목차

1. build.gradle 에 dependency 추가

- commons-io
- commons-fileupload

2. servlet-context.xml 에 아래 코드 추가

```
<!-- step5. 파일 업로드를 위한 MultipartResolver 설정 -->
<!-- 단위는 byte 로 100,000,000byte 이기 때문에 100MB 로 설정 -->
<beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <beans:property name="maxUploadSize" value="100000000" />
    <beans:property name="maxInMemorySize" value="100000000" />
</beans:bean>
```

3. 파일 업로드

3.1 테이블 생성

```
CREATE TABLE TB_Upload_File (
      uploadFileNo   NUMBER(10)     generated as identity
    , fileName       VARCHAR2(50)   NOT NULL
    , fileSize       NUMBER(10)     NOT NULL
    , contentType    VARCHAR2(30)   NOT NULL

    , PRIMARY KEY(uploadFileNo)
);
```

3.2 모델 클래스 만들기

```
public class ModelUploadFile {

    private Integer uploadFileNo   ;
    private String  fileName       ;
    private Long    fileSize       ;
    private String  contentType    ;

    // getter & setter 만들기

    // 기본 생성자 만들기

    // toString() 만들기

}
```

3.3 mapperUpload.xml 파일 생성(Mybatis mapper)

```
<mapper namespace="mapper.mapperUpload">

    <select id="selectUploadFile" parameterType="ModelUploadFile" resultType="ModelUploadImage">

    </select>

    <insert id="insertUploadFile" parameterType="ModelUploadFile">

    </insert>

    <delete id="deleteUploadFile" parameterType="int">

    </delete>

</mapper>
```

3.4 RepositoryFiles 클래스 작성

- RepositoryFiles 클래스는 한번에 여러개의 파일을 업로드하는 경우에 사용된다.
- MultipartFile 클래스는 commons-fileupload 라이브러리에 들어있다. import 를 하면 사용 할 수 있다.

```
import java.util.List;

import org.springframework.stereotype.Repository;
import org.springframework.web.multipart.MultipartFile;

@Repository
public class RepositoryFiles {

    private List<MultipartFile> files;

    public List<MultipartFile> getFiles() {
        return files;
    }

    public void setFiles(List<MultipartFile> files) {
        this.files = files;
    }

    public RepositoryFiles() {
        super();
    }
}
```

3.5 IDaoUpload 인테이스 , DaoUpload 클래스 만들기

3.6 IServiceUpload , ServiceUpload 클래스 만들기

3.7 fileupload.jsp 작성

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<html>
<head>
    <meta charset="utf-8" />
    <title>file  upload demo</title>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script>
        <!-- jquery 로 파일 첨부 input 태그 추가 -->
        $(document).ready(function() {

            //add more file components if Add is clicked
            $('#addFile').click(function() {

                var fileIndex = $('#fileview tr').children().length;
                $('#fileview').append(
                '<tr><td>' +
                '   <input type="file" name="files['+ fileIndex +']" />' +
                '</td></tr>');
            });
        });
    </script>
</head>
<body>
    <h3>Spring Single File Upload</h3>
    <form:form method="post" action="./fileuploadone" modelAttribute="uploadForm" enctype="multipart/form-data">
```

```
        Upload Directory :
        <input type="text" name="upDir" value="c:/upload/" />
        <br />
        <br />
        <input type="file"  name="file" />
        <br />
        <input type="submit" id="uploadone"   value="Upload One" />
    </form:form>

    <hr>

    <h3>Spring Multi File Upload</h3>
    <form:form method="post" action="./fileuploadmulti" modelAttribute="uploadForm" enctype="multipart/form-data">

        Upload Directory :
        <input type="text" name="upDir" value="c:/upload/" />
        <br />
        <br />
        <input type="button" id="addFile" value="File Add" />

        <table id="fileview">
            <tr>
                <td><input type="file"  name="files[0]" /></td>
            </tr>
        </table>
        <br />
        <input type="submit" id="uploadmulti" value="Upload Multi" />
    </form:form>

    <hr>

    <h3>Drag &amp;amp; Drop Multi File Upload</h3>
    <form:form method="post" action="" modelAttribute="uploadForm" enctype="multipart/form-data">
```

```
    </form:form>
</body>
</html>
```

3.8 uploadsuccess.jsp 작성

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<html>
<head>
    <meta charset="utf-8" />
    <title>File Upload</title>
</head>

<body>
    <p>upload ok!!</p>
    <ol>
        <!--  아래 files 는 컨트롤러에서 직접 넘겨준 모델명 -->
        <c:forEach var="file" items="${files}">
            <li>${file}</li>
        </c:forEach>

        <!-- 컨트롤러에서 @ModelAttribute 로 선언된 객체는 자동으로 view 로 전달  -->
        <br />
        <br />Uload Path : ${uploadForm.upDir}
    </ol>
</body>
</html>
```

3.9 UploadController 컨트롤러 작성

```
import java.io.*;
import java.util.*;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import com.mvc.model.*;
import com.mvc.repository.*;
import com.mvc.service.*;


@Controller
public class UploadController  {

    @Autowired
      private IServiceUpload uploadsrv;


    /**
     * http://localhost/upload/fileupload
     */
    @RequestMapping(value = "/upload/fileupload", method = RequestMethod.GET)
    public String fileupload(Model model)  {

        return "upload/fileupload";
    }

    /**
```

```
     * http://localhost/upload/fileuploadone
     * @throws IOException
     * @throws IllegalStateException
     */
    @RequestMapping(value = "/upload/fileuploadone", method = RequestMethod.POST)
    public String fileuploadone(Model model
            , @ModelAttribute MultipartFile file
            , @RequestParam String upDir
            ) throws IllegalStateException, IOException  {

        // uploadsuccess.jsp 에 출력할 파일이름.
        List<String> filelist = new ArrayList<String>();

        // 폴더 존재 여부 검사
        File dir = new File( upDir );
        if (!dir.exists()) {
            dir.mkdir();
        }

          // 로컬 파일을 서버로 올리기 위한 코드
          String fileName = file.getOriginalFilename();
          String filepath = upDir + "/" + fileName;
          File f = new File( filepath );
          file.transferTo( f );

        // uploadsuccess.jsp 에 출력할 파일이름 저장
        filelist.add(fileName);

        // DB insert 처리를 위한 코드
        ModelUploadFile attachfile = new ModelUploadFile();
        attachfile.setFileName( f.getName() );
        attachfile.setFileSize( (Long)f.length() );
```

```
        attachfile.setContentType( file.getContentType() ); // 확장자

        // DB insert
        uploadsrv.insertAttachFile(attachfile);

        model.addAttribute("files", filelist);

        return "upload/uploadsuccess";
    }
    /**
     * http://localhost/upload/fileuploadmulti
     * @throws IOException
     * @throws IllegalStateException
     */
    @RequestMapping(value = "/upload/fileuploadmulti", method = RequestMethod.POST)
    public String fileuploadmulti( Model model
            , @ModelAttribute RepositoryFiles uploadForm
            , @RequestParam String upDir ) throws IllegalStateException, IOException  {

        logger.info("UploadController.fileuploadmulti");

        List<MultipartFile> files = uploadForm.getFiles();

        // 폴더 존재 여부 검사
        File dir = new File( upDir );
        if (!dir.exists()) {
            dir.mkdir();
        }

        // uploadsuccess.jsp 에 출력할 파일이름.
        List<String> filelist = new ArrayList<String>();

        if( files != null && files.size() > 0 ){
```

```
            for (MultipartFile file : files) {
                // 로컬 파일을 서버로 올리기 위한 코드
                String fileName = file.getOriginalFilename();
                String filepath = upDir + "/" + fileName;
                File f = new File( filepath );
                file.transferTo( f );

                // uploadsuccess.jsp 에 출력할 파일이름 저장
                filelist.add(fileName);

                // DB insert 처리를 위한 코드
                ModelUploadFile attachfile = new ModelUploadFile();
                attachfile.setFileName( f.getName() );
                attachfile.setFileSize( (Long)f.length() );
                attachfile.setContentType( file.getContentType() ); // 확장자

                // DB insert
                uploadsrv.insertAttachFile(attachfile);
            }
        }

        model.addAttribute("files", filelist);

        return "upload/uploadsuccess";
    }
}
```

4. 이미지 파일을 BLOB 형식의 컬럼에 저장하기

이미지를 DB 테이블에 저장하는 방법에는 크게 2 가지가 존재한다.

1. BLOB 컬럼에 바이너리 형태로 저장하는 방식
2. CLOB 컬럼에 문자열 형태로 저장하는 방식

지금부터 이미지 파일을 LOB 형식의 컬럼에 등록을 해보는 방법과 LOB 형식의 데이터를 이미지태그에 출력하는 방법을 해보도록 하겠습니다.

4.1 이미지 저장할 테이블 생성

```
CREATE TABLE TB_Upload_Image (
      uploadImageNo  NUMBER(10)     generated as identity
    , fileName       VARCHAR2(50)   NOT NULL
    , fileSize       NUMBER(10)     NOT NULL
    , contentType    VARCHAR(50)    NOT NULL
    , imageBytes     BLOB  -- 사진 저장 컬럼. 바이너리로 이미지 저장
    , imageBase64    CLOB  -- 사진 저장 컬럼. BASE64 로 이미지 저장

    , PRIMARY KEY(uploadImageNo)
);
```

4.2 모델 클래스 생성

```
public class ModelUploadImage {

    private Integer uploadImageNo   ;
    private String  fileName       ;
    private Long    fileSize       ;
    private String  contentType    ;
    private CommonsMultipartFile image;
    private byte[] imageBytes;
    private String imageBase64;

    // getter & setter 만들기

    // 기본 생성자 만들기

    // toString() 만들기

}
```

4.3 mapperUpload.xml 파일 생성(Mybatis mapper)

```
<mapper namespace="mapper.mapperUpload">


    <select id="getImageByte" parameterType="int" resultType="ModelUploadImage">
        SELECT uploadImageNo, fileName, contentType, fileSize, imageBytes, imageBase64
          FROM TB_Upload_Image
         WHERE uploadImageNo = #{uploadImageNo}
    </select>

    <insert id="insertPhoto" parameterType="map" statementType="CALLABLE">

        declare
            s2  number := 0;
        begin
            INSERT INTO
            TB_Upload_Image(   fileName   ,   fileSize   ,   contentType   ,   imageBytes   ,   imageBase64   )
            VALUES( #{file.fileName}, #{file.fileSize}, #{file.contentType}, #{file.imageBytes}, #{file.imageBase64} )
            RETURNING uploadImageNo INTO s2;

            #{result, jdbcType=INTEGER, mode=OUT} := s2;
        end;
    </insert>

</mapper>
```

4.4 IDaoUpload, DaoUpload 만들기

```
public interface IDaoUpload {
    // 중략

    int insertPhoto(ModelUploadImage attachfile);
    ModelUploadImage getImageByte(int attachfileno);
}
```

```
import java.util.HashMap;
import java.util.Map;

import org.apache.ibatis.session.SqlSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Repository;

import com.mvc.model.*;

@Repository
public class DaoUpload implements IDaoUpload {

    @Autowired
    @Qualifier("sqlSession")
    private SqlSession session;


    public int insertPhoto(ModelUploadImage attachfile) {

        Map<String, Object> map = new  HashMap<String, Object>();
        map.put("file"   , attachfile);
        map.put("result", null);
```

```
        session.insert("mybatis.mapper.mapperUpload.insertPhoto", map);
        int result = map.get("result") != null ? (int) map.get("result") : -1;

        return result;
    }

    public ModelUploadImage getImageByte(int attachfileno) {
        return session.selectOne("mybatis.mapper.mapperUpload.getImageByte", attachfileno);
    }
}
```

4.5 IServiceUpload, ServiceUpload 만들기

```
import com.mvc.model.*;

public interface IServiceUpload {
    int insertAttachFile(ModelUploadFile attachfile);

    int insertPhoto(ModelUploadImage attachfile);
    ModelUploadImage getImageByte(int attachfileno);
}
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.mvc.dao.*;
import com.mvc.model.*;
```

```
@Service("serviceupload")
public class ServiceUpload implements IServiceUpload {

    // SLF4J Logging
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private IDaoUpload uploaddao;

    @Override
    public int insertPhoto(ModelUploadImage attachfile) {

        int result = -1;
        try {
            result = uploaddao.insertPhoto(attachfile);
        } catch (Exception e) {
            logger.error("insertAttachFile " + e.getMessage() );
        }

        return result;
    }

    @Override
    public ModelUploadImage getImageByte(int attachfileno) {
        ModelUploadImage result = null;
        try {
            result = uploaddao.getImageByte(attachfileno);
        } catch (Exception e) {
            logger.error("insertAttachFile " + e.getMessage() );
        }

        return result;
    }
```

```
}
```

4.6 테스트 코드

```
import static org.junit.Assert.*;

import javax.imageio.ImageIO;

import org.apache.commons.io.FilenameUtils;
import org.junit.BeforeClass;
import org.junit.FixMethodOrder;
import org.junit.Test;
import org.junit.runners.MethodSorters;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.awt.image.BufferedImage;
import java.io.*;
import java.nio.file.Files;
import java.util.Base64;

import com.mvc.model.*;
import com.mvc.service.*;

@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class TestServiceUpload {

    private static ApplicationContext context = null;
    private static IServiceUpload service = null;
```

```
    @BeforeClass
    public static void setUpBeforeClass() throws Exception {

        context= new ClassPathXmlApplicationContext("file:src/main/webapp/WEB-INF/spring/appServlet/servlet-
context.xml");
        service=context.getBean(IServiceUpload.class);
    }

    @Test
    public void test_insertPhoto() {

        // convert image to byte

        try {
            File file = new File("src/test/resources/image.png");
            BufferedImage originalImage = ImageIO.read(file.getAbsoluteFile());

            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ImageIO.write(originalImage, "jpg", baos);
            baos.flush();
            byte[] photoBytes = baos.toByteArray();

            ModelUploadImage fileupload = new ModelUploadImage();
            fileupload.setFileName( file.getName() );
            fileupload.setFileSize( file.length() );
            fileupload.setContentType( Files.probeContentType( file.toPath() ) );
            fileupload.setImageBytes(photoBytes);
            fileupload.setImageBase64( Base64.getEncoder().encodeToString(baos.toByteArray()) );

            int result = service.insertPhoto(fileupload);

            assertNotSame(-1, result);

        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }

    @Test
    public void test_getImageByte() {

        // convert byte to image

        try {

            int attachfileno = 1;
            ModelUploadImage result = service.getImageByte(attachfileno);

            // convert byte array back to BufferedImage
            InputStream in = new ByteArrayInputStream( result.getImageBytes() );
            BufferedImage bImageFromConvert = ImageIO.read(in);

            File file = new File( "c:\\" + result.getFileName() );

            File dir = new File(file.getAbsolutePath());
            if (!dir.exists()) {
                dir.mkdir();
            }

            ImageIO.write(bImageFromConvert, FilenameUtils.getExtension(file.getName()), file );

            assertSame(1, result.getUploadImageNo() );

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

4.7 DB 에 파일을 BLOB 로 등록하기 위한 Contoller

```
import java.io.*;
import java.util.*;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import com.mvc.model.*;
import com.mvc.repository.*;
import com.mvc.service.*;

@Controller
public class UploadController  {

    private static final Logger logger = LoggerFactory.getLogger(UploadController.class);

    @Autowired
      private IServiceUpload uploadsrv;

    /**
     * 사진 업로드를 위한 화면
     * @return
```

```
     */
    @RequestMapping(value="/upload/imageupload", method = RequestMethod.GET)
    public String saveImage() {
        return "upload/imageupload";
    }

    /**
     * 사진 파일 업로드 후 DB 저장
     * @param vo
     * @return
     */
    @RequestMapping(value="/upload/imageupload", method = RequestMethod.POST)
    public String saveImage(Model model
            , @RequestParam String upDir
            , @ModelAttribute ModelUploadImage vo ) {

        Integer attachfileno = null;

        try {
            vo.setFileName( vo.getImage().getOriginalFilename() );
            vo.setFileSize( (Long)vo.getImage().getSize() );
            vo.setContentType( vo.getImage().getContentType() ); // 확장자
            vo.setImageBytes( vo.getImage().getBytes() );
            vo.setImageBase64( Base64.getEncoder().encodeToString( vo.getImage().getBytes() ) );

            attachfileno = uploadsrv.insertPhoto(vo);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return "redirect:/upload/imageview/"+ Integer.toString( attachfileno );
    }
    /**
     * 이미지 뷰페이지
```

```
     * @return
     */
    @RequestMapping(value="/upload/imageview/{attachfileno}", method = RequestMethod.GET)
    public String imageview(Model model, @PathVariable int attachfileno) {

        ModelUploadImage result = uploadsrv.getImageByte(attachfileno);

        model.addAttribute("attachfileno", attachfileno);
        model.addAttribute("contentType", result.getContentType() );
        model.addAttribute("imageBase64", result.getImageBase64() );

        return "upload/imageview";
    }

    /**
     * img 태그의 src 에 이미지를 출력하기 위한 메서드
     * @return
     */
    @RequestMapping(value="/upload/getphoto/{attachfileno}", method = RequestMethod.GET)
    public ResponseEntity<byte[]> getImageByte(@PathVariable int attachfileno) {

        ModelUploadImage result = uploadsrv.getImageByte(attachfileno);

       byte[] imageContent =  result.getImageBytes();
       final HttpHeaders headers = new HttpHeaders();
       headers.setContentType( MediaType.valueOf( result.getContentType() ) );

       return new ResponseEntity<byte[]>(imageContent, headers, HttpStatus.OK);
    }
}
```

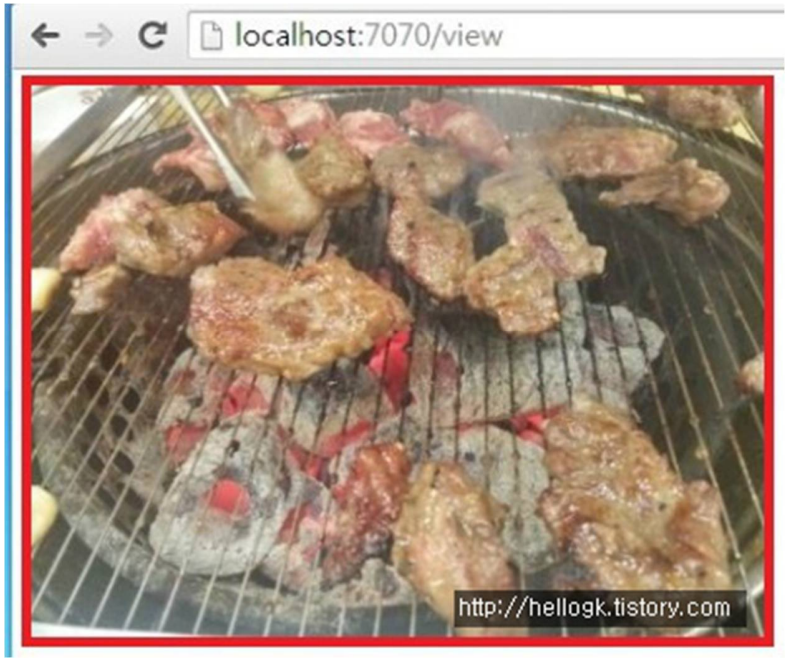
4.8 DB 에 저장할 파일처리를 위한 뷰페이지(imageupload.jsp)

```jsp
<%@ page session="false" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<html>
<head>
    <meta charset="utf-8" />
      <title> 테이블 컬럼에 이미지 저장하는 예제 </title>
</head>
<body>
    <form action="/upload/imageupload" enctype="multipart/form-data" method="post">

        Upload Directory :
        <input type="text" name="upDir" value="c:/upload/" />
        <br />
        <br />
        <input type="file" name="image" />
        <br />
        <input type="submit" value="이미지저장"/>
    </form>
</body>
</html>
```

4.9 뷰페이지(imageview.jsp)

```
<%@ page session="false" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<head>
    <meta charset="utf-8" />
      <title>helloworld</title>
</head>
<body>
    <!-- img 태그의 src 경로는 byte 이미지 가져오는 컨트롤러 호출(/getImageByte) -->
    <img alt="" src="/upload/getphoto/${attachfileno}" />
    <hr />
    <img src="data:${contentType};base64,${imageBase64}" />
</body>
</html>
```

임의로 body 태그내에 작성 후 /view 컨트롤러를 호출해보도록 하겠습니다.



호출결과 정상적으로 byte 데이터를 DB 로부터 가져와서 화면에 출력이 되었습니다.

5. Reference

http://aircook.tistory.com/entry/Spring-Ibatis-프레임워크-구성시-오라클-LOB-타입-사용하기

https://anirbanchowdhury.wordpress.com/2012/05/15/mybatis-upload-retrieve-download-file-or-image/
http://www.databasesql.info/article/534369750/

http://hellogk.tistory.com/129

http://stackoverflow.com/questions/12059872/how-to-select-a-blob-column-from-database-using-ibatis
http://stackoverflow.com/questions/17055558/select-a-blob-column-from-oracle-db-using-mybatis

http://viralpatel.net/blogs/tutorial-save-get-blob-object-spring-3-mvc-hibernate/

http://javahonk.com/save-and-retrieve-image-from-database/

http://forum.spring.io/forum/spring-projects/web/52397-multipart-fileupload-problem-in-fileuploadaction-junit-testing

http://stackoverflow.com/questions/8799378/how-to-write-unit-test-for-commonsmultipartfile-with-mock-in-spring

http://stackoverflow.com/questions/7879620/how-to-unit-test-file-uploads-with-mockhttpservletrequest

convert byte array to base64 string in java

http://stackoverflow.com/questions/2418485/how-do-i-convert-a-byte-array-to-base64-in-java