</label>
29     <input type="text" name="name"><br>
30     <input type="button" value="파일추가" onClick="fn_addFile()"><br>
31     <div id="d_file">
32 </div>
33     <input type="submit" value="업로드"/>
34 </form>
35 </body>
36 </html>
```


28.1 다중 파일 업로드하기

```

private List<String> fileProcess(MultipartHttpServletRequest multipartRequest)
throws Exception{
    List<String> fileList= new ArrayList<String>();
    Iterator<String> fileNames = multipartRequest.getFileNames();
    while(fileNames.hasNext()){
        String fileName = fileNames.next();
        MultipartFile mFile = multipartRequest.getFile(fileName);
        String originalFileName=mFile.getOriginalFilename();
        fileList.add(originalFileName);
        File file = new File(CURR_IMAGE_REPO_PATH + "\\" + fileName);
        if(mFile.getSize()!=0){
            if(! file.exists()){
                if(file.getParentFile().mkdirs()){
                    file.createNewFile();
                }
            }
            mFile.transferTo(new File(CURR_IMAGE_REPO_PATH + "\\" + originalFileName));
        }
    }
    return fileList;
}

```

첨부된 파일 이름을 가져옵니다.

파일 이름에 대한 MultipartFile 객체를 가져옵니다.

실제 파일 이름을 가져옵니다.

파일 이름을 하나씩 fileList에 저장합니다.

첨부된 파일이 있는지 체크합니다.

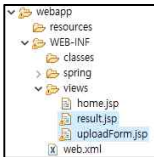
경로에 파일이 없으면 그 경로에 해당하는 디렉터리를 만든 후 파일을 생성합니다.

임시로 저장된 multipartFile을 실제 파일로 전송합니다

첨부한 파일 이름이 저장된 fileList를 반환합니다.

28.1 다중 파일 업로드하기

7. 파일 업로드창과 업로드한 파일을 표시해 주는 결과창을 나타낼 JSP 파일을 다음과 같이 준비합니다.



28.1 다중 파일 업로드하기

8.파일 업로드창인 uploadForm.jsp를 다음과 같이 작성합니다.

코드 28-5 pro28/src/main/webapp/WEB-INF/views/uploadForm.jsp

```
...
<!DOCTYPE html >
<html>
<head>
  <meta "charset=utf-8">
  <title>파일 업로드 하기</title>
</script src="http://code.jquery.com/jquery-latest.js"></script>
<script>
  var cnt=1;
  function fn_addFile(){
    $("#d_file").append("<br>"+<input type='file' name='file'+cnt+' ' />");
    cnt++;
  }
</script>
</head>
<body>
  <h1>파일 업로드 하기</h1>
  <form method="post" action="${contextPath}/upload" enctype="multipart/form-data">
  <label>아이디:</label>
  <input type="text" name="id">
  <label>이름:</label>
  <input type="text" name="name">
  <input type="button" value="파일 추가" onClick="fn_addFile()">
  <div id="d_file">
  <input type="submit" value="업로드"/>
</form>
</body>
</html>
```

파일 업로드 name 값을 다르게 하는 변수입니다.

파일 추가를 클릭하면 동작으로 파일 업로드를 추가합니다. name 속성의 값으로 file+cnt를 설정함으로써 값을 다르게 해줍니다.

파일 업로드 시 enctype은 multipart/form-data로 설정해야 합니다.

텍스트 박스에 ID를 입력 받아 전송합니다.

텍스트 박스에 이름을 입력 받아 전송합니다.

자바스크립트를 이용해 (div) 태그 안에 파일 업로드를 추가합니다.

파일 추가를 클릭하면 동작으로 파일 업로드를 추가합니다.

28.1 다중 파일 업로드하기

9. 결과창을 나타내는 result.jsp를 다음과 같이 작성합니다

코드 28-6 pro28/src/main/webapp/WEB-INF/views/result.jsp

```
...
<!DOCTYPE html>
<html>
<head>
  <meta "charset=UTF-8">
  <title>결과창</title>
</head>
<body>
  <h1>업로드가 완료되었습니다.</h1>
  <label>아이디:</label>
  <input type="text" name="id" value='${map.id}' readonly><br>
  <label>이름:</label>
  <input type="text" name="name" value='${map.name}' readonly><br>

  <div class="result-images">
    <c:forEach var="imageFileName" items="${map.fileList}" >
      
      <br><br>
    </c:forEach>
  </div>
  <a href='${contextPath }/form'> 다시 업로드 하기 </a>
</body>
</html>
```

코드 28-4 pro28/src/main/java/com/myspring/pro28/ex01/FileUploadController.java

```
package com.myspring.pro28.ex01;

...
@Controller
public class FileUploadController {
  private static final String CURR_IMAGE_REPO_PATH = "c:\\spring\\image_repo";
  @RequestMapping(value="/form")
  public String form() {
    return "uploadForm";
  }
```

업로드창인 uploadForm.jsp를 반환합니다.

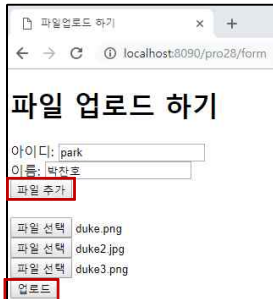
map으로 넘어온 매개변수 값을 표시합니다.

map으로 넘어온 매개변수 값을 표시합니다.

업로드한 파일들을 forEach문을 이용해 태그에 표시합니다.

28.1 다중 파일 업로드하기

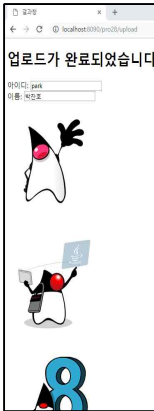
10. `http://localhost:8090/pro28/form`으로 요청하여 ID와 이름을 입력하고 파일추가를 클릭하여 `duke.png`, `duke2.jpg`, `duke3.png` 세 개의 파일을 첨부합니다. 그리고 업로드를 클릭합니다.



The screenshot shows a web browser window with the title '파일업로드 하기'. The address bar displays 'localhost:8090/pro28/form'. The page content includes the title '파일 업로드 하기' in large black text. Below the title, there are two input fields: '아이디:' with the value 'park' and '이름:' with the value '박찬호'. A red rectangular box highlights the '파일 추가' button. Below this, there is a list of three file selection items, each consisting of a '파일 선택' button and a file name: 'duke.png', 'duke2.jpg', and 'duke3.png'. A red rectangular box highlights the '업로드' button at the bottom of the form.

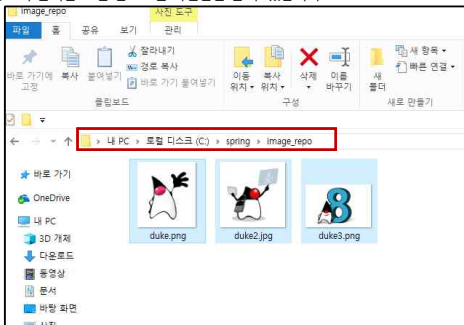
28.1 다중 파일 업로드하기

11. 결과창으로 넘어가면서 전송된 매개변수 값들과 업로드된 이미지 세 개가 표시됩니다



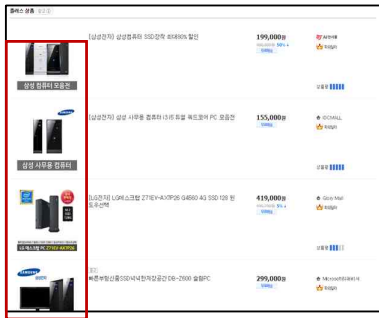
28.1 다중 파일 업로드하기

12. 지정한 경로의 폴더를 보면 업로드된 파일들을 볼 수 있습니다



28.2 썸네일 이미지 사용하기

- 썸네일 이미지를 이용하면 상품 목록 나열 시 빠르게 표시할 수 있음

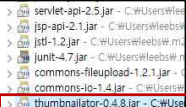


28.2 썸네일 이미지 사용하기

1. 다음과 같이 pom.xml에 썸네일 라이브러리를 설정하면 thumbnailator-0.4.8.jar가 설치됩니다.

코드 28-7 pro28/pom.xml

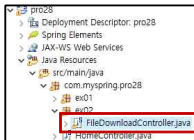
```
...  
<dependency>  
  <groupId>net.coobird</groupId>  
  <artifactId>thumbnailator</artifactId>  
  <version>0.4.8</version>  
</dependency>  
<dependencies>
```



```
> <img alt="JAR icon" data-bbox="338 625 355 642"/> servlet-api-2.5.jar - C:\Users\Wleebsw...  
> <img alt="JAR icon" data-bbox="338 652 355 669"/> jsp-api-2.1.jar - C:\Users\Wleebsw...  
> <img alt="JAR icon" data-bbox="338 679 355 696"/> jstl-1.2.jar - C:\Users\Wleebsw...  
> <img alt="JAR icon" data-bbox="338 706 355 723"/> junit-4.7.jar - C:\Users\Wleebsw...  
> <img alt="JAR icon" data-bbox="338 733 355 750"/> commons-fileupload-1.2.1.jar - C...  
> <img alt="JAR icon" data-bbox="338 760 355 777"/> commons-io-1.4.jar - C:\Users\W...  
> <img alt="JAR icon" data-bbox="338 787 355 804"/> thumbnailator-0.4.8.jar - C:\Use...
```

28.2 썸네일 이미지 사용하기

2. 컨트롤러에 요청해 썸네일 이미지를 생성한 후 다운로드해 보겠습니다. 다음 위치에 FileDownloadController 클래스 파일을 준비합니다.



28.2 썸네일 이미지 사용하기

3. 원본 이미지에 대해 썸네일 이미지 파일을 생성한 후 다운로드할 수 있도록 다음과 같이 작성합니다.

코드 28-8 pro28/src/main/java/com/myspring/pro28/ex01/FileDownloadController.java

```
package com.myspring.pro28.ex02;
...
import net.coobird.thumbnailator.Thumbnails;

@Controller
public class FileDownloadController {
    private static String CURR_IMAGE_REPO_PATH = "c:\\spring\\image_repo";
    @RequestMapping("/download")
    protected void download(@RequestParam("imageFileName") String imageFileName,
                            HttpServletResponse response) throws Exception {
        OutputStream out = response.getOutputStream();
        String filePath = CURR_IMAGE_REPO_PATH + "\\\" + imageFileName;
        File image = new File(filePath);
        int lastIndex = imageFileName.lastIndexOf(".");
        String fileName = imageFileName.substring(0, lastIndex);
```

확장자를 제외한 원본 이미지 파일
의 이름을 가져옵니다.

28.2 썸네일 이미지 사용하기

```
File thumbnail = new File(CURR_IMAGE_REPO_PATH+"\\ "+ "thumbnail"+"\\ "+ fileName+".png");
```

```
if (image.exists()) {  
    thumbnail.getParentFile().mkdirs();  
    Thumbnails.of(image).size(50,50).outputFormat("png").toFile(thumbnail);  
}
```

원본 이미지 파일 이름과 같은 이름의 썸네일
파일에 대한 File 객체를 생성합니다.

```
FileInputStream in = new FileInputStream(thumbnail);  
byte[] buffer = new byte[1024 * 8];  
while (true) {  
    int count = in.read(buffer);  
    if (count == -1)  
        break;  
    out.write(buffer, 0, count);  
}
```

원본 이미지 파일을 가로세로가
50픽셀인 png 형식의 썸네일 이미
지 파일로 생성합니다.

```
in.close();  
out.close();
```

생성된 썸네일 파일을 브라우저로
전송합니다.

```
}
```

```
}
```

28.2 썸네일 이미지 사용하기

4. 다음은 실행 결과입니다. /form으로 요청한 후 세 개의 이미지 파일을 첨부하고 업로드를 클릭합니다.



파일업로드 하기

← → ↻ ⓘ localhost:8090/pro28/form

파일 업로드 하기

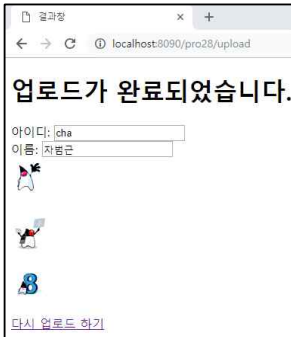
아이디:

이름:

<input type="button" value="파일 선택"/>	duke.png
<input type="button" value="파일 선택"/>	duke2.jpg
<input type="button" value="파일 선택"/>	duke3.png

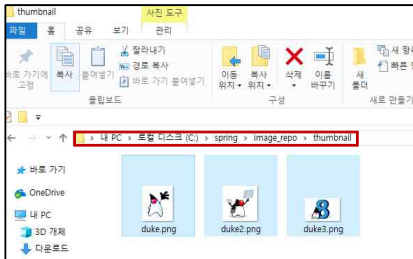
28.2 썸네일 이미지 사용하기

5. 결과창에 각 이미지에 대한 썸네일 이미지가 표시됩니다



28.2 썸네일 이미지 사용하기

6. 이미지 저장 폴더 하위에 있는 thumbnail 폴더를 보면 다음과 같이 썸네일 이미지들이 있습니다.



28.2 썸네일 이미지 사용하기

• 28.2.1 썸네일 이미지 바로 출력하기

1. 원본 이미지를 썸네일 이미지로 바로 출력하는 방법은 다음과 같습니다.

코드 28-9 pro28/src/main/java/com/myspring/pro28/ex01/FileUploadController.java

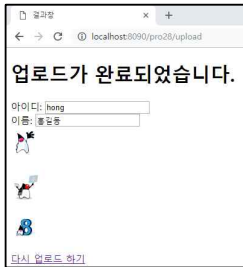
```
...
@RequestMapping("/download")
protected void download(@RequestParam("imageFileName") String imageFileName,
                        HttpServletResponse response) throws Exception {
    OutputStream out = response.getOutputStream();
    String filePath = CURR_IMAGE_REPO_PATH + "\\\" + imageFileName;
    File image = new File(filePath);
    int lastIndex = imageFileName.lastIndexOf(".");
    String fileName = imageFileName.substring(0, lastIndex);
    File thumbnail = new File(CURR_IMAGE_REPO_PATH + "\\\" + \"thumbnail\" + \"\\\" + fileName + \".png\");
    if(image.exists()) {
        Thumbnails.of(image).size(50,50).outputFormat(\"png\").toOutputStream(out);
    }else{
        return;
    }
    byte[] buffer = new byte[1024 * 8];
    out.write(buffer);
    out.close();
}
}
```

원본 이미지에 대한 썸네일 이미지를 생성한 후
OutputStream 객체에 할당합니다

썸네일 이미지를 OutputStream 객체를
이용해 브라우저로 전송합니다.

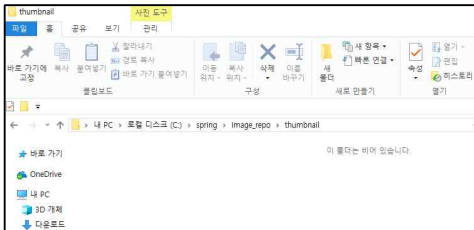
28.2 썸네일 이미지 사용하기

2. 썸네일 이미지 저장 폴더의 이미지들을 삭제한 후 다시 실행해 보세요.



28.2 썸네일 이미지 사용하기

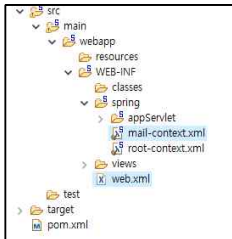
3. 브라우저에 표시되는 결과는 앞에서와 같지만 해당 경로의 폴더를 보면 썸네일 이미지 파일은 따로 생성되지 않았습니다.



28.3 스프링 이메일 사용하기

- 스프링에선 이메일 라이브러리를 이용해서 쉽게 이메일 기능을 구현할 수 있음
- 구글의 SMTP 서버를 이용해서 이메일 기능 구현

1. 이메일 기능 설정을 위한 XML 파일들을 준비합니다.



28.3 스프링 이메일 사용하기

2. pom.xml 파일을 다음과 같이 작성합니다. 이메일 기능을 사용하기 위해 코어 스프링 라이브러리 버전을 4.1.1.RELEASE로 변경한 후 관련 라이브러리를 추가합니다

코드 28-10 pro28/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd"> <modelVersion>4.0.0</modelVersion>
  <groupId>com.myspring</groupId>
  <artifactId>pro28</artifactId>
  <name>pro28</name>
  <packaging>war</packaging>
  <version>1.0.0-BUILD-SNAPSHOT</version>
  <properties>
    <java-version>1.6</java-version>
    <org.springframework-version>4.1.1.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
```

스프링 라이브러리 버전을 4.1.1.RELEASE로 변경합니다.

28.3 스프링 이메일 사용하기

```
<org.slf4j-version>1.6.6</org.slf4j-version>  
</properties>
```

...

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-context-support</artifactId>  
  <version>${org.springframework-version}</version>  
</dependency>
```

스프링 4 이상 사용 시
추가합니다.

```
<dependency>  
  <groupId>javax.mail</groupId>  
  <artifactId>javax.mail-api</artifactId>  
  <version>1.5.4</version>  
</dependency>  
<dependency>  
  <groupId>com.sun.mail</groupId>  
  <artifactId>javax.mail</artifactId>  
  <version>1.5.3</version>  
</dependency>
```

자바 이메일 라이브러리들을 추가합니다.

...

28.3 스프링 이메일 사용하기

3. web.xml에서는 설정 파일이 여러 개인 경우 톰캣 컨테이너 실행 시 spring 폴더에 있는 모든 설정 파일들을 읽어 들이도록 지정합니다

코드 28-11 pro28/src/main/webapp/WEB-INF/web.xml

```
...
<!-- The definition of the Root Spring Container shared by all Servlets and
Filters -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/*.xml</param-value>
</context-param>
...
```

28.3 스프링 이메일 사용하기

4. 구글 SMTP 서버와 연동해서 실습하므로 스프링의 `JavaMailSenderImpl` 클래스를 이용해 메일 서버와 관련된 정보를 설정하도록 `mail-context.xml`을 작성합니다.

코드 28-12 pro28/src/main/webapp/WEB-INF/spring/mail-context.xml

```

...
<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
  <property name="host" value="smtp.gmail.com"/>
  <property name="port" value="465" />
  <property name="username" value="****@gmail.com" />
  <property name="password" value="메일비밀번호"/>
  <property name="javaMailProperties">
    <props>
      <prop key="mail.transport.protocol">smtp</prop>
      <prop key="mail.smtp.auth">true</prop>
      <prop key="mail.smtp.starttls.enable">true</prop>
      <prop key="mail.smtp.socketFactory.class">javax.net.ssl.SSLSocketFactory</prop>
      <prop key="mail.debug">true</prop>
    </props>
  </property>
</bean>

<bean id="preConfiguredMessage" class="org.springframework.mail.SimpleMailMessage">
  <property name="to" value="수신메일주소"></property>
  <property name="from" value="****@gmail.com"></property>
  <property name="subject" value="테스트 메일입니다."/>
</bean>
...

```

구글 SMTP 메일 서버의 포트는 465 또는 587입니다.

메일을 보냈을 때 실제 수신자에게 메일을 보내는 host 서버에 구글의 SMTP 서버를 설정합니다.

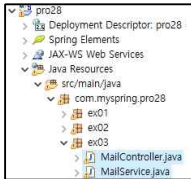
자신의 구글 메일 계정과 비밀번호를 설정합니다.

메일 전달 프로토콜 세부 속성을 설정합니다.

수신자에게 메일을 정기적으로 보내는 경우 송수신 메일 주소와 제목을 미리 지정해서 보낼 수 있습니다.

28.3 스프링 이메일 사용하기

5. 이제 실제 자바 코드로 메일을 전송해 보겠습니다. 다음과 같이 자바 클래스 파일들을 준비합니다.



28.3 스프링 이메일 사용하기

6. MailController 클래스를 다음과 같이 작성합니다. @EnableAsync를 지정해서 메서드를 호출할 경우 비동기로 동작하게 하는 @Async 애너테이션 기능을 사용할 수 있습니다

코드 28-13 pro28/src/main/java/com/myspring/pro28/ex03/MailController.java

```
package com.myspring.pro28.ex03;

...
@Controller
@EnableAsync
public class MailController {
    @Autowired
    private MailService mailService;

    @RequestMapping(value = "/sendMail.do", method = RequestMethod.GET)
    public void sendSimpleMail(HttpServletRequest request,
                              HttpServletResponse response) throws Exception{
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        mailService.sendMail("*****@naver.com", "테스트 메일", "안녕하세요. 테스트 메일입니다.");
        mailService.sendPreConfiguredMail("테스트 메일입니다.");
        out.print("메일을 보냈습니다!!");
    }
}
```

비동기 선언

mailService의 sendMail() 메서드로 메일 관련 값(주소, 제목, 내용)을 전달합니다.

mail-context.xml에 설정한 메일 주소로 내용을 보냅니다.

28.3 스프링 이메일 사용하기

7. 이번에는 MailService 클래스입니다.

코드 28-14 pro28/src/main/java/com/myspring/pro28/ex03/MailService.java

```
package com.myspring.pro28.ex03;
```

```
...
```

```
@Service("mailService")
```

```
public class MailService {
```

```
    @Autowired
```

```
    private JavaMailSender mailSender;
```

```
    @Autowired
```

```
    private SimpleMailMessage preConfiguredMessage;
```

mail-context.xml에서 설정한 빈을
자동으로 주입합니다.

```
@Async
```

```
public void sendMail(String to, String subject, String body) {
```

```
    MimeMessage message = mailSender.createMimeMessage();
```

MimeMessage 타입 객체를
생성합니다.

```
    try {
```

```
        MimeMessageHelper messageHelper =
```

```
            new MimeMessageHelper(message, true, "UTF-8");
```

```
        messageHelper.setCc("#####@naver.com");
```

이메일 보내기 위해 MimeMessageHelper
객체를 생성합니다.

```
        messageHelper.setFrom("xxxxxx@naver.com", "홍길동");
```

```
        messageHelper.setSubject(subject);
```

```
        messageHelper.setTo(to);
```

```
        messageHelper.setText(body);
```

```
        mailSender.send(message);
```

메일 수신 시 지정한 이름으로 표시되게 합니다.
지정하지 않으면 송신 메일 주소가 표시됩니다.

제목, 수신처 내용을 설정해 메일을 보냅니다.

```
    } catch (Exception e) {
```

28.3 스프링 이메일 사용하기

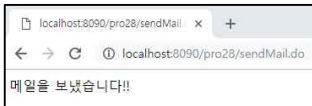
```
e.printStackTrace();
    }
}

@Async
public void sendPreConfiguredMail(String message) {
    SimpleMailMessage mailMessage = new SimpleMailMessage(preConfiguredMessage);
    mailMessage.setText(message);
    mailSender.send(mailMessage);
}
}
```

mail-context.xml에서 미리 설정한 수신 주소로
메일 내용을 보냅니다

28.3 스프링 이메일 사용하기

8. `http://localhost:8090/pro28/sendMail.do`로 요청합니다.

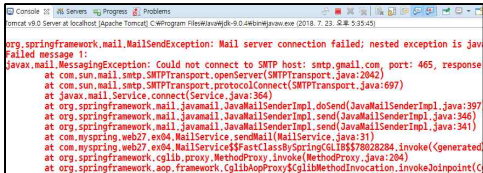


9. 설정한 메일 계정에 접속하여 수신 메일함을 확인해 보면 홍길동으로부터 메일이 온 것을 볼 수 있습니다.



28.3 스프링 이메일 사용하기

메일 보내기 실행 시 오류가 발생했다면?



```
org.springframework.mail.MailSendException: Mail server connection failed; nested exception is java
Failed message 1:
javax.mail.MessagingException: Could not connect to SMTP host: smtp.gmail.com, port: 465, response
    at com.sun.mail.smtp.SMTPTransport.openServer(SMTPTransport.java:2042)
    at com.sun.mail.smtp.SMTPTransport.protocolConnect(SMTPTransport.java:697)
    at javax.mail.Service.connect(Service.java:364)
    at org.springframework.mail.javamail.JavaMailSenderImpl.doSend(JavaMailSenderImpl.java:397)
    at org.springframework.mail.javamail.JavaMailSenderImpl.send(JavaMailSenderImpl.java:346)
    at org.springframework.mail.javamail.JavaMailSenderImpl.send(JavaMailSenderImpl.java:341)
    at com.nyspring.web27.ex04.MailService.sendMail(MailService.java:31)
    at com.nyspring.web27.ex04.MailService$$FastClassBySpringCGLIB$$70028284.invoke(<generated>)
    at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204)
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(C
```

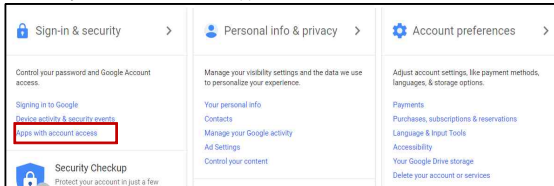
28.3 스프링 이메일 사용하기

1. 구글 계정으로 로그인한 후 사용자 계정의 Google 계정을 클릭합니다.



28.3 스프링 이메일 사용하기

2. Signin & security(로그인 및 보안) 항목 중 Apps with account access를 클릭합니다.



3. 항목들 중 Allow less secure apps를 ON으로 설정합니다.



28.4 HTML 형식 메일 보내기

- 상품 이미지나 링크 등으로 구성된 HTML 형식의 이메일로, 이를 클릭하면 해당 상품 페이지로 이동

상품 이미지와 링크가 포함된 판촉 이메일

 <p>CLOVIS</p> <p>8% 10% OFF \$5,100</p> <p>[33데이] 여성 클라비스 군일 249종 89% 6,333원</p>	 <p>KAZTO</p> <p>8% 10% OFF \$2,333</p> <p>[33데이] 품세일-이 생활용품 90%OFF 위메프가 2,333원</p>
 <p>8% 10% OFF \$5,100</p> <p>[33데이] 오뚜기 컵누를 10+5개 위메프가 11,333원</p>	 <p>8% 10% OFF \$5,100</p> <p>[33데이] 품귀잡이 맛집 +30% 위메프가 1,333원</p>
 <p>8% 10% OFF \$5,100</p> <p>[33데이] 퓨어 베게슬 1+1 / 봄 이불 위메프가 5,333원</p>	 <p>8% 10% OFF \$5,100</p> <p>[33데이] 여성구두TOP은 스타일모델+30% 위메프가 3,333원</p>
 <p>8% 10% OFF \$5,100</p> <p>[33데이] 품세일-이 생활용품 90%OFF 위메프가 2,333원</p>	 <p>LAVER</p> <p>[33데이] 여성 클라비스 군일 249종 89% 6,333원</p>

28.3 스프링 이메일 사용하기

1. 다음과 같이 MailController 클래스에 HTML 태그를 작성해 StringBuffer에 저장한 후 문자열로 내용을 보냅니다.

코드 28-15 pro28/src/main/java/com/myspring/pro28/ex04/MailController.java

```
package com.myspring.pro28.ex04;

...

@Controller
@EnableAsync
public class MailController {

    @Autowired
    private MailService mailService;

    @RequestMapping(value = "/sendMail.do", method = RequestMethod.GET)
    public void sendSimpleMail(HttpServletRequest request,
                               HttpServletResponse response) throws Exception{
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
```

28.3 스프링 이메일 사용하기

```

StringBuffer sb = new StringBuffer();  *————— StringBuffer 변수 sb를 선언합니다.
sb.append("<html><body>");
sb.append("<meta http-equiv='Content-Type' content='text/html; charset=euc-kr'>");
sb.append("<h1>"+제품소개+"<h1><br>");
sb.append("신간 도서를 소개합니다.<br><br>");
sb.append("<a href='http://www.kyobobook.co.kr/product/detailViewKor.laf?ejkGb=KOR  

    &mallGb=KOR&barcode=9788956746425&orderClick=LAG&Kc=#N'>");
sb.append("<img src='http://image.kyobobook.co.kr/images/book/xtlarge/425/  

    x9788956746425.jpg' /> </a><br>");
sb.append("</a>");
sb.append("<a href='http://www.kyobobook.co.kr/product/detailViewKor.laf?  

    ejkGb=KOR&mallGb=KOR&barcode=9788956746425&orderClick=LAG&Kc=#N'>  

    상품보기</a>");
sb.append("</body></html>");  *————— 문자열로 HTML 태그를 작성한 후 sb에 저장합니다.
String str=sb.toString();  *————— 문자열로 변환합니다.
mailService.sendMail("*****@naver.com", "신상품을 소개합니다. ", str);
out.print("메일을 보냈습니다!!");  *————— HTML 형식의 내용을 메일로 보냅니다.
}
}

```

28.3 스프링 이메일 사용하기

2. 메일 내용이 HTML로 표시되게 하려면 반드시 `MimeMessageHelper`의 `setText()` 메서드의 두 번째 인자값을 `true`로 설정해야 합니다.

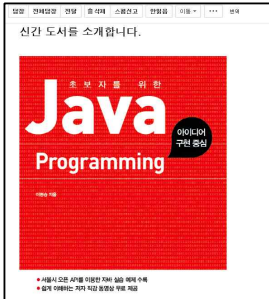
코드 28-16 `pro28/src/main/java/com/myspring/pro28/ex04/MailService.java`

```
...
@Async
public void sendMail(String to, String subject, String body){
    MimeMessage message = mailSender.createMimeMessage();
    try {
        MimeMessageHelper messageHelper = new MimeMessageHelper(message, true, "UTF-8");
        messageHelper.setSubject(subject);
        messageHelper.setTo(to);
        messageHelper.setFrom("*****@naver.com", "홍길동");
        messageHelper.setText(body, true);
        mailSender.send(message);
    }catch(Exception e){
        e.printStackTrace();
    }
}
...
```

반드시 `true`로 설정해야 합니다

28.3 스프링 이메일 사용하기

3. 브라우저에 요청하여 메일을 수신합니다. 그리고 수신 메일 본문에 있는 이미지를 클릭합니다



28.3 스프링 이메일 사용하기

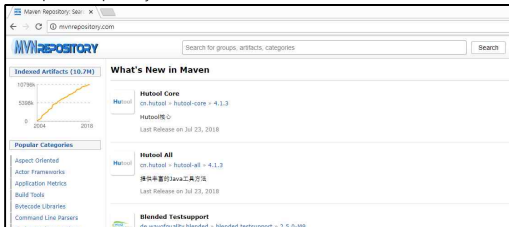
4. 그러면 지정한 웹 페이지가 브라우저에 나타납니다

The screenshot shows the KYOBO 교보문고 website. At the top, there's a navigation bar with links like '국내도서', '외국도서', 'eBook', '웹소설', '기프트', '음반', and '종교정터'. Below this is a search bar and a breadcrumb trail: '홈 > 국내도서 > 컴퓨터/IT > 프로그래밍 언어 > Java'. The main content area features a large red book cover for 'Java Programming' by O'Reilly. To the right of the cover, the text reads '초보자를 위한 Java Programming 아이디어 구현 중심' and '이탈송 저음 | 알보문하사 | 2015년 08월 14일 출간'. Below this is a star rating of 4.5 and a price of 27,000 KRW. The page also includes a '구매하기' button and a '책의 다른 상품 정보' section at the bottom.

28.3 스프링 이메일 사용하기

- pom.xml에 설정하는 라이브러리 정보를 찾는 방법
 - MySQL 드라이버에 대한 설정 정보를 가져오는 과정

1. <http://mvnrepository.com>으로 접속합니다.

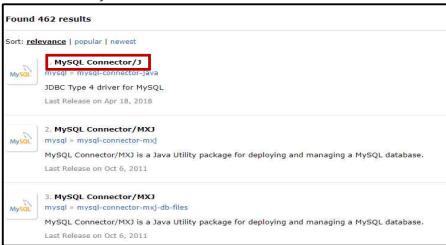


28.3 스프링 이메일 사용하기

2. 검색창에 mysql이라 입력하고 Search를 클릭합니다.



3. 검색 목록에서 MySQL Connector/J를 클릭합니다.



28.3 스프링 이메일 사용하기

4. 6.0.6 버전을 클릭합니다.

Categories

MySQL Drivers

Tags

mysql

database

connector

driver

Used By

2,619 artifacts

Central (67)

Jahia (1)

	Version	Repository
8.0.x	8.0.11	Central
	8.0.9-rc	Central
	8.0.8-dmr	Central
	8.0.7-dmr	Central
6.0.x	6.0.6	Central
	6.0.5	Central
	6.0.4	Central
	6.0.3	Central
	6.0.2	Central
	5.1.46	Central
	5.1.45	Central
	5.1.44	Central

28.3 스프링 이메일 사용하기

5. Maven 탭의 <dependency> 태그 부분을 복사해 pom.xml에 붙여 넣습니다.

Note: There is a new version for this artifact

New Version8.0.11

MavenGradleSBTIvyGrapeLeiningenBuildr

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>6.0.6</version>
</dependency>
```

☒ Include comment with link to declaration

6. 스프링 부트에서 설정하려면 Gradle 탭을 클릭한 후 그루비로 된 정보를 복사해 붙여 넣습니다.

Note: There is a new version for this artifact

New Version8.0.11

MavenGradleSBTIvyGrapeLeiningenBuildr

```
// https://mvnrepository.com/artifact/mysql/mysql-connector-java
compile group: 'mysql', name: 'mysql-connector-java', version: '6.0.6'
```

☒ Include comment with link to declaration

28.3 스프링 이메일 사용하기

7. pom.xml에 설정한 후 라이브러리를 다운로드합니다.

```
163      <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
164      <dependency>
165          <groupId>mysql</groupId>
166          <artifactId>mysql-connector-java</artifactId>
167          <version>6.0.6</version>
168      </dependency>
```

8. Maven Dependency에 MySQL 드라이버가 설치된 것을 확인할 수 있습니다.

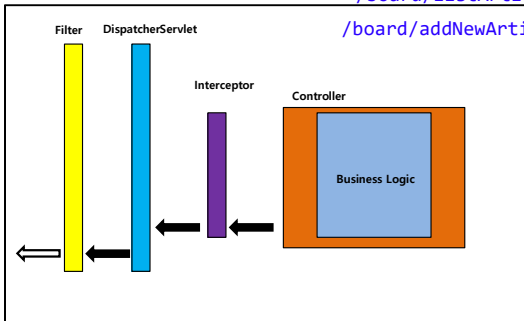
```
> junit-4.7.jar - C:\Users\wleeb\m2\re
> commons-fileupload-1.2.1.jar - C:\Use
> commons-io-1.4.jar - C:\Users\wleeb\
> thumbnailator-0.4.8.jar - C:\Users\wleeb
> spring-context-support-4.1.1.RELEASE.jar
> javax.mail-api-1.5.4.jar - C:\Users\wleeb
> activation-1.1.jar - C:\Users\wleeb\m
> javax.mail-1.5.3.jar - C:\Users\wleeb\m
> mysql-connector-java-6.0.6.jar - C:\Use
```

28.5 스프링 인터셉터 사용하기

인터셉터(Interceptor)

- 브라우저 요청 시 요청 메서드 호출 전후에 개발자가 원하는 기능을 수행함
- 필터와 기능이 유사하지만 필터보다 좀 더 자유롭게 위치를 변경해서 기능을 수행함
- 쿠키(Cookie) 제어, 파일 업로드 등의 작업을 수행함

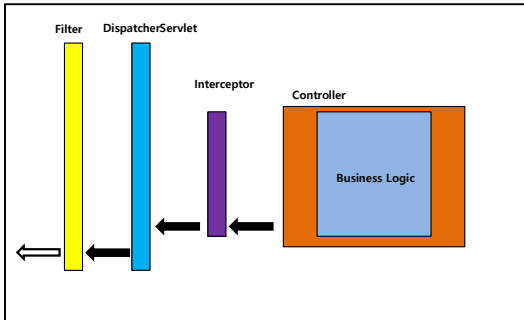
인터셉터와 필터 동작 과정



28.5 스프링 인터셉터 사용하기

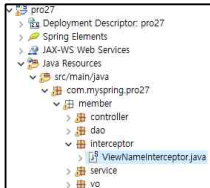
스프링의 **HandlerInterceptor** 클래스의 여러 가지 메서드

메서드	기능
preHandle()	컨트롤러 실행 전 호출됩니다.
postHandle()	컨트롤러 실행 후 DispatcherServlet이 뷰로 보내기 전에 호출됩니다
afterCompletion()	뷰까지 수행하고 나서 호출됩니다.



28.6 인터셉터 사용해 요청명에서 뷰이름 얻기

3. member 패키지 하위에 interceptor 패키지를 만든 후 ViewNameInterceptor 클래스를 작성합니다.



28.6 인터셉터 사용해 요청명에서 뷰이름 얻기

4. 인터셉터 수행 시 `preHandle()` 메서드로 전달된 `request`에서 추가한 `getViewName()` 메서드를 이용해 뷰이름 가져온 후 `request`에 바인딩합니다.

코드 28-23 `pro28/src/main/java/com/myspring/pro27/member/interceptor/ViewNameInterceptor.java`

```
package com.myspring.pro27.member.interceptor;

...

public class ViewNameInterceptor extends HandlerInterceptorAdapter {
    @Override
    public boolean preHandle(HttpServletRequest request,
                             HttpServletResponse response, Object handler) {
        try {
            String viewName = getViewName(request);
            request.setAttribute("viewName", viewName);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }
}
```

`getViewName()` 메서드를 이용해
브라우저의 요청명에서 뷰이름을
가져옵니다.

뷰이름을 `request`에 바인딩합니다.

28.3 스프링 이메일 사용하기

```
@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response,
                        Object handler, ModelAndView modelAndView) throws Exception {
}
```

```
@Override
public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
                           Object handler, Exception ex) throws Exception {
}
```

요청명에서 뷰이름을 반환합니다.

```
private String getViewName(HttpServletRequest request) throws Exception {
    String contextPath = request.getContextPath();
    String uri = (String) request.getAttribute("javax.servlet.include.request_uri");
    if (uri == null || uri.trim().equals("")) {
        uri = request.getRequestURI();
    }
    ...
}
```

28.6 인터셉터 사용해 요청명에서 뷰이름 얻기

5. 컨트롤러에서는 request에 바인딩된 뷰이름을 가져와 뷰리졸버로 반환합니다.

코드 28-24 pro28/src/main/java/com/myspring/pro28/member/controller/MemberControllerImpl.java

```
package com.myspring.pro27.member.controller;
...
@Controller("memberController")
public class MemberControllerImpl implements MemberController {
    @Autowired
    private MemberService memberService;
    @Autowired
    MemberVO memberVO ;

    @Override
    @RequestMapping(value="/member/listMembers.do" ,method = RequestMethod.GET)
    public ModelAndView listMembers(HttpServletRequest request,
                                   HttpServletResponse response) throws Exception {
        //String viewName = getViewName(request);
        String viewName = (String)request.getAttribute("viewName");
        List membersList = memberService.listMembers();
        ModelAndView mav = new ModelAndView(viewName);
        mav.addObject("membersList", membersList);
        return mav;
    }
}
```

인터셉터에서 바인딩된 뷰이름을 가져옵니다.

28.6 인터셉터 사용해 요청명에서 뷰이름 얻기

...

```
@RequestMapping(value = "/member/*Form.do", method = RequestMethod.GET)
private ModelAndView form(@RequestParam(value= "result", required=false) String result,
    HttpServletRequest request,
    HttpServletResponse response) throws Exception {
    //String viewName = getViewName(request);
    String viewName = (String)request.getAttribute("viewName");
    ModelAndView mav = new ModelAndView();
    mav.addObject("result",result);
    mav.setViewName(viewName);
    return mav;
}
```

...

인터셉터에서 바인딩된 뷰이름을 가져옵니다.

28.6 인터셉터 사용해 요청명에서 뷰이름 얻기

6. `http://localhost:8090/pro27/member/loginForm.do`로 로그인창을 요청하여 뷰이름을 인터셉터에서 가져옵니다. 그리고 이를 컨트롤러에 전달합니다

