KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

스프링 MVC의 Controller

[KB] IT's Your Life

KB 국민은행

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);
```

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
    }

}
```

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
}
```

홍길동 환영합니다.

1 Model이라는 데이터 전달자

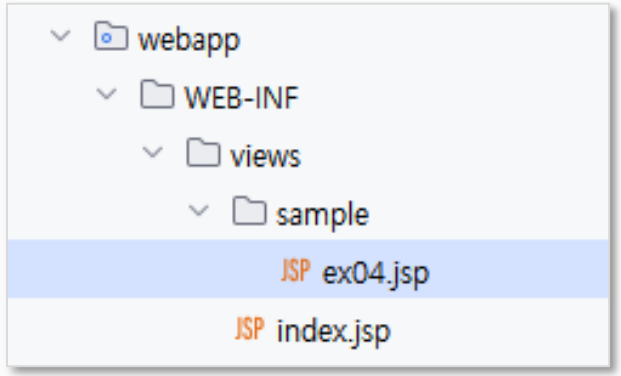
- **@ModelAttribute**
  - DTO 쿼리 파리미터는 자동으로 뷰로 저달됨
  - 기본 자료형 쿼리 파라미터는 뷰로 전달되지 않음
  - @ModelAttribute("파라미터명")
    - 쿼리 파라미터를 request 스코프에 저장
    - 파라미터명이 스코프의 키가 됨

SampleController.java

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

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

PAGE

→ page는 뷰에 전달되지 않음

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";
    }
}
```

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

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

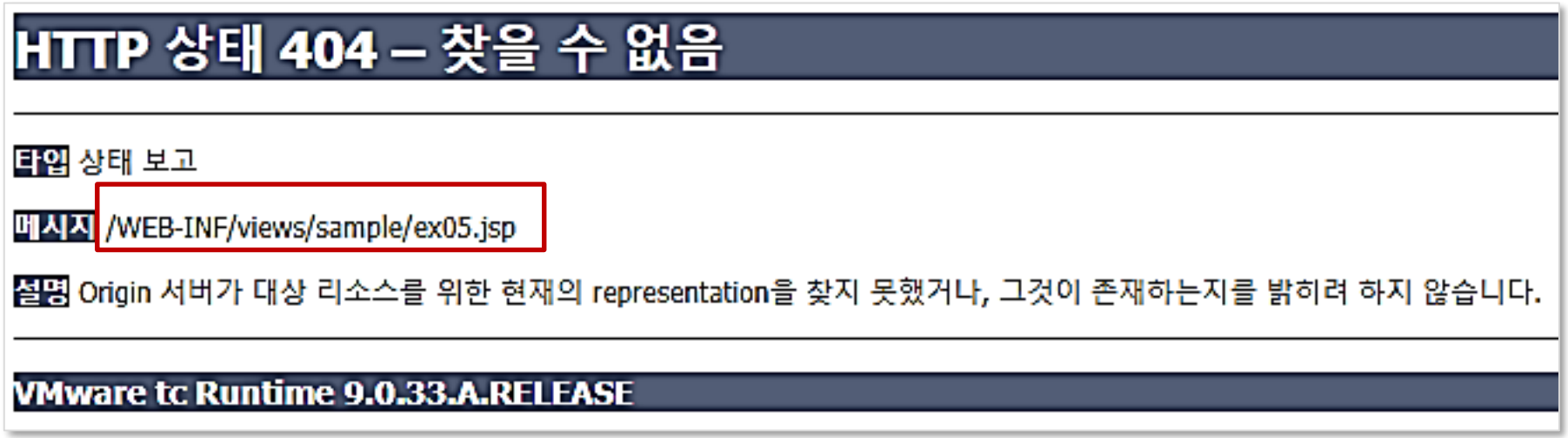
PAGE 9

Controller 메서드의 리턴 타입

- String
  - jsp 뷰의 경로/이름으로 해석
- void
  - 호출한 URL과 동인 이름의 jsp로 해석
- VO, DTO 타입
  - JSON 타입의 데이터로 변환해서 브라우저로 응답
- ResponseEntity 타입
  - Http 헤더 정보와 내용을 가공하여 직접 브라우저로 응답
- Model, ModelAndView
  - Model로 데이터를 변환하거나 뷰이름을 같이 지정
- HttpHeaders
  - 응답에 내용 없이 Http 헤더 메시지만 전달

void 타입

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



→ 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

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");
```

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
```

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";

    }
}
```

2 Controller의 리턴 타입

객체 타입

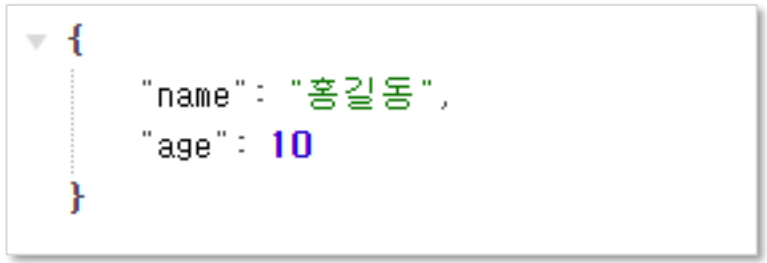
- JSON 타입 응답하는 경우 사용
- jackson-databind 라이브러리 필요

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

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;
    }
}
```

http://localhost:8080/sample/ex07



2 Controller의 리턴 타입

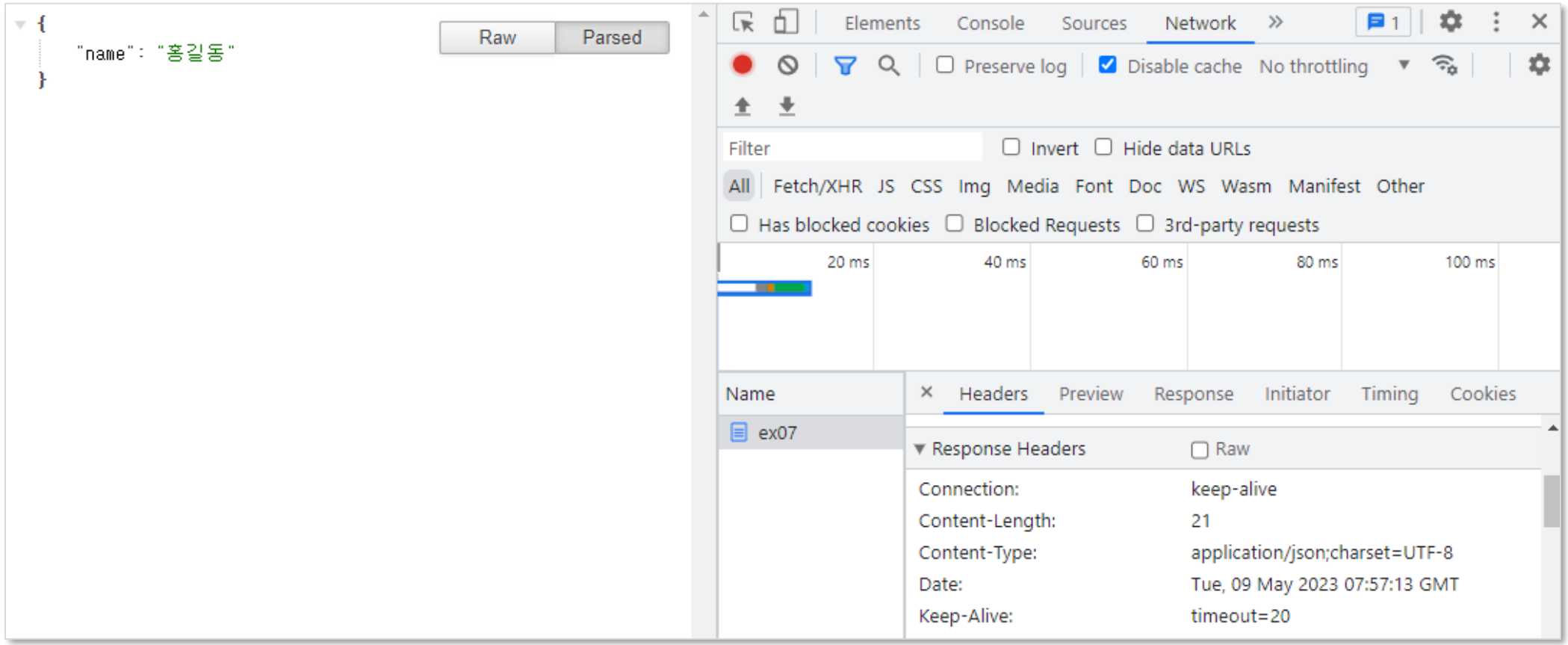
ResponseEntity 타입

- 브라우저로 직접 응답하는 경우
- 응답 헤더 설정
- 응답 바디 설정

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);
    }
}
```

http://localhost:8080/sample/ex08



파일 업로드 방법

Servlet 3.0 기능 이용

- 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

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

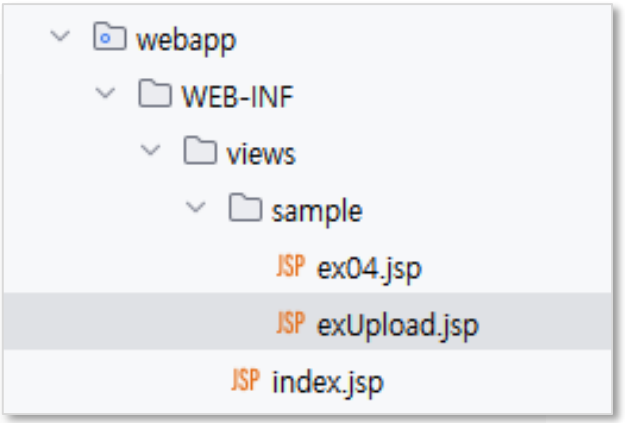
```
package org.scoula.config;
…

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

    …

    @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);
    }
}
```

SampleController.java

```
…
public class SampleController {
    …

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

}
```

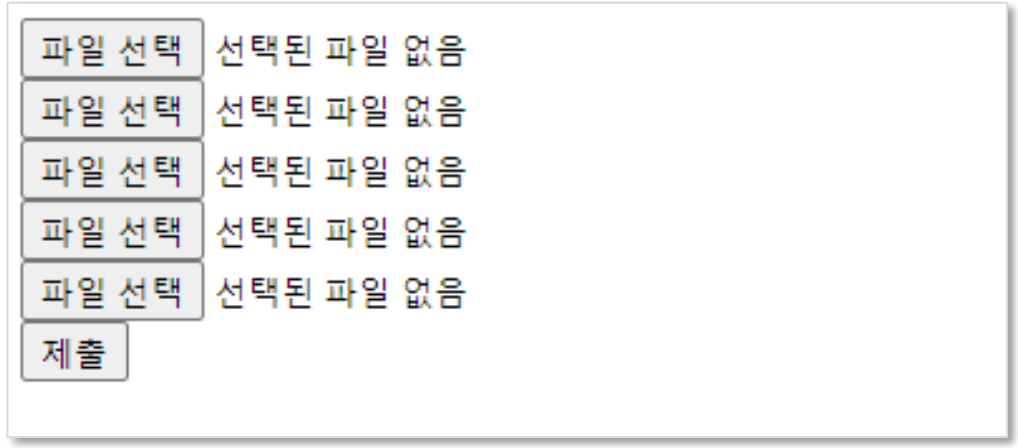
views/sample/exUpload.jsp

```
<%@ 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" >
      <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>
```



views/sample/exUpload.jsp

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



multipart encoding

```
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]

.
.
.
```

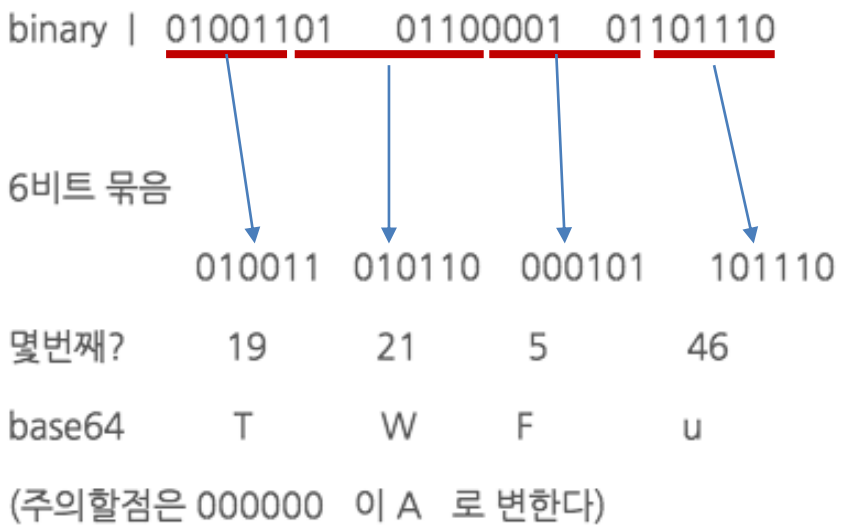
base64 encoding

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	/



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

SampleController.java

```
…
public class SampleController {
    …

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

        for(MultipartFile file : files) {
            log.info("-------------------------------");
            log.info("name:" + file.getOriginalFilename());
            log.info("size:" + file.getSize());
        }

    }
}
```


insert title here
localhost
파일 선택 161fa1ac3a8684a.jpg
파일 선택 888627.jpg
파일 선택 20171211_154611.jpg
파일 선택 구형가계1.jpg
파일 선택 구형가계2.jpg
제출
HTTP Status 404 – Not Found
Type Status Report
Message /WEB-INF/views/sample/exUploadPost.jsp
Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.
INFO : org.zerock.controller.SampleController - name:161fa1ac3a8684a.jpg
INFO : org.zerock.controller.SampleController - size:1018843
INFO : org.zerock.controller.SampleController - name:888627.jpg
INFO : org.zerock.controller.SampleController - size:232402
INFO : org.zerock.controller.SampleController - name:20171211_154611.jpg
INFO : org.zerock.controller.SampleController - size:413005
INFO : org.zerock.controller.SampleController - name:구형가계1.jpg
INFO : org.zerock.controller.SampleController - size:121493
INFO : org.zerock.controller.SampleController - name:구형가계2.jpg
INFO : org.zerock.controller.SampleController - size:304061

4 Controller의 Exception 처리

스프링 MVC의 예외 처리

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

Controller의 Exception 처리

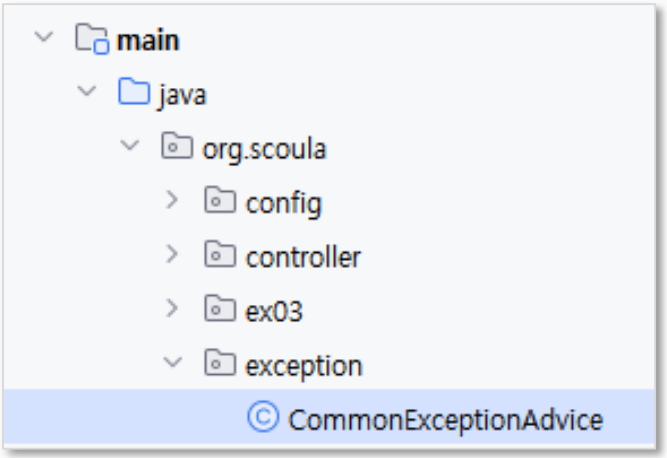
@ControllerAdvice

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

- org.scoula.exception.CommonExceptionAdvice.java

org.scoula.exception.CommonExceptionAdvice.java

```
package org.scoula.exception;
…

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

@ControllerAdvice
@Log4j
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";
    }

}
```

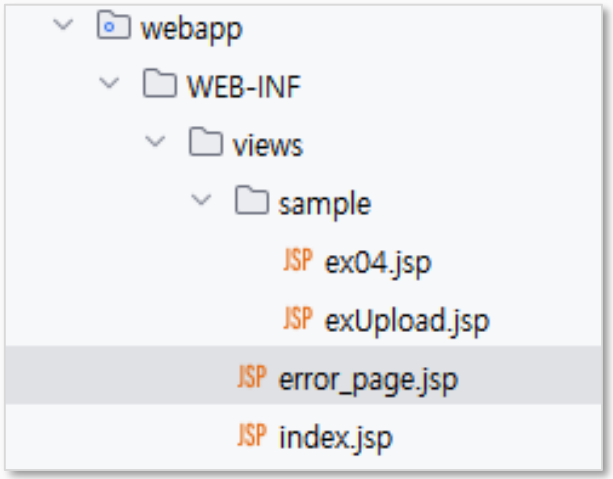


ServletConfig.java

```
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)

4 Controller의 Exception 처리

- 404 에러 페이지

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
            …
    }
}
```

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) {

        return "custom404";
    }
}
```

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 URL은 존재하지 않습니다.</h1>
</body>
</html>
```