SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Spring Framework

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

SpringFramework -
File Upload, Download, Interceptor

SAMSUNG SW ACADEMY FOR YOUTH

목차

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

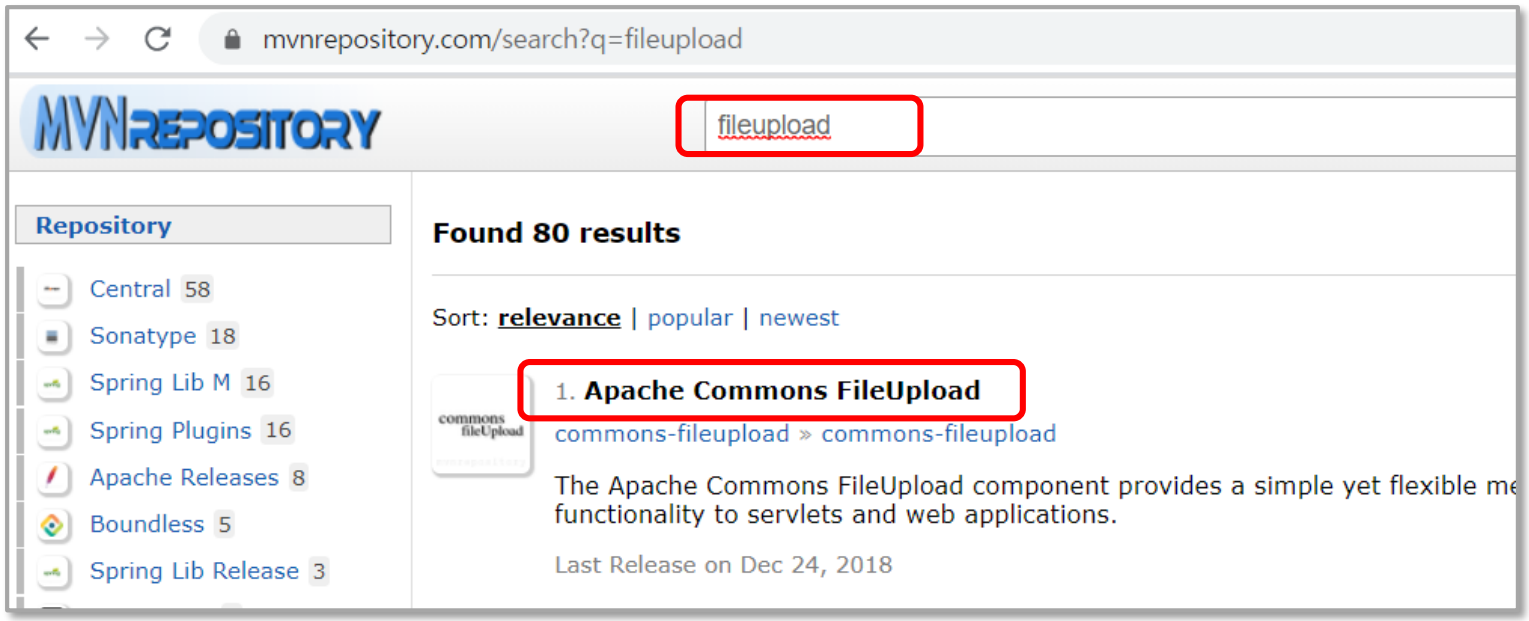
FileUpload

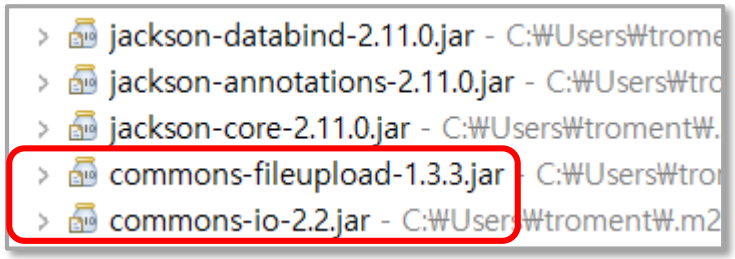
삼성 청년 SW 아카데미

Framework - FileUpload

✓ File Upload.

- pom.xml : commons-fileupload library 추가.





library download 확인

```
<!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.3</version>
</dependency>
```

Framework - FileUpload

- ✓ **File Upload**.
  - servlet-context.xml
  - property
    - maxUploadSize : 최대 업로드 가능한 파일의 바이트크기.
    - maxInMemorySize : 디스크에 임시 파일을 생성하기 전에 메모리에 보관할 수 있는 최대 바이트 크기.

```
<beans:bean id="multipartResolver"
                class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <beans:property name="defaultEncoding" value="UTF-8"/>
        <beans:property name="maxUploadSize" value="52428800"/> <!-- 50MB -->
        <beans:property name="maxInMemorySize" value="1048576"/> <!-- 1MB -->
</beans:bean>
```

Framework - FileUpload

✓ File Upload.

- write.jsp : form 설정

```
<form id="writeform" method="post" enctype="multipart/form-data" action="">
   <div class="form-group" align="left">
      <label for="subject">제목:</label>
      <input type="text" class="form-control" id="subject" name="subject">
   </div>
   <div class="form-group" align="left">
      <label for="content">내용:</label>
      <textarea class="form-control" rows="15" id="content" name="content"></textarea>
   </div>
   <div class="form-group" align="left">
      <label for="subject">파일:</label>
      <input type="file" class="form-control" name="upfile" multiple="multiple">
   </div>
   <button type="button" id="writeBtn" class="btn btn-primary">글작성</button>
   <button type="reset" class="btn btn-warning">초기화</button>
</form>
```

Framework - FileUpload

✓ File Upload.

- GuestBookController.java

```
public class GuestBookDto {

   private int articleno;
   private String userid;
   private String subject;
   private String content;
   private String regtime;
   private List<FileInfoDto> fileInfos;
```

- FileInfoDto.java

```
public class FileInfoDto {

   private String saveFolder;
   private String originFile;
   private String saveFile;
```

Framework - FileUpload

✓ File Upload.

- GuestBookController.java

```
@RequestMapping(value = "/write", method = RequestMethod.POST)
public String write(GuestBookDto guestBookDto, @RequestParam("upfile") MultipartFile[] files, … {
…
  String realPath = servletContext.getRealPath("/upload");
  String today = new SimpleDateFormat("yyMMdd").format(new Date());
  String saveFolder = realPath + File.separator + today;
  File folder = new File(saveFolder);
  if(!folder.exists())
    folder.mkdirs();
  List<FileInfoDto> fileInfos = new ArrayList<FileInfoDto>();
  for (MultipartFile mfile : files) {
    FileInfoDto fileInfoDto = new FileInfoDto();
    String originalFileName = mfile.getOriginalFilename();
    if (!originalFileName.isEmpty()) {
      String saveFileName = UUID.randomUUID().toString() +
                                    originalFileName.substring(originalFileName.lastIndexOf('.'));
      fileInfoDto.setSaveFolder(today);
      fileInfoDto.setOriginFile(originalFileName);
      fileInfoDto.setSaveFile(saveFileName);
```

Framework - FileUpload

✓ File Upload.

- GuestBookController.java

```
        System.out.println(mfile.getOriginalFilename() + "    " + saveFileName);
        mfile.transferTo(new File(folder, saveFileName));
      }
      fileInfos.add(fileInfoDto);
    }
    guestBookDto.setFileInfos(fileInfos);
    guestBookDto.setUserid(memberDto.getUserid());
    try {
      guestBookService.writeArticle(guestBookDto);
      return "guestbook/writesuccess";
    } catch (Exception e) {
      e.printStackTrace();
      model.addAttribute("msg", "글작성 중 문제가 발생했습니다.");
      return "error/error";
    }
  } else {
    model.addAttribute("msg", "로그인 후 사용 가능한 페이지입니다.");
    return "error/error";
  }
```

Framework - FileUpload

✓ File Upload.

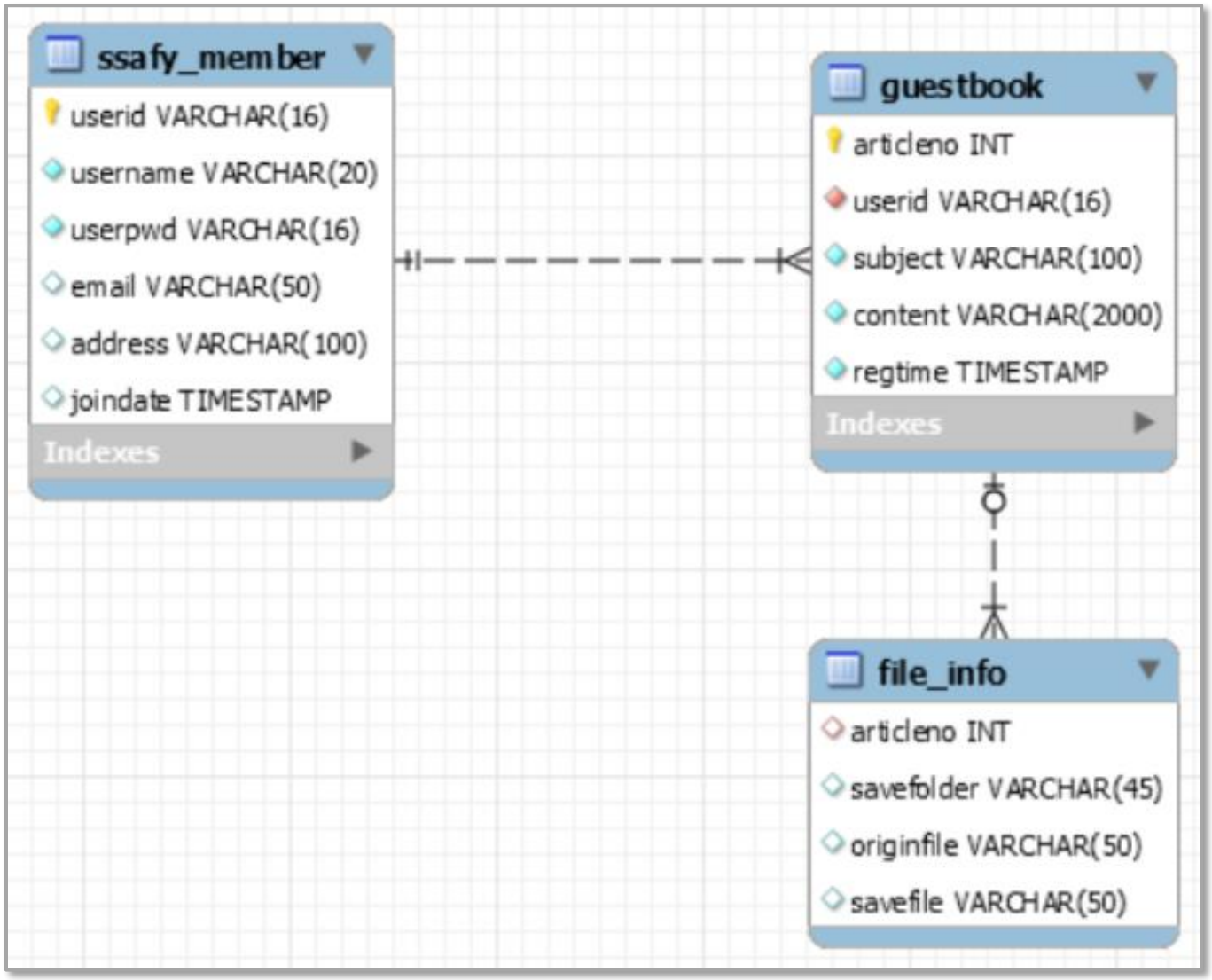
- GuestBookServiceImpl.java

```
@Override
@Transactional
public void writeArticle(GuestBookDto guestBookDto) throws Exception {
  if(guestBookDto.getSubject() == null || guestBookDto.getContent() == null) {
    throw new Exception();
  }
  GuestBookMapper guestBookMapper = sqlSession.getMapper(GuestBookMapper.class);
  guestBookMapper.writeArticle(guestBookDto);
  System.out.println(">>>>>>>>>>>>>>>>>> " + guestBookDto.getArticleno());
  guestBookMapper.fileRegister(guestBookDto);
}
```

Framework - FileUpload

✓ File Upload.

- Table Schema



Framework - FileUpload

✓ File Upload.

- guestbook.xml

```
<insert id="writeArticle" parameterType="GuestBookDto">
  insert into guestbook (userid, subject, content, regtime)
  values (#{userid}, #{subject}, #{content}, now())
  <selectKey resultType="int" keyProperty="articleno" order="AFTER">
    SELECT LAST_INSERT_ID()
  </selectKey>
</insert>

<insert id="fileRegister" parameterType="GuestBookDto">
  insert into file_info (articleno, savefolder, originfile, savefile)
  values
  <foreach collection="fileInfos" item="fileinfo" separator=" , ">
    (#{articleno}, #{fileinfo.saveFolder}, #{fileinfo.originFile}, #{fileinfo.saveFile})
  </foreach>
</insert>
```

Framework - FileUpload

- ✓ File Upload.
  - guestbook.xml

```
<resultMap type="GuestBookDto" id="guestBookList">
  <result property="articleno" column="articleno"/>
  <result property="userid" column="userid"/>
  <result property="subject" column="subject"/>
  <result property="content" column="content"/>
  <result property="regtime" column="regtime"/>
  <collection property="fileInfos" column="articleno" javaType="list"
                                   ofType="FileInfoDto" select="fileInfoList"/>
</resultMap>
```

Framework - FileUpload

✓ File Upload.

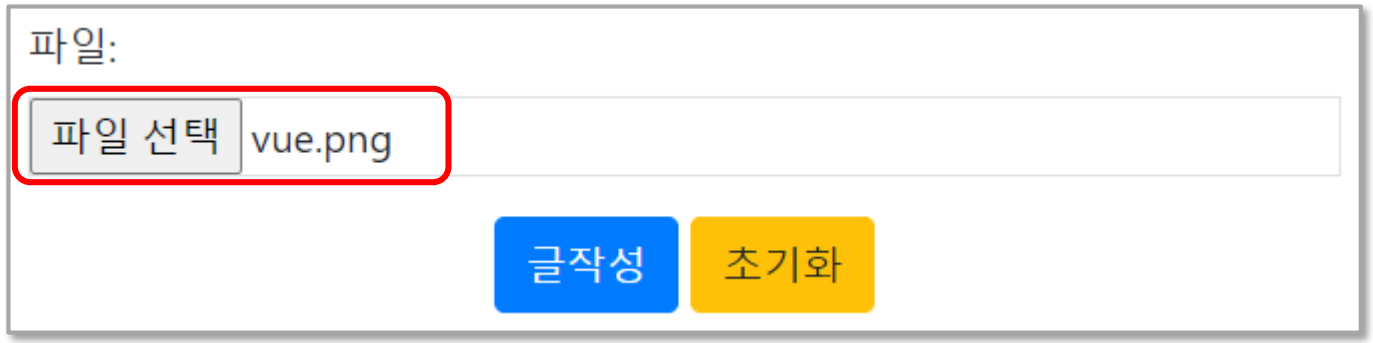
- guestbook.xml

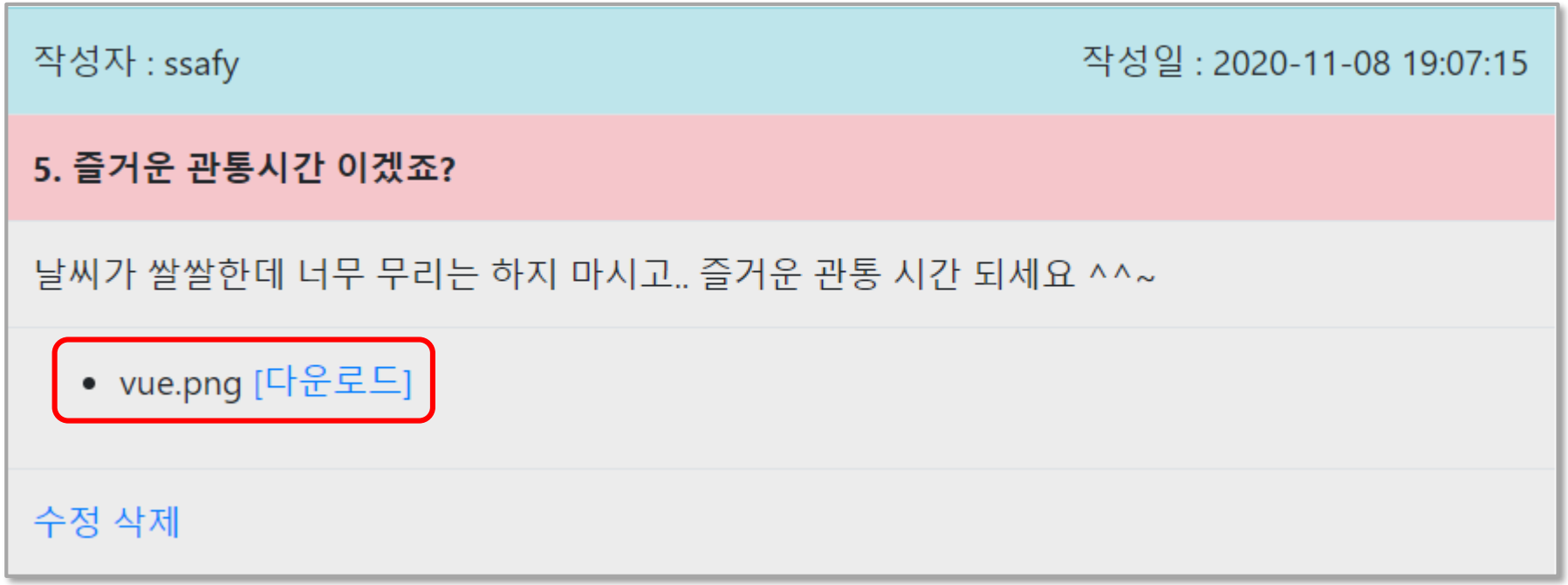
```
<select id="listArticle" parameterType="map" resultMap="guestBookList">
  select articleno, userid, subject, content, regtime
  from guestbook
  <if test="word != null and word != ‘’”>
    <if test="key == 'subject’”>
      where subject like concat('%', #{word}, '%’)
    </if>
    <if test="key != 'subject’”>
      where ${key} = #{word}
    </if>
  </if>
  order by articleno desc
  limit #{start}, #{spp}
</select>
<select id="fileInfoList" resultType="FileInfoDto">
  select savefolder, originfile, savefile
  from file_info
  where articleno = #{articleno}
</select>
```

Framework - FileUpload

✓ File Upload.

- 실행 화면.

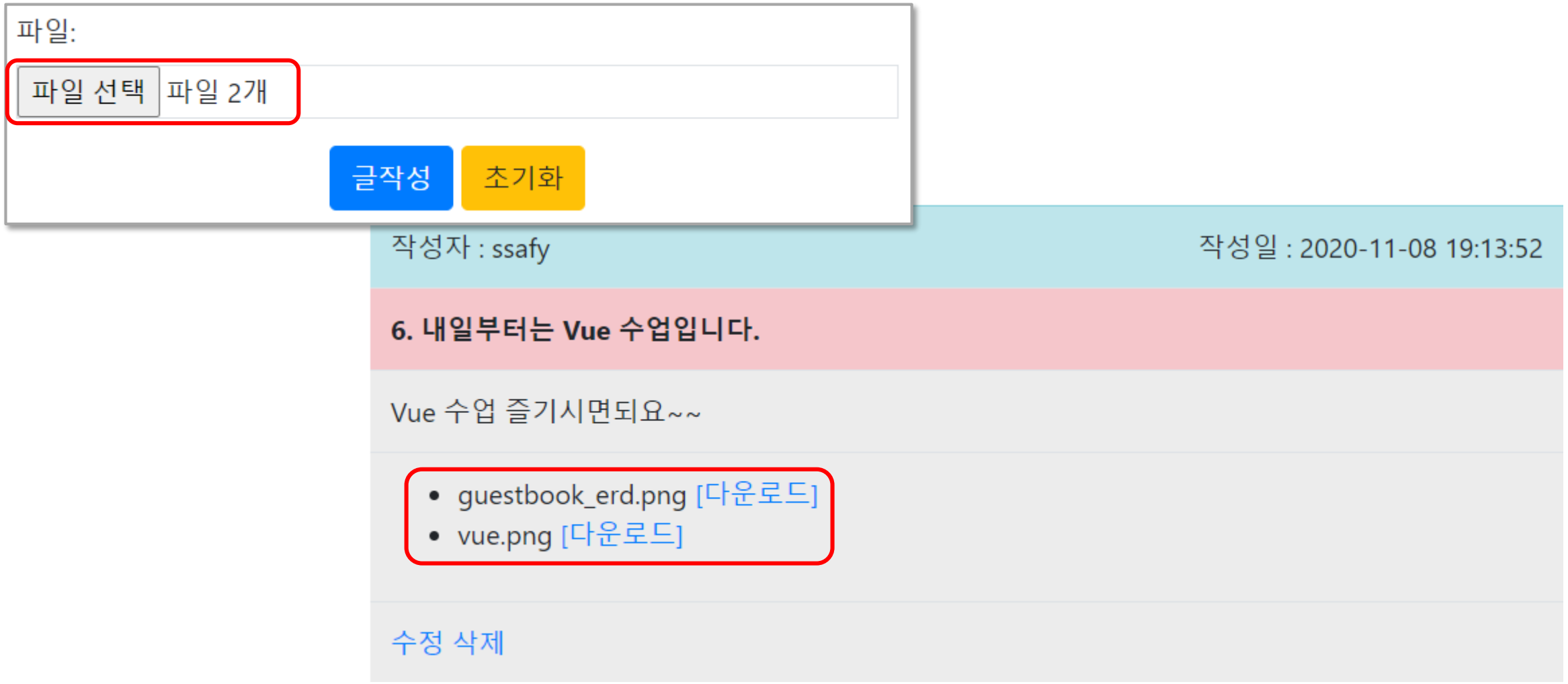




Framework - FileUpload

✓ File Upload.

- 실행 화면 (다중 업로드).



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

FileDownload

삼성 청년 SW 아카데미

Framework - FileDownload

✓ File Download

- list.jsp

```
//file download
$('.filedown').click(function() {
  alert("원본 :  " + $(this).attr('ofile') + "      실제 :  " + $(this).attr('sfile'));
  $(document).find('[name="sfolder"]').val($(this).attr('sfolder'));
  $(document).find('[name="ofile"]').val($(this).attr('ofile'));
  $(document).find('[name="sfile"]').val($(this).attr('sfile'));
  $('#downform').attr('action', '${root}/article/download')
      .attr('method', 'get').submit();
});

<ul>
  <c:forEach var="file" items="${article.fileInfos}">
  <li>${file.originFile}
    <a href="#" class="filedown" sfolder="${file.saveFolder}"
              sfile="${file.saveFile}" ofile="${file.originFile}">[다운로드]</a>
  </c:forEach>
</ul>
```

Framework - FileDownload

✓ File Download

- servlet-context.xml

```
<!-- fileDownload -->
<beans:bean
      id="fileDownLoadView“ class="com.ssafy.guestbook.controller.FileDownLoadView"/>

<!-- BeanNameViewResolver 설정 -->
<beans:bean id="fileViewResolver"
     class="org.springframework.web.servlet.view.BeanNameViewResolver">
  <beans:property name="order" value="0" />
</beans:bean>
<!-- //fileDownload -->
```

Framework - FileDownload

✓ **File Download.**

- GuestBookController.java

```
@RequestMapping(value="/download", method=RequestMethod.GET)
public ModelAndView downloadFile(@RequestParam("sfolder") String sfolder,
                                 @RequestParam("ofile") String ofile,
                                 @RequestParam("sfile") String sfile, HttpSession session) {
  MemberDto memberDto = (MemberDto) session.getAttribute("userinfo");
  if(memberDto != null) {
    Map<String, Object> fileInfo = new HashMap<String, Object>();
    fileInfo.put("sfolder", sfolder);
    fileInfo.put("ofile", ofile);
    fileInfo.put("sfile", sfile);

    return new ModelAndView("fileDownLoadView", "downloadFile", fileInfo);
  } else {
    return new ModelAndView("redirect:/");
  }
}
```

Framework - FileDownload

✓ File Download.

- FileDownLoadView.java

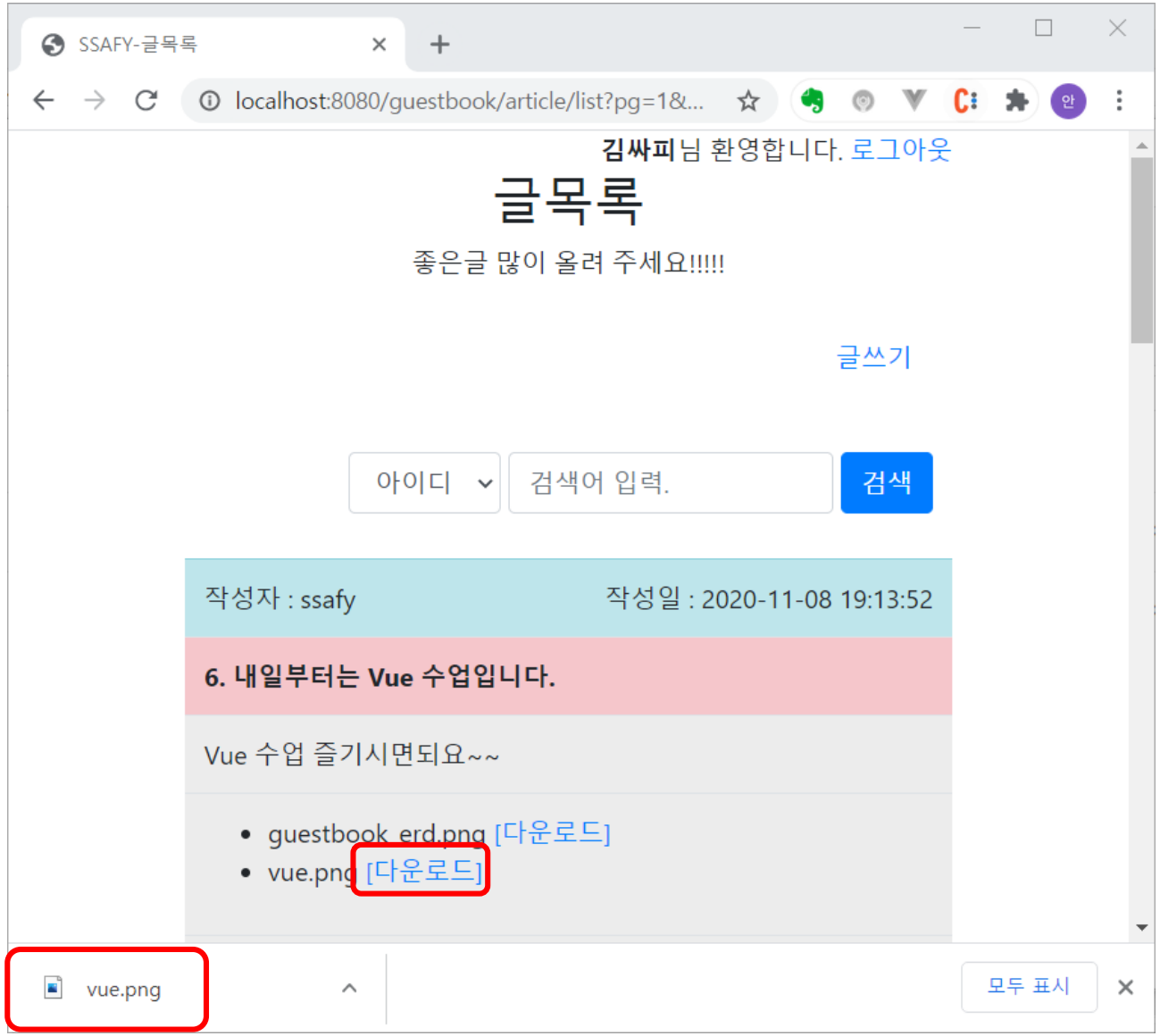
```
response.setContentType(getContentType());
response.setContentLength((int) file.length());

String header = request.getHeader("User-Agent");
boolean isIE = header.indexOf("MSIE") > -1 || header.indexOf("Trident") > -1;
String fileName = null;
// IE는 다르게 처리
if (isIE) {
  fileName = URLEncoder.encode(originalFile, "UTF-8").replaceAll("\\+", "%20");
} else {
  fileName = new String(originalFile.getBytes("UTF-8"), "ISO-8859-1");
}
response.setHeader("Content-Disposition", "attachment; filename=\"" + fileName + "\";");
response.setHeader("Content-Transfer-Encoding", "binary");
OutputStream out = response.getOutputStream();
FileInputStream fis = null;
try {
  fis = new FileInputStream(file);
  FileCopyUtils.copy(fis, out);
```

Framework - FileDownload

✓ File Download.

- 실행 화면.



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Interceptor

삼성 청년 SW 아카데미

✓ HandlerInterceptor를 통한 요청 가로채기.

- Controller가 요청을 처리하기 전/후 처리.
- 로깅, 모니터링 정보 수집, 접근 제어 처리 등의 실제 Business Logic과는 분리되어 처리해야 하는 기능들을 넣고 싶을 때 유용함.
- interceptor를 여러 개 설정 할 수 있음(순서주의!!).

Framework - Interceptor

✓ Interceptor.

- HandlerInterceptor 제공 method.

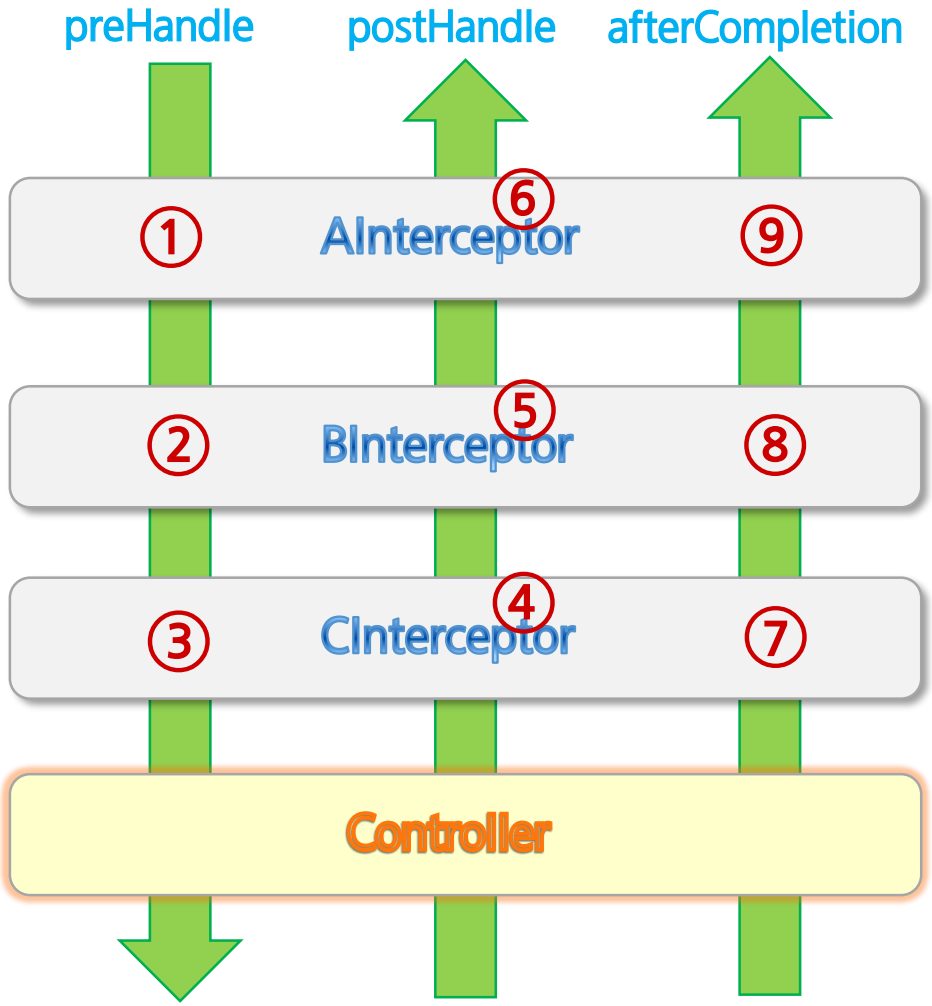
HandlerInterceptor method
boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) • false를 반환하면 request를 바로 종료.
void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) • Controller 수행 후 호출.
void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) • view를 통해 클라이언트에 응답을 전송한 뒤 실행. • 예외가 발생하여도 실행.

✓ HandlerInterceptor를 통한 요청 가로채기.

- Interceptor 호출 순서.

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.AInterceptor"/>
    </mvc:interceptor>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.BInterceptor"/>
    </mvc:interceptor>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.CInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

servlet-context.xml



✓ HandlerInterceptor를 통한 요청 가로채기.

- HandlerInterceptor 인터페이스 구현.
- HandlerInterceptorAdaptor 클래스 제공.

```
public class LoggingInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler) throws Exception {
        System.out.println("intercept!! preHandle");
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {
        System.out.println("intercept!! postHandle");
    }

    @Override
    public void afterCompletion(HttpServletRequest request,
            HttpServletResponse response, Object handler, Exception ex)
            throws Exception {
        System.out.println("intercept!! afterCompletion");
    }
}
```

✓ HandlerInterceptor를 통한 요청 가로채기.

- Interceptor 설정하기. : servlet-context.xml

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.LoggingInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

- 위의 경우 요청 처리시 interceptor의 preHandle, postHandle, afterCompletion 함수의 호출 순서는?
  - Controller method 전/후/응답 완료 후 호출됨을 확인.

```
정보: Server startup in 1754 ms
intercept!! preHandle
intercept!! postHandle
intercept!! afterCompletion
```

✓ HandlerInterceptor를 통한 요청 가로채기.

- 여러 개의 interceptor 등록.
  - EtcInterceptor.java interceptor 추가.

```
public class EtcInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler) throws Exception {
        System.out.println("EtcInterceptor!! preHandle");
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {
        System.out.println("EtcInterceptor!! postHandle");
    }

    @Override
    public void afterCompletion(HttpServletRequest request,
            HttpServletResponse response, Object handler, Exception ex)
            throws Exception {
        System.out.println("EtcInterceptor!! afterCompletion");
    }
}
```

✓ HandlerInterceptor를 통한 요청 가로채기.

- Interceptor 설정하기. : servlet-context.xml

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.LoggingInterceptor"/>
    </mvc:interceptor>
    <mvc:interceptor>
        <mvc:mapping path="/*.html"/>
        <bean class="com.test.hello.EtcInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

- 위의 경우 요청 처리시 interceptor의 preHandle, postHandle, afterCompletion 함수의 호출 순서는?
  - Interceptor 2개 등록 수행 결과(순서 확인!!!).

```
정보: Server startup in 1788 ms
LoggingInterceptor!! preHandle
EtcInterceptor!! preHandle
 >>>>>>>>>>>>>>>>>>  hello method call! <<<<<<<<<<<<<<<<<<
EtcInterceptor!! postHandle
LoggingInterceptor!! postHandle
EtcInterceptor!! afterCompletion
LoggingInterceptor!! afterCompletion
```

Framework - Interceptor

✓ Interceptor - session check.

- servlet-context.xml

```
<beans:bean id="confirm" class="com.ssafy.interceptor.ConfirmInterceptor"/>

<interceptors>
  <interceptor>
    <mapping path="/article/write"/>
    <mapping path="/article/modify"/>
    <mapping path="/article/delete"/>
    <beans:ref bean="confirm"/>
  </interceptor>
</interceptors>
```

Framework - Interceptor

✓ Interceptor - session check.

- ConfirmInterceptor.java

```
public class ConfirmInterceptor extends HandlerInterceptorAdapter {

  @Override
  public boolean preHandle(HttpServletRequest request,
      HttpServletResponse response, Object handler) throws Exception {
    HttpSession session = request.getSession();
    MemberDto memberDto = (MemberDto) session.getAttribute("userinfo");
    if(memberDto == null) {
      response.sendRedirect(request.getContextPath());
      return false;
    }
    return true;
  }
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미