■ Groups 테이블과 Idols 테이블 변경

1) Idols테이블

Column Name	Data Type	Nullable	Default	Primary Key
NO	NUMBER(9,0)	No	-	1
NAME	VARCHAR2(100)	No	-	-
IMAGE	VARCHAR2(200)	No	-	-
BIRTH_DATE	DATE	No	-	-
GROUP_NO	NUMBER(9,0)	No	-	-
				1 - 5

- image는 파일업로드할 사진

2) Groups 테이블

Column Name	Data Type	Nullable	Default	Primary Key
NO	NUMBER(9,0)	No	-	1
NAME	VARCHAR2(100)	No	-	-
DEBUT_DATE	DATE	No	-	-
MEMBER_NUM	NUMBER(3,0)	No	-	-
				1 - 4

- member_num은 멤버수

■ 로그인인터셉터의 구현

1) 로그인이 되어 있지 않을 경우 insertGroup.html / updateGroup.html / deleteGroup.html에는 접근이 불가능해야 함

2) interceptor로 처리

3) org.imw.ims.interceptor.LoginCheckInterceptor 생성

```
package org.imw.ims.interceptor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import org.imw.ims.vo.Member;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;


public class LoginCheckInterceptor extends HandlerInterceptorAdapter{
	//전처리
		//컨트롤러의 해당메서드가 작동되기 전
		@Override
		public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler)
					throws Exception {

			//리턴값이 true면 그대로 작동
			//리턴값이 false면 작동안함

			//로그인했는지 확인해서 로그인 안되어있으면
			//index.html로 이동

			//System.out.println("LoginCheckInterceptor 전처리");

			//세션얻어옴
			HttpSession session = request.getSession();

			Member loginMember =
					(Member)session.getAttribute("loginMember");

			if(loginMember==null) {
				//로그인이 안되어 있음
				System.out.println("로그인안되어있음");
				response.sendRedirect("/index.html");

				return false;

			}

			System.out.println("LoginCheckInterceptor:"+loginMember);

			//로그인이 되어있음(아무 할일이 없음)
			return true;
		}

		//후처리
		//컨트롤러의 메서드가 리턴할때(끝나고 난후)
		@Override
		public void postHandle(HttpServletRequest request, HttpServletResponse response,
```

```
Object handler,
                                ModelAndView modelAndView) throws Exception {
                        // TODO Auto-generated method stub

                        System.out.println("후처리");

                }


}
```

```
<mvc:interceptors>
        <mvc:interceptor>
                <mvc:mapping path="/deleteGroup.html"/>
                <mvc:mapping path="/insertGroup.html"/>
                <mvc:mapping path="/updateGroup.html"/>
                <bean class="org.imw.ims.interceptor.LoginCheckInterceptor"/>
        </mvc:interceptor>
</mvc:interceptors>
```

■ 인터셉터란?

1) intercept는 '가로채다'라는 뜻으로 농구에서 상대가 가진 볼을 빼앗을때 '인터셉트'했다고 표현

2) Spring MVC에서는 요청(예:/update.html)에 대해서 Controller가 작동되기 전에 그 요청을 가로채서 처리하고 컨트롤러로 넘기거나 응답을 할 수 있음

3) 로그인이 필요한 요청에 대한 전처리 가능

4) Spring MVC의 요청 처리 순서

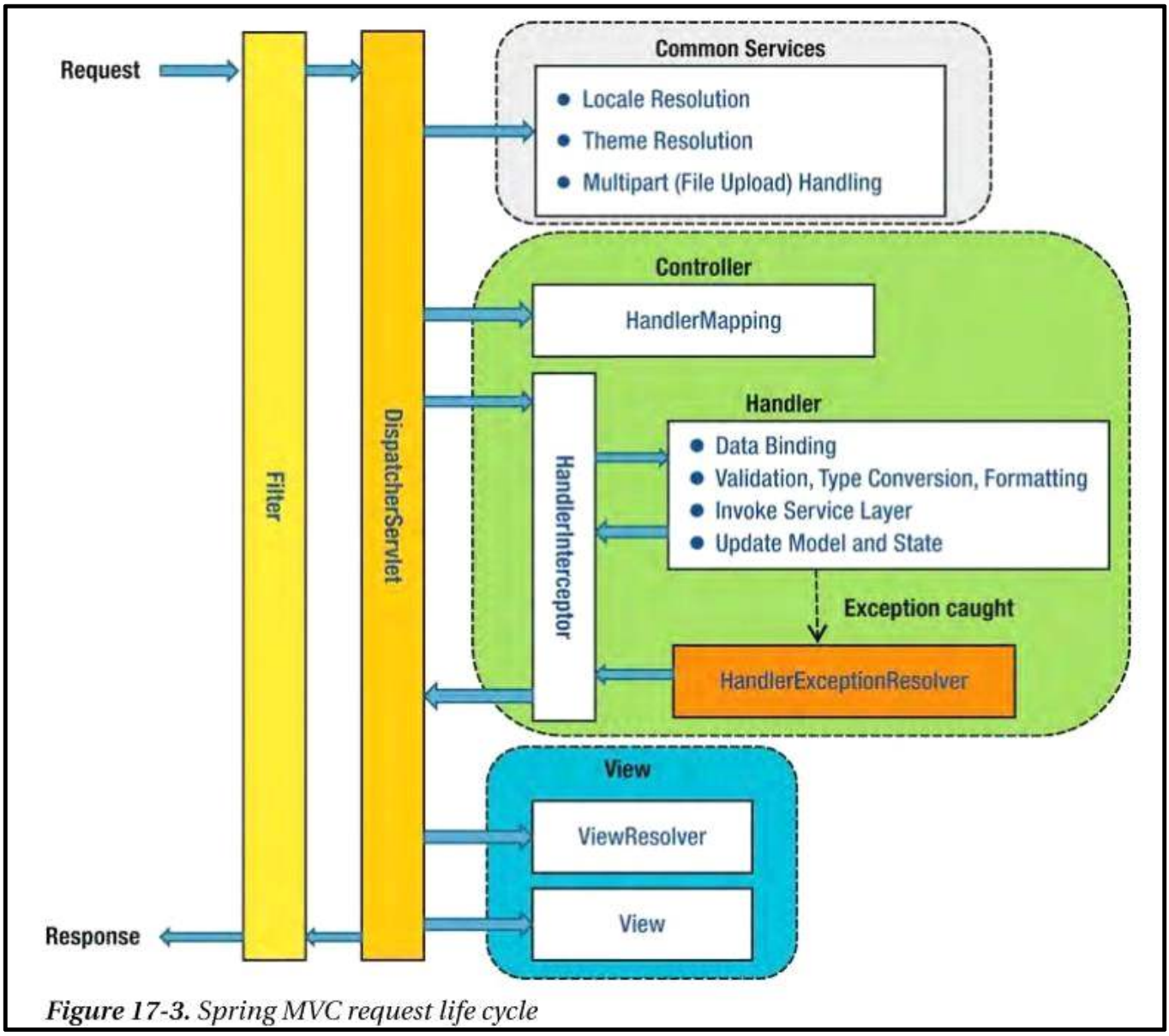
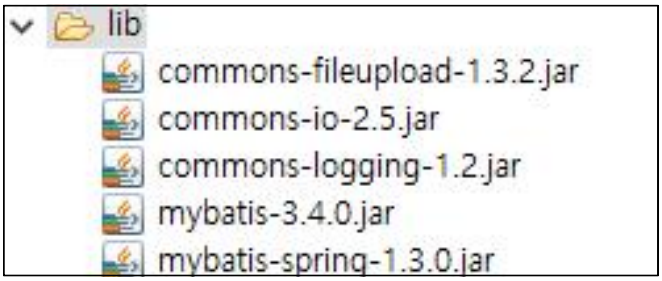


Figure 17-3. Spring MVC request life cycle

■ 아이돌 insert할때 파일 업로드 구현

1) common-io / common-fileupload 가져다 놓기



2) Multipart처리를 위한 리졸버 설정

```
<!-- MultipartResovler -->
	<bean id="multipartResolver"
	p:defaultEncoding="UTF-8"
	p:maxUploadSize="104857600"
	class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/>
```

3) Test를 위한 FileUploadController 생성

```
package org.imw.ims.controller;

@Controller
public class FileUploadController {

	@RequestMapping(value="/upload.html",method=RequestMethod.GET)
	public void form() {

	}

	@RequestMapping(value="/upload.html",method=RequestMethod.POST)
	public String upload(MultipartFile image,Idol idol ,HttpServletRequest request,
RedirectAttributes ra) {

		//1) ServletContext얻기
		ServletContext sc = request.getServletContext();

		//2) 기본경로 얻기
		String path = sc.getRealPath("");
```

```
                //3) upload경로
                String uploadPath = path+"upload"+File.separator;

                //4) 200x200로 작게 만든 image경로
                String profilePath = path+"profile"+File.separator;

                //5) 고유한 값을 위한 UUID
                UUID uuid = UUID.randomUUID();
                String fileName = uuid+image.getOriginalFilename();

                //6) 경로+파일이름
                String fullPath = uploadPath+fileName;

                //7) 실제 생성될 파일
                File file = new File(fullPath);

                try {
                        //8) 파일 옮기기
                        image.transferTo(file);
                        System.out.println("파일 생성 성공");
                        //9) 리사이징 (200x200)
                        ResizeImageUtil.resize(fullPath, profilePath+fileName, 200);

                } catch (Exception e) {
                        System.out.println("파일 옮기기 실패");
                }

                //업로드된 사진의 이름
                ra.addFlashAttribute("src",fileName);

                return "redirect:upload.html";
        }
}
```

4) test를 위한 upload.jsp생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html lang="ko">
<head>
<meta charset="UTF-8">
<title>업로드 테스트</title>
```

```
<link rel="stylesheet" href="/css/font-awesome.min.css" />
<style>
#selectImage {
	width: 200px;
	height: 200px;
	display: block;
	border: 1px solid #424242;
	cursor: pointer;
	text-align: center;
	position: relative;
	margin: 10px 0;
}

.fa-camera {
	position: absolute;
	left: 0;
	top: 0;
	width: 200px;
	height: 200px;
	display: block;
	font-size: 150px;
	line-height: 200px;
}

#image {
	width: 200px;
	height: 200px;
	position: absolute;
	opacity: 0;
	left: 0;
	top: 0;
	cursor: pointer;
}

#imageCanvas {
	position: absolute;
	left: 0;
	top: 0;
	background: #fff;
	display: none;
}

#selectImage.selected #imageCanvas {
	display: block;
}

#selectImage.selected #deleteBtn {
```

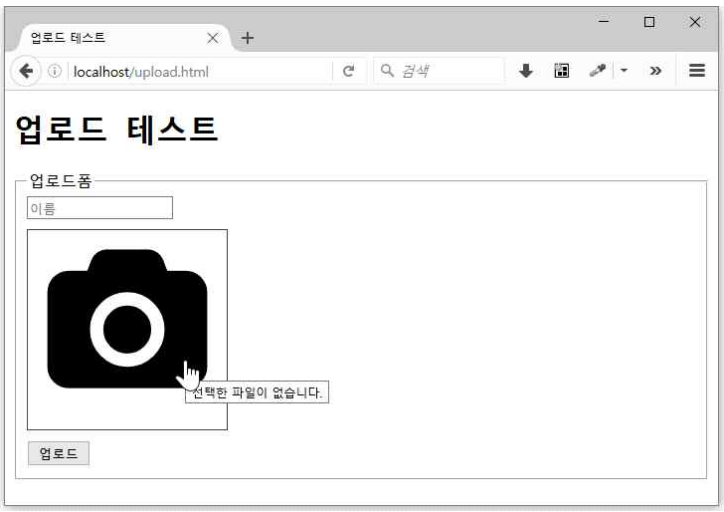
```
        z-index: 2;
        display: block;
}

#selectImage.selected #loader {
        display: none;
}

#deleteBtn {
        position: absolute;
        right: 0;
        top: 0;
        width: 30px;
        height: 30px;
        border-radius: 15px;
        padding: 0;
        border: none;
        font-size: 30px;
        background: transparent;
        display: none;
        cursor: pointer;
}

#loader {
        width: 200px;
        height: 200px;
        background: url(img/loader.gif);
        background-repeat: no-repeat;
        background-position: center;
        position: absolute;
        left: 0;
        top: 0;
        background-color: #fff;
        z-index: 3;
        cursor: default;
        display: none;
}
</style>
</head>
<body>
<h1>업로드 테스트</h1>
<form action="upload.html" method="post" enctype="multipart/form-data">
        <fieldset>
                <legend>업로드폼</legend>
                <input type="text" name="name" placeholder="이름" />
                <div id="selectImage" class="">
                        <button type="button" id="deleteBtn">
```

```
                                <i class="fa fa-times-circle"></i>
                        </button>
                        <i class="fa fa-camera"></i>
                        <canvas id="imageCanvas" width="200" height="200"></canvas>
                <input type="file" name="image" id="image">
                <div id="loader"></div>
        </div>
        <button>업로드</button>
</fieldset>
</form>
<c:if test="${src!=null }">
        <h2>프로필이미지</h2>
        <img src="profile/${src }" />
        <h2>실제이미지</h2>
        <img src="upload/${src }" />
</c:if>
<script src="/js/jquery.js"></script>
<script>
        var $image = $("#image"),
                $canvas = $("#imageCanvas"),
                canvas = $canvas.get(0),
                ctx = canvas.getContext("2d"),
                $loader = $("#loader"),
                $selectImage = $("#selectImage"),
                $deleteBtn = $("#deleteBtn");

        $deleteBtn.click(function() {
                $selectImage.removeClass("selected");
                ctx.clearRect(0, 0, 200, 200);
                $image.val("");
        });

        $image.change(function() {

                $loader.show();

                var file = this.files[0];

                var reg = /^image\//;

                if (!reg.test(file.type)) {

                        $loader.hide();
                        alert("이미지를 선택해주세요~");

                        $image.val("");
```

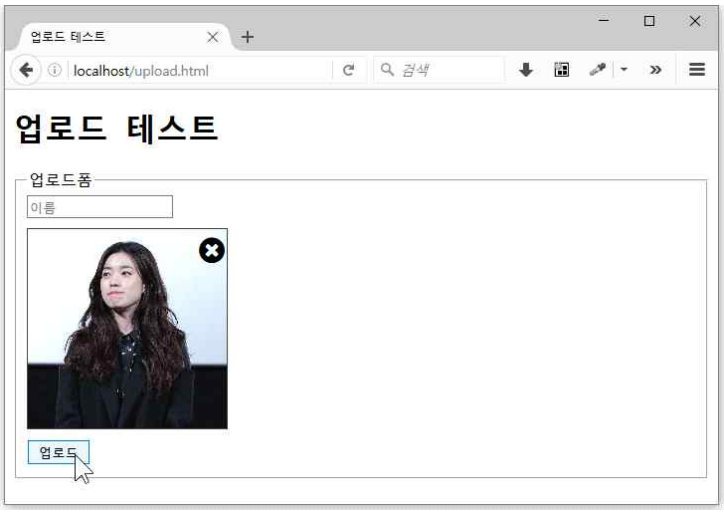
```
                return;
        }

        var reader = new FileReader();

        //파일을 DataURL로 읽어옵니다.
        reader.readAsDataURL(file);

        //다 읽었으면 onload 이벤트 발생
        reader.onload = function() {
                //alert("다 읽었어요!");

                //alert(reader.result);

                //$("img").attr("src",reader.result);

                //이미지 객체 생성후

                img = new Image();
                img.src = reader.result;

                //이미지 로딩후
                img.onload = function() {

                        var x = 0;
                        var y = 0;

                        var width = img.width;
                        var height = img.height;

                        var rate = width / height;

                        // alert(rate);

                        if (rate >= 1) {

                                //alert("가로더김");
                                width = height;
                                x = (img.width - img.height) / 2;

                        } else {
                                // alert("세로더김");
                                height = width;
                                y = (img.height - img.width) / 2;
                        }

                        img.width = 200;
```

```
                                img.height = 200;
                                ctx.clearRect(0, 0, 200, 200);
                                ctx.drawImage(img, x, y, width, height, 0, 0, 200, 200);

                                $loader.hide();

                                $selectImage.addClass("selected");

                        };
                };

        });
</script>
</body>
</html>
```

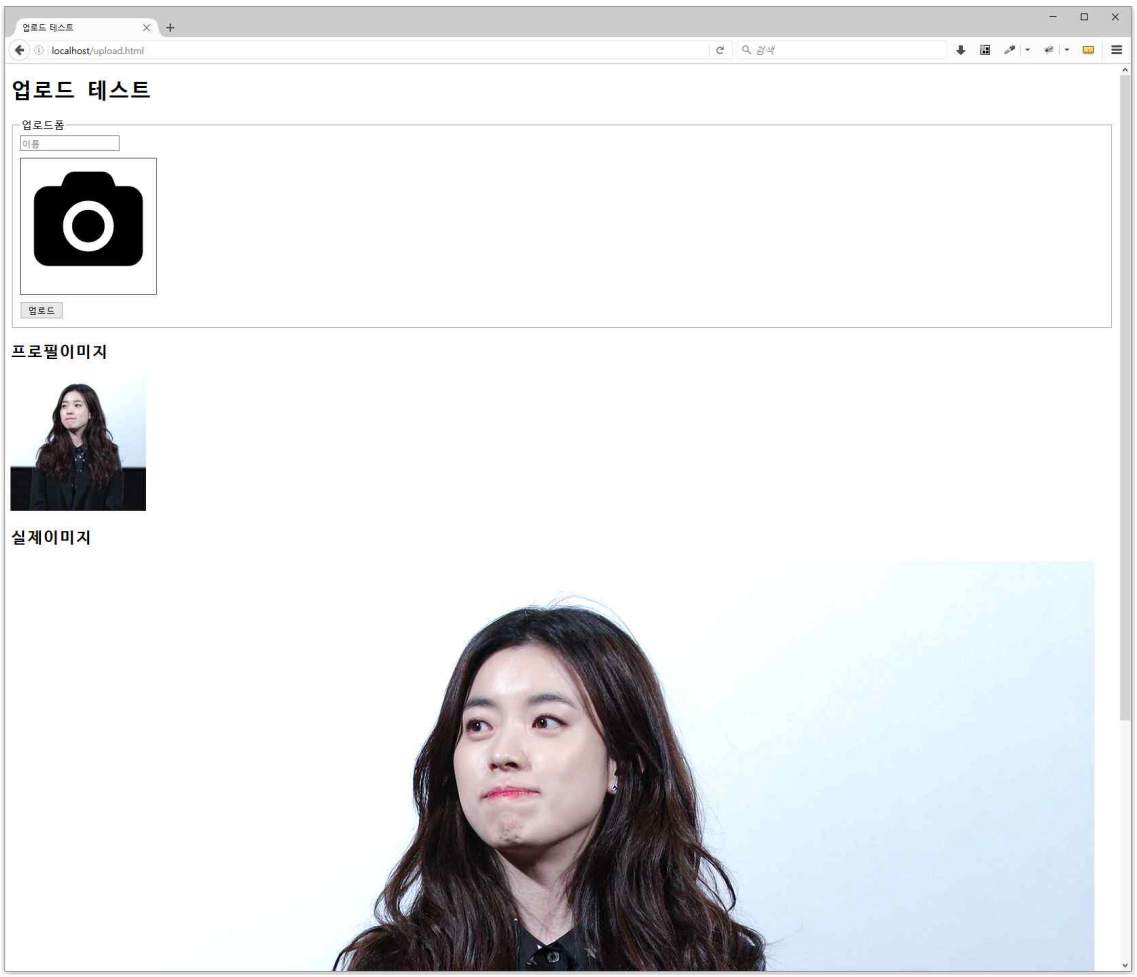
- 파일 선택 전



- 파일 선택 후



- 업로드 후



■ Idol VO / DAO / Service / mapper파일 만들기

1) Idol

```
package org.imw.ims.vo;

import java.sql.Date;

public class Idol {

	private int no,groupNo,year,date,month;
	private String name,image;
	private Date birthDate;



	public int getYear() {
		return year;
```

```
	}

	public void setYear(int year) {
		this.year = year;
	}

	public int getDate() {
		return date;
	}

	public void setDate(int date) {
		this.date = date;
	}

	public int getMonth() {
		return month;
	}

	public void setMonth(int month) {
		this.month = month;
	}

	public Idol() {
		// TODO Auto-generated constructor stub
	}

	public int getGroupNo() {
		return groupNo;
	}



	public void setGroupNo(int groupNo) {
		this.groupNo = groupNo;
	}



	public String getImage() {
		return image;
	}



	public void setImage(String image) {
		this.image = image;
	}



	public Date getBirthDate() {
```

```
                if(birthDate==null) birthDate = Date.valueOf(year+"-"+month+"-"+date);
                return birthDate;
        }


        public void setBirthDate(Date birthDate) {
                this.birthDate = birthDate;
        }


        public int getNo() {
                return no;
        }
        public void setNo(int no) {
                this.no = no;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }


}
```

2) IdolsDAO / IdolsDAOImpl

3) IdolsService / IdolsServiceImpl

4) idols.xml

■ Transaction 적용하기

1) transactionManger추가

```
<!-- 트랜잭션매니져 -->
<bean id="transactionManager"
 p:dataSource-ref="dataSource"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager"/>
```

2) annotaion으로 트랜잭션적용 가능하게

```
<tx:annotation-driven/>
```

3) 필요한 Service에 @Transactional 어노테이션 추가

```
@Transactional
@Override
public boolean add(Idol idol) {

	if(idolsDAO.insert(idol)>0) {

		if(groupsDAO.updateMemberNum(idol.getGroupNo())>0){
			return true;
		}

	}
	return false;
}
```