KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

스프링 MVC의 Controller

[KB] IT's Your Life

KB 국민은행

BL을 통해서 결과가 만들어졌고 뷰로 전달해야하는 상황 ==> scope 활용

scope page request session application

1 Model이라는 데이터 전달자

Controller 메서드의 매개변수로 Model 타입

- 컨트롤러에서 생성된 데이터를 JSP에 전달
  - request 스코프에 속성으로 저장
  - jsp로 forward

- Servlet에서 모델2 방식으로 데이터를 전달하는 방식

```
    request.setAttribute("serverTime", new java.util.Date());

    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-
INF/jsp/home.jsp");

    dispatcher.forward(request, response);
```

1 Model이라는 데이터 전달자

HomeController.java

```
package org.scoula.controller;

import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;


@Controller
@Slf4j
public class HomeController {

    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute("name", "홍길동");

        return "index";                    // View의 이름
    }

}
```

모델 Dependency 요구됨

스코프에 담을 맵이라고 생각.

BL의 결과가 model매개변수를 통해서 전달됨

setAttribute대신에 addAttribute 키 밸류 쌍

1 Model이라는 데이터 전달자

views/index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>${name} 환영합니다.</h1>
</body>
</html>
```

```
@GetMapping("/")
public String home(Model model) {
    model.addAttribute("name", "홍길동");

    return "index";      // View의 이름
}
```

request scope 에 해당됨.

전에 배운 EL 표기

1 Model이라는 데이터 전달자

- **@ModelAttribute**
  - DTO 쿼리 파리미터는 자동으로 뷰로 저달됨
  - 기본 자료형 쿼리 파라미터는 뷰로 전달되지 않음
  - @ModelAttribute("파라미터명")
    - 쿼리 파라미터를 request 스코프에 저장
    - 파라미터명이 스코프의 키가 됨

앞에서 봤던 DTO는 리퀘스트 스코프에 담김.
수동으로 model에 추가할 필요 없음. queryparam으로 가져온 애들은 따로 추가해야함. 할 생각이면

이때 이 어노테이션을 사용

1 Model이라는 데이터 전달자

SampleController.java

캐멀케이스 클래스 이름으로 담김

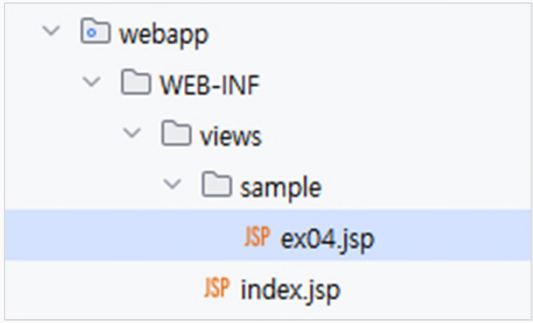
그냥 이름으로 담김

DTO와 DTO가 아닌 것

```
...
public class SampleController {
        ...

        @GetMapping("/ex04")
        public String ex04(SampleDTO dto, int page) {
                log.info("dto: " + dto);
                log.info("page: " + page);

                return "sample/ex04";
        }
}
```

views/sample/ex04.jsp

```
<!DOCTYPE html>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
		<h2>SAMPLE DTO ${sampleDTO}</h2>
		<h2>PAGE ${page}</h2>
</body>
</html>
```



1 Model이라는 데이터 전달자

- **http://localhost:8080/sample/ex04?name=aaa&age=11&page=9**

```
INFO  org.scoula.ex03.controller.SampleController(ex04:84) - dto: SampleDTO(name=aaa, age=11)
INFO  org.scoula.ex03.controller.SampleController(ex04:85) - page: 9
```

```
SAMPLE DTO SampleDTO(name=aaa, age=11)
PAGE
```

DTO는 전달 됐는니

page는 누락됨

→ page는 뷰에 전달되지 않음

어떻게 해야할까?

SampleController.java

```
...
public class SampleController {
        ...

        @GetMapping("/ex04")
        public String ex04(SampleDTO dto, @ModelAttribute("page") int page) {
                log.info("dto: " + dto);
                log.info("page: " + page);

                return "sample/ex04";
        }
}
```

이름이 된다. 저 어노테이션의 값이니까 리퀘스트 스코프의 키값으로 값은 밸류로 저장된다

http://localhost:8080/sample/ex04?name=aaa&age=11&page=9

```
SAMPLE DTO SampleDTO(name=aaa, age=11)

PAGE 9
```

2 Controller의 리턴 타입

- **Controller 메서드의 리턴 타입** 메서드의 리턴의 의미.

뷰의 이름으로 해석

- String
  - jsp 뷰의 경로/이름으로 해석
- void 컨텍스트 경로는 빼고
  - 호출한 URL과 동일 이름의 jsp로 해석

젤 많이 쓰이는 2가지

rest통신 할때많이 쓰임 결과가 JSON응답

- VO, DTO 타입
  - JSON 타입의 데이터로 변환해서 브라우저로 응답
- ResponseEntity 타입
  - Http 헤더 정보와 내용을 가공하여 직접 브라우저로 응답

특수한경우

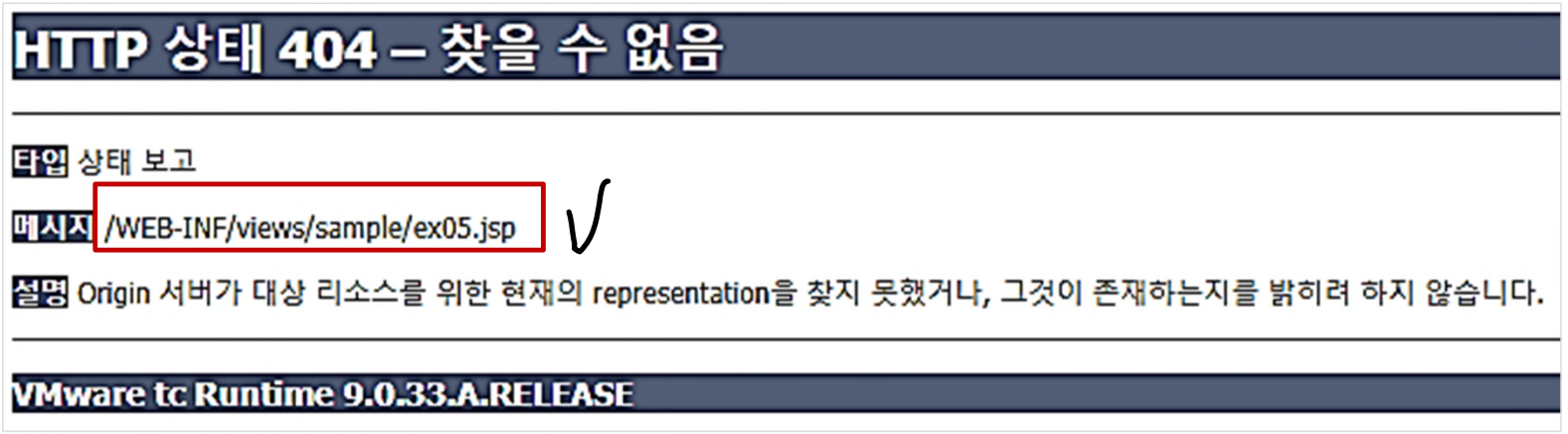
- Model, ModelAndView 모델 데이터와 뷰 정보를 같이 리턴하는 특수한 경우.
  - Model로 데이터를 변환하거나 뷰이름을 같이 지정
- HttpHeaders
  - 응답에 내용 없이 Http 헤더 메시지만 전달

2 Controller의 리턴 타입

void 타입

```
@GetMapping("/ex05")
public void ex05() {
  log.info("/ex05.....");
}
```

url이 ㅇ뷰의 이름이 됨. 컨텍스트 경로 빼고.

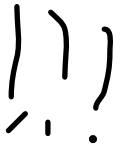


→ 요청 url을 jsp의 경로로 해석

/**sample/ex05** → /WEB-INF/views/**sample/ex05**.jsp

2 Controller의 리턴 타입

- **String 타입**
  - forward: 포워드 방식으로 처리하는 경우
    - "뷰 이름" 문자열 리턴
  - redirect: 리다이렉트 방식으로 처리하는 경우
    - "redirect:요청url" 문자열 리턴



2 Controller의 리턴 타입

HomeController.java

```
...
@Controller
@Slf4j
public class HomeController {

   @GetMapping("/")
   public String home(Model model) {
      model.addAttribute("name", "홍길동");

      return "index";                    // forward
   }
}
```

RedirectAttributes

Servlet에서 redirect 방식

```
response.sendRedirect("/sample/ex06?name=aaa&age=10");
```

재요청시 리퀘스트 스코프 생명주기 끝남.
전에 스코프에 담긴거 유지가 안됨.

Spring MVC 방식

```
// RedirectAttributes ra매개변수 지정 가정
// addFlashAttribute(이름, 값) 메서드로 지정

ra.addAttribute("name", "AAA");
ra.addAttribute("age", 10);

return "redirect:/sample/ex06"; // 뷰 이름이 아닌 요청 경로를 제시
```

첫요청에대한 응답에 첫요청에 대한 결과를
쿼리 스트링에 담아서 보낼 수 있고

또는

간단히 세션스코프에 임시적으로 넣어서

스프링에서는 이를 지워함

RedirectAttributes라는 객체를 매개변수로
DI요청하고
addAttribute메서드 사용.

세션 에 저장은
addFlashAttribute메서드 사용

2 Controller의 리턴 타입

SampleController.java

```
...
public class SampleController {
                ...

   @GetMapping("/ex06")
   public String ex06(RedirectAttributes ra) {
      log.info("/ex06.....");
      ra.addAttribute("name", "AAA");
      ra.addAttribute("age", 10);

      return "redirect:/sample/ex06-2";

   }
}
```

URL바뀜

- http://localhost:8080/sample/ex06 요청

→ http://localhost:8080/sample/ex06-2?name=AAA&age=10 로 리다이렉트 됨

addFlashAttribute메서드는? addAttribute와 사용법이 같다 하지만 redirect후 없어지는 일회용 키-밸류쌍을 저장시킨다 세션스코프에 저장한것처럼

2 Controller의 리턴 타입

- **객체 타입**
  - JSON 타입 응답하는 경우 사용
  - jackson-databind 라이브러리 필요

JSON문자열 <-> 자바객체 파싱이 필요함

기본 자바에는 변환해주는게 없다.

jackson 혹은 gson 라이브러리가 필요함

```
implementation 'com.fasterxml.jackson.core:jackson-databind:2.9.4'
```

dependency있는지 확인해보자 ㅠbuild.gradle파일에서

SampleController.java

```
...
@Controller
public class SampleController {
 ...

        @GetMapping("/ex07")
        public @ResponseBody SampleDTO ex07() {
                log.info("/ex07.....");

                SampleDTO dto = new SampleDTO();
                dto.setAge(10);
                dto.setName("홍길동");

                return dto;
        }
}
```

JSON응답을 내보내겠다는 어노테이션

response바디파트에 들어갈 객체다라는 의미!!!

BL결과를 리턴

http://localhost:8080/sample/ex07

```
{
    "name": "홍길동",
    "age": 10
}
```

응답으로 JSON문자열이 왔다. 자바객체를 반환했는데

바디밖에 핸들링

2 Controller의 리턴 타입

- **ResponseEntity 타입**
  - 브라우저로 직접 응답하는 경우
    - 응답 헤더 설정
    - 응답 바디 설정

응답 구성할때 직접적으로 핸들링할 수 있다

SampleController.java

```
...
import org.springframework.http.HttpHeaders;
...

@Controller
public class SampleController {
  ...

        @GetMapping("/ex08")
        public ResponseEntity<String> ex08() {
                log.info("/ex08.....");

                // {"name": "홍길동"}
                String msg = "{\"name\": \"홍길동\"}";

                HttpHeaders header = new HttpHeaders();
                header.add("Content-Type", "application/json;charset=UTF-8");

                return new ResponseEntity<>(msg, header, HttpStatus.OK);
        }
}
```

body 부분 객체타입

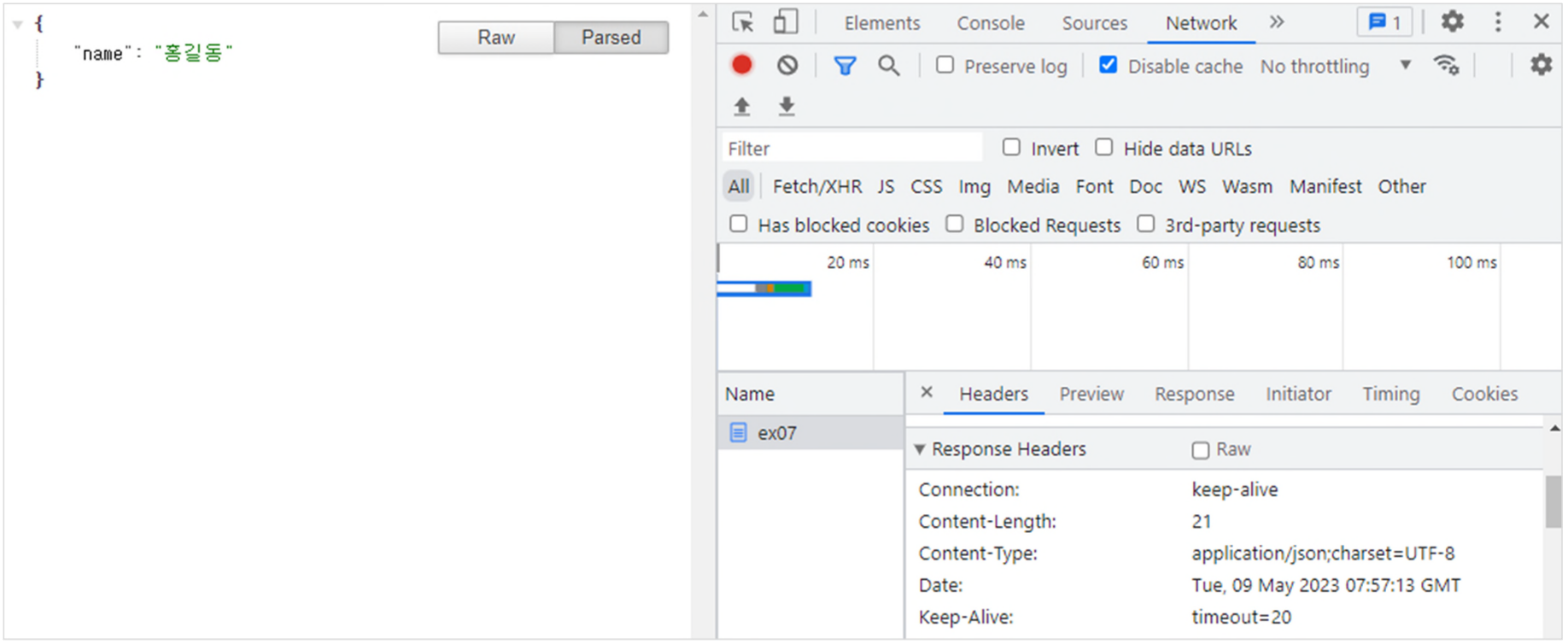
타입에 DTO를 넣어도 될까?

헤더 커스터마이징

바디 헤더 상태코드 다 지정.

2 Controller의 리턴 타입

http://localhost:8080/sample/ex08



중요한 파일 업로드 기능. servlet3.0 이전에는 외부 라이브러리를 사용해야했다

파일 업로드 방법

Servlet 3.0 기능 이용

자체 기능 사용

- multipart 설정
  - location: 업로드 처리 디렉토리 경로
  - maxFileSize: 업로드 가능한 파일 하나의 최대 크기
  - maxRequestSize: 업로드 가능한 전체 최대 크기(여러 파일 업로드 하는 경우)
  - fileSizeThreshold: 메모리 파일의 최대 크기(이보다 작으면 실제 메모리에서만 작업)

업로드 디렉토리 준비

- c:₩upload

업로드했을때 여기다가 저장하겠다

ServletConfig.java

```
...
@EnableWebMvc
@ComponentScan(basePackages = { "org.scoula.controller" })
public class ServletConfig implements WebMvcConfigurer{
        ...

        //      Servlet 3.0 파일 업로드 사용시 - MultipartResolver 빈 등록
        @Bean
        public MultipartResolver multipartResolver() {
                StandardServletMultipartResolver resolver
                        = new StandardServletMultipartResolver();
                return resolver;
        }
}
```

WebConfig.java

```
package org.scoula.config;
...

@Slf4j
@Configuration
public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {
    final String LOCATION = "c:/upload";
    final long MAX_FILE_SIZE = 1024 * 1024 * 10L;   // 10M
    final long MAX_REQUEST_SIZE =  1024 * 1024 * 20L; // 20M
    final int FILE_SIZE_THRESHOLD = 1024 * 1024 * 5;          // 5M

                ...

    @Override
    protected void customizeRegistration(ServletRegistration.Dynamic registration) {
        MultipartConfigElement multipartConfig = new MultipartConfigElement(
                LOCATION,   // 업로드 처리 디렉토리 경로
                MAX_FILE_SIZE,          // 업로드 가능한 파일 하나의 최대 크기
                MAX_REQUEST_SIZE,    // 업로드 가능한 전체 최대 크기(여러 파일 업로드 하는 경우)
                FILE_SIZE_THRESHOLD                  // 메모리 파일의 최대 크기(이보다 작으면 실제 메모리에서만 작업)
            );
        registration.setMultipartConfig(multipartConfig);
    }
}
```

파일 업로드 제약조건과 위치

한파일마다 한번요청시

-1:크기 제한 없음

이크기보다 작으면 메모리써.
이보다 크면 파일에 임시파일로

SampleController.java

form을 통해서 업로드하니까 폼으로 가는 url

```
...
public class SampleController {
  ...

        @GetMapping("/exUpload")
        public void exUpload() {
                log.info("/exUpload.....");
        }

}
```

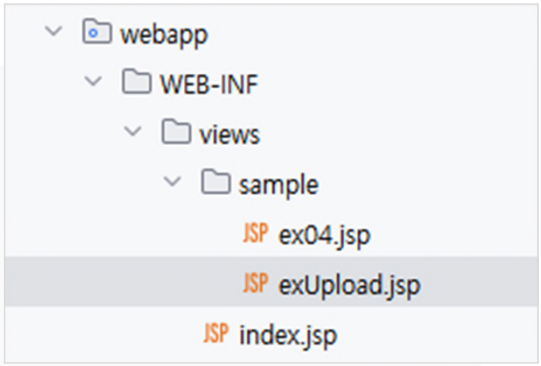
exUpload.jsp 뷰 이름으로 이동

폼을 구성할때 주의사항 바이너리파일을 업로드해야하니까 post메서드로. get메서드는 크기제한 걸리ㅓㅁ.
인코딩 타입을 multipart/form-data로 해줘야함.

3 파일 업로드

KB국민은행

views/sample/exUpload.jsp




```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Insert title here</title>
 </head>
 <body>
  <form action="/sample/exUploadPost" method="post" enctype="multipart/form-data" accept-charset="UTF-8" >
   <div>
    <input type="file" name="files" />
   </div>
   <div>
    <input type="file" name="files" />
   </div>
   <div>
    <input type="file" name="files" />
   </div>
   <div>
    <input type="file" name="files" />
   </div>
   <div>
    <input type="file" name="files" />
   </div>
```

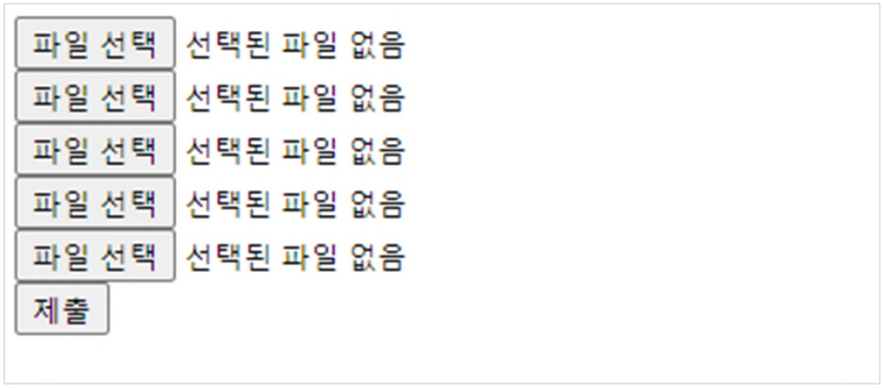
multi part
클라이언트가 서버로 데이터를 보낼때
어떤 인코딩을 쓸건지는 3가지.
1 json
2 url encoding a=b& c=d& ...
3 multipart
바이러니 데이터를 전송할때 사용하는 인코딩 방법
part란? input하나가 파트이다.

part는 자체적으로 헤더와 바디를 가진다.
파트마다 바디를가지고 그 바디를 설명하는 헤더가 있다.

views/sample/exUpload.jsp

```
    <div>
      <input type="submit" />
    </div>
  </form>
 </body>
</html>
```

http://localhost:8080/sample/exUpload



multipart encoding

기존 헤더 정보가 있고

```
Content-Type: multipart/form-data; boundary=frame
<<body>>
--frame
Content-Type: image/jpeg
Content-Length: 33377

[JPEG data]
--frame
Content-Type: image/jpeg
Content-Length: 33377

[JPEG data]

.
.
.
```

기존 헤더

파트를 나눌때 "frame"이라는 문자열을 써서 경계를 나누겠다

파트

헤더

구분 개행

바디

base64 encoding

64가지 정보로 나누어 인코딩

http는 텍스트 기반 프로토콜이니까 64개의 글자로 표현된다. 바이너리 데이터가

파트

base64 인코딩

- Binary Data를 Text로 바꾸는 Encoding(binary-to-text encoding schemes)
- Binary Data를 Character set에 영향을 받지 않는 공통 ASCII 영역의 문자로만 이루어진 문자열로 바꾸는 Encoding

- base64 색인표

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	
15	P	31	f	47	v	63	/

워본 바이너리 데이터

base64 인코딩을 하면 크기가 원본보다 33% 커짐(3byte → 4byte)

SampleController.java

파일 업로드 부분만

나머지는 DTO나 변수로 처리하면 되고

```
...
public class SampleController {
  ...

        @PostMapping("/exUploadPost")
        public void exUploadPost(ArrayList<MultipartFile> files) {

                for(MultipartFile file : files) {
                        log.info("--------------------------------");
                        log.info("name:" + file.getOriginalFilename()); // 윈도우 OS: 한글 파일명인 경우 깨짐
                        log.info("size:" + file.getSize());
                }

        }
}
```

지금 파일들을 받음 리스트로

오늘은 업로드만 시키는 동작까지만

3 파일 업로드

뷰는 없으니까!

로그 잘찍히는지

4 Controller의 Exception 처리

스프링 MVC의 예외 처리

- @ExceptionHandler와 @ControllerAdvice를 이용한 처리
- @ResponseEntity를 이용하는 예외 메시지 구성

에러에는 크게 2가지가 있다.
404 에러, 500에러
가장 흔하게 발생하는 에러들이다.
매번 에러 코드를 직접 하는것보다는 한곳에 모아서 중앙집중식으로 관리하는게 편한다

500에러 처리는 @EceptionHAndler을 써서 특정 예ㅎ외를 감지할 수 있다.
@ControllerAdvice AOP를 통해 처리.

4 Controller의 Exception 처리

@ControllerAdvice

- HTTP 상태코드 500 Internal Serveer Error에 대응하기 위한 기법
- AOP(Aspect-Oriented-Programming)을 이용

- org.scoula.exception.CommonExceptionAdvice.java

관점 지향 프로그래밍. 스프링의 대표기능중 하나
BL을 어떤 관점에서 실행하고 싶으냐.
성능 모니터링 관점에서 실행하고 싶다. 시간측정이 필요하겠죠.
이번에는 보안관점에서 실행하고 싶다.
인증했는지에 대한 검증이 필요함
예외를 처리하는 관점에서 실행하고 싶다.
trycatch문 필요.

이때 BL은 고정되어있는
관점에따라서 전처리 후처리가 맞게 바뀌어야 한다.

이를 잘 하게 하는 프로그래밍이 AOP

4 Controller의 Exception 처리

@Controller
↑
@ControllerAdvice

org.scoula.exception.CommonExceptionAdvice.java

```
package org.scoula.exception;
...

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
@Log4j2
public class CommonExceptionAdvice {

    @ExceptionHandler(Exception.class)
    public String except(Exception ex, Model model) {

        log.error("Exception ....." + ex.getMessage());
        model.addAttribute("exception", ex);
        log.error(model);
        return "error_page";
    }
}
```

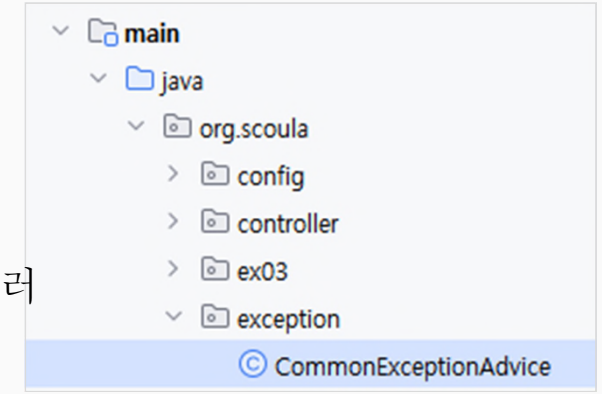
AOP.

에러가 발생했을때 포워딩되서 오는 컨트롤러

예외발생시 나한테로 와
컨트롤러 실행하다가,,

DI요구.

리턴하는 의미도 컨트롤러 다른 메서드와 같음
뷰의 이름임

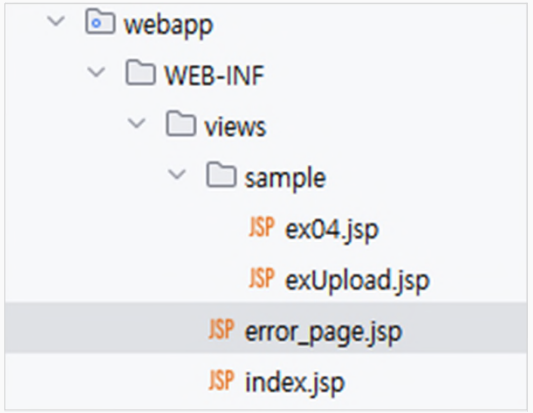


ServletConfig.java

```
package org.scoula.config;

…

@EnableWebMvc
@ComponentScan(basePackages = {
      "org.scoula.controller",
                              "org.scoula.exception",
      "org.scoula.ex03.controller"
}
public class ServletConfig implements WebMvcConfigurer{
  …
}
```

views/error_page.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" import="java.util.*"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h4><c:out value="${exception.getMessage()}"></c:out></h4>

 <ul>
  <c:forEach items="${exception.getStackTrace() }" var="stack">
    <li><c:out value="${stack}"></c:out></li>
  </c:forEach>
 </ul>

</body>
</html>
```



확인

- http://localhost:8080/sample/ex04?name=aaa&age=11
  - page 파리미터 누락

No primary or default constructor found for int

- org.springframework.web.method.annotation.ModelAttributeMethodProcessor.createAttribute(ModelAttrib
- org.springframework.web.servlet.mvc.method.annotation.ServletModelAttributeMethodProcessor.createAtt
- org.springframework.web.method.annotation.ModelAttributeMethodProcessor.resolveArgument(ModelAtt
- org.springframework.web.method.support.HandlerMethodArgumentResolverComposite.resolveArgument(
- org.springframework.web.method.support.InvocableHandlerMethod.getMethodArgumentValues(Invocable
- org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerM
- org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHand
- org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerM
- org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(
- org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMet
- org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:991)
- org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:925)

404 에러 페이지

- 404 에러는 서버에서 Exception을 발생시키지 않음
  - 예외 클래스: NoHandlerFoundException
- 404에러를 Exception으로 처리하려면 설정 필요
  - Advice에서 NoHandlerFoundException 예외 처리

컨트롤러까지 가지 않았으니

디폴트는 예외가 발생하지 않음

예외가 발새하도록 설정 필요

WebConfig.java

```
package org.scoula.config;
…

public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer{
   …

   @Override
   protected void customizeRegistration(ServletRegistration.Dynamic registration) {
      registration.setInitParameter("throwExceptionIfNoHandlerFound", "true");

      // 파일 업로드 설정
                                                …
   }
}
```

핸들러가 없을때 즉 404일때

예외가 발생하게끔 지정

CommonExceptionAdvice.java

```
package org.scoula.exception;
...

public class CommonExceptionAdvice {
    @ExceptionHandler(Exception.class)
    public String except(Exception ex, Model model) {
        log.error("Exception ....." + ex.getMessage());
        model.addAttribute("exception", ex);
        log.error(model);
        return "error_page";
    }

    @ExceptionHandler(NoHandlerFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public String handle404(NoHandlerFoundException ex) {
        log.error(ex);
        model.addAttribute("uri", request.getRequestURI());
        return "custom404";
    }
}
```

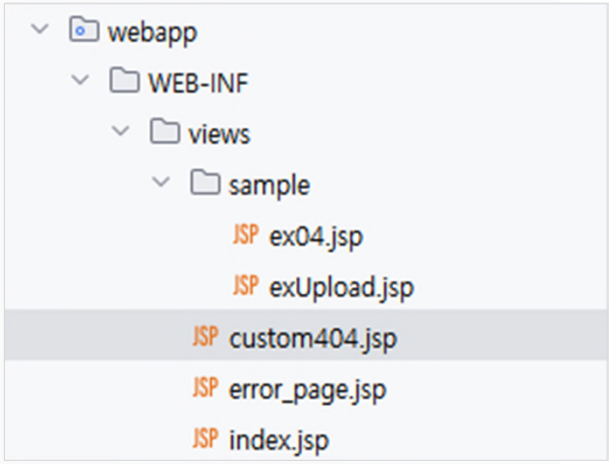
404로 응답 보내고

실제 요청 객체를 uri이름으로 저장함. 리퀘스트 스코프에

뷰 이름 custom404.jsp

views/custom404.jsp

```
<!DOCTYPE html>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h1>해당 URL(${uri})은 존재하지 않습니다.</h1>
</body>
</html>
```



500 404 처리하는 코드도 템플릿에 추가하면 좋겠다!!

또한 멀티파트 처리하는 코드도 템플릿처리!

4 Controller의 Exception 처리

404 에러 페이지

http://localhost:8080/sample/badurl?page=1

해당 URL(/sample/badurl)은 존재하지 않습니다.