

파일 업로드

강사 : 강병준

파일 업로드 처리

1. 파일 업로드가 필요한 경우 html 폼의 enctype을 multipart/form-data으로 설정해야 합니다.
2. 인코딩 타입이 Multipart 인 경우 파라미터나 업로드 한 파일을 구하려면 전송데이터를 알맞게 처리해 주어야 합니다.
3. 스프링은 Multipart 기능을 제공하기 때문에 추가적인 처리없이 Multipart 형식으로 전송된 파라미터와 파일정보를 쉽게 구할 수 있습니다.

MultipartResolver 설정

1. Multipart 지원 기능을 사용하려면 먼저 MultipartResolver를 스프링 설정 파일에 등록해 주어야 합니다.
2. MultipartResolver는 Multipart 형식으로 데이터가 전송된 경우 해당 데이터를 스프링 MVC에서 사용할 수 있도록 변환해 줍니다
3. @PathVariable 어노테이션을 이용해서 Multipart로 전송된 파라미터와 파일을 사용할 수 있도록 해줍니다.
4. 스프링이 기본적으로 제공하는 MultipartResolver는 CommonsMultipartResolver 인데 이 클래스는 Commons FileUpload API를 이용해서 Multipart를 처리합니다.
5. Maven dependency

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.1</version>

</dependency>

6. 위 클래스의 빈을 생성하는 코드를 스프링 설정 파일에 추가

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

</bean>

MultipartResolver 설정

CommonsMultipartResolver 클래스의 프로퍼티

1. long maxUploadSize: 업로드 가능한 파일의 크기로 단위는 byte, 기본값은 -1로 크기 제한 없음
2. int maxInMemorySize: 디스크에 임시 파일을 생성하기 전에 메모리에 보관할 수 있는 최대 바이트 크기로 기본값은 10240바이트
3. String defaultEncoding: 요청을 파싱할 때 사용할 캐릭터 인코딩으로 지정하지 않을 경우 request.setCharacterEncoding으로 설정한 값이 사용되며 이 메서드를 호출하지 않는 경우 ISO-8859-1이 됩니다.

처리 방법

1. 요청을 처리하는 메서드의 매개변수로
`@RequestParam("파라미터이름") MultipartFile 변수명` 의 형태로
받아서 처리
2. 요청을 `MultipartHttpServletRequest`(`HttpServletRequest`로 부터
상속)로 받아서 처리하는 방법
 - 1) `Iterator<String> getFileNamees()`: 파일의 이름 목록을 리턴
 - 2) `MultiPartFile getFile(String name)`: 파라미터의 이름이 `name`인 업로드
파일 정보를 리턴
 - 3) `List<MultiPartFile> getFiles(String name)`: 파라미터의 이름이 `name`인
업로드 파일 정보를 `List`로 리턴
 - 4) `Map<String, MultipartFile> getFileMap()`: 파일에 대한 정보를 맵으로
리턴
3. `Command` 객체를 이용해서 처리 가능

MultipartFile 인터페이스

1. 업로드 한 파일 정보 및 파일 데이터를 표현하기 위한 용도로 사용되는 인터페이스
2. 주요 메서드
 - 1) String getName(): 파라미터 이름을 리턴
 - 2) String getOriginalFileName(): 업로드 한 파일의 원본 이름
 - 3) boolean isEmpty()
 - 4) long getSize()
 - 5) byte[] getBytes(): 업로드 한 파일의 데이터
 - 6) void transferTo(File dest) throws IOException: 파일을 지정한 파일에 저장
3. getBytes()로 파일 데이터를 구한 후 데이터베이스나 파일에 저장하거나 transferTo를 이용해서 파일의 내용을 다른 파일에 전송
4. transferTo를 이용할 때는 파일 객체를 생성한 후 MultipartFile 인터페이스의 transfer 메서드로 대입해서 호출하며 됩니다.

File Upload

```
package com.ch.ch06;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;@Controller
public class FileUploadController {
    private static final Log LOG = LogFactory.getLog(FileUploadController.class);
    @RequestMapping(value = "/upload", method = RequestMethod.GET)
    public String showUploadForm() {
        return "upload";
    }
}
```

File Upload

```
@RequestMapping(value="/upload",method=RequestMethod.POST)  
public String upload(@RequestParam("file") MultipartFile mf,  
    Model model, HttpSession session) throws IllegalStateException,  
    IOException {  
    String fileName = mf.getOriginalFilename();  
    int fileSize = (int)mf.getSize();  
    // mf.transferTo(new File("/gov/"+fileName));  
    String path = session.getServletContext().getRealPath("/upload");  
    String fpath = path + "/" + fileName;  
    FileOutputStream fs = new FileOutputStream(fpath);  
    fs.write(mf.getBytes());  
    fs.close();  
    model.addAttribute("fileName", fileName);  
    model.addAttribute("fileSize", fileSize);  
    return "uploadResult";  
    }  
}
```


File Upload

upload.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>파일 업로드</title>
</head>
<body>
<h1>파일 업로드</h1>
<form method="post" enctype="multipart/form-data">
  <dl>
    <dt>업로드 파일</dt>
    <dd><input type="file" name="uploadFile"></dd>
  </dl>
  <button type="submit">업로드</button>
</form>
</body>
</html>
```

File Upload

1. 이전 프로젝트의 pom.xml 파일에 추가

```
<dependency>
```

```
    <groupId>commons-fileupload</groupId>
```

```
    <artifactId>commons-fileupload</artifactId>
```

```
    <version>1.3.1</version>
```

```
</dependency>
```

2. jsp에 내용 추가

```
<a href="fileupload">파일 업로드 처리</a> <br />
```

Spring MVC Interceptor

강사 강병준

servlet-context.xml 추가

```
<interceptors>
```

```
  <interceptor>
```

```
    <mapping path="/doA"/>
```

```
    <mapping path="/doB"/>
```

```
    <beans:ref bean="sampleInterceptor"/>
```

```
  </interceptor>
```

```
</interceptors>
```

```
<beans:bean id="sampleInterceptor"
```

```
  class="interceptor.SampleInterceptor">
```

```
</beans:bean>
```

TestController작성

```
package com.ch.ch06;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class TestController {
    @RequestMapping("doA")
    public String doA(Model model) {
        model.addAttribute("result", "doA처리 결과");
        return "doA";
    }
    @RequestMapping("doB")
    public String doB(Model model) {
        model.addAttribute("result", "doB처리 결과");
        return "doA";
    }
}
```

SampleInterceptor작성

```
package interceptor;
import java.lang.reflect.Method;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
public class SampleInterceptor extends HandlerInterceptorAdapter {
    public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler,
    ModelAndView modelAndView) throws Exception {
        System.out.println("post handle.....");
    }
    public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler) throws Exception {
        System.out.println("pre handle.....");
        return true;
    }
}
```

SampleInterceptor수정

```
public boolean preHandle(HttpServletRequest request,  
    HttpServletResponse response, Object handler) throws Exception {  
    System.out.println("pre handle.....");  
    HandlerMethod method = (HandlerMethod) handler;  
    Method methodObj = method.getMethod();  
    System.out.println("Bean: " + method.getBean());  
    System.out.println("Method: " + methodObj);  
    return true;  
}
```

pre handle.....

Bean: com.ch.ch06.TestController@61c8c3cb

Method: public java.lang.String

com.ch.ch06.TestController.doB(org.springframework.ui.Model)

SampleInterceptor수정

```
public void postHandle(HttpServletRequest request,  
    HttpServletResponse response, Object handler,  
    ModelAndView modelAndView) throws Exception {  
    System.out.println("post handle.....");  
    Object result = modelAndView.getModel().get("result");  
    if(result != null){  
        System.out.println("result exists");  
        request.getSession().setAttribute("result", result);  
    }  
}
```

pre handle.....

Bean: com.ch.ch06.TestController@1e69d8ab

Method: public java.lang.String

com.ch.ch06.TestController.doA(org.springframework.ui.Model)

post handle.....

result exists

HttpSession을 이용한 로그인 처리

TestController에 추가

```
public class TestController {  
    @RequestMapping("/loginForm")  
    public String logForm() {  
        return "loginForm";  
    }  
}
```

servlet-context.xml 추가

```
<interceptor>  
    <mapping path="/doA"/>  
    <mapping path="/list"/>  
    <beans:ref bean="loginTest"/>  
</interceptor>
```

LoginInterceptor.java작성

```
package interceptor;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.springframework.ui.ModelMap;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
public class LoginInterceptor extends HandlerInterceptorAdapter {
    public void postHandle(HttpServletRequest request, HttpServletResponse
        response, Object handler, ModelAndView modelAndView) throws Exception {
    }
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
        Object handler) throws Exception {
        HttpSession session = request.getSession();
        if (session.getAttribute("id") == null) {
            response.sendRedirect("loginForm"); return false;
        }
        return true;
    }
}
```