9_2_2022_SPRING

로그인 기능 넣기

▼ UsersController.java

```
package com.livelikesloth.step03.users.controller;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class UsersController {
  //로그인 요청을 처리할 컨트롤러 메소드
  @RequestMapping("/users/login")
  public ModelAndView login(String id, HttpSession session) {
    //입력한 아이디 비밀번호가 유효한 정보이면 로그인 처리를 한다.
    session.setAttribute("id", id); //입력한 아이디를 session 영역에 담는다.
    return new ModelAndView("users/login");
  }

  //로그 아웃 처리를 할 컨트롤러 메소드
  @RequestMapping("/users/logout")
  public String logout(HttpSession session) {
    //session 영역에 id라는 키값으로 저장된 값 삭제하기
    session.removeAttribute("id"); //로그아웃 처리
    return "redirect:/home.do"; //인덱스 페이지로 리다이렉트 시키기
  }

  //로그아웃 폼 요청 처리를 할 컨트롤러 메소드
  @RequestMapping("/users/loginform")
  public String loginform() {

    return "users/loginform";
  }
}
```

Aspect

▼ Aspectj Expression

- execution(* *(..)) → 접근 가능한 모든 메소드 point cut
- execution(* test.service.*.*(..)) → test.service 패키지의 모든 메소드 point cut
- execution(void insert*(..)) → 리턴 type은 void 이고 메소드명이 insert 로 시작하는 모든 메소드 Point cut
- execution(* delete*(*)) → 메소드명이 delete로 시작하고 인자로 1개 전달받는 메소드 point cut (aop가 적용되는 위치)
- execution(* delete*(*,)) → 메소드 명이 delete로 시작하고 인자로 2개 전달받는 메소드 point cut (aop 적용되는 위치)
- execution(String update*(Integer,*)) → 메소드 명이 update 로 시작하고 리턴 type은 String
  메소드의 첫번째 인자는 Integer type, 두번 째 인자는 아무 type 다되는 메소드 point cut(aop 적용되는 위치)

▼ 코드

```
package com.livelikesloth.step03.aspect;

import java.net.URLEncoder;

import javax.servlet.http.HttpServletRequest;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.ModelAndView;

@Aspect  //aspect 역할을 할 수 있도록
@Component// component scan을 통해 bean이 될 수 있도록
public class AuthAspect {

  /*
   * @Around("aspectj expression")
   *
   * execution(MedelAndView auth*(..) 라는 aspectj expression 은
   *
   * 리턴 type 은 MedelAndView 이고  메소드의 이름은 auth로 시작하는 모든 메소드를 가리킨다.
   *
   * 따라서 spring 이 관리하는 객체중에 위의 모양을 가지고 있는 메소드는
   *
   * 아래의 aspect 가 적용이 된다.
   */
     @Around("execution(org.springframework.web.servlet.ModelAndView auth*(..))")
     public Object loginCheck(ProceedingJoinPoint joinPoint) throws Throwable {
        Object[] args=joinPoint.getArgs();
        //메소드에 전달된 인자중에서 HttpServletRequest 객체를 찾는다.
        for(Object tmp:args) {
           if(tmp instanceof HttpServletRequest) {
              //찾았으면 원래 type 으로 casting
```

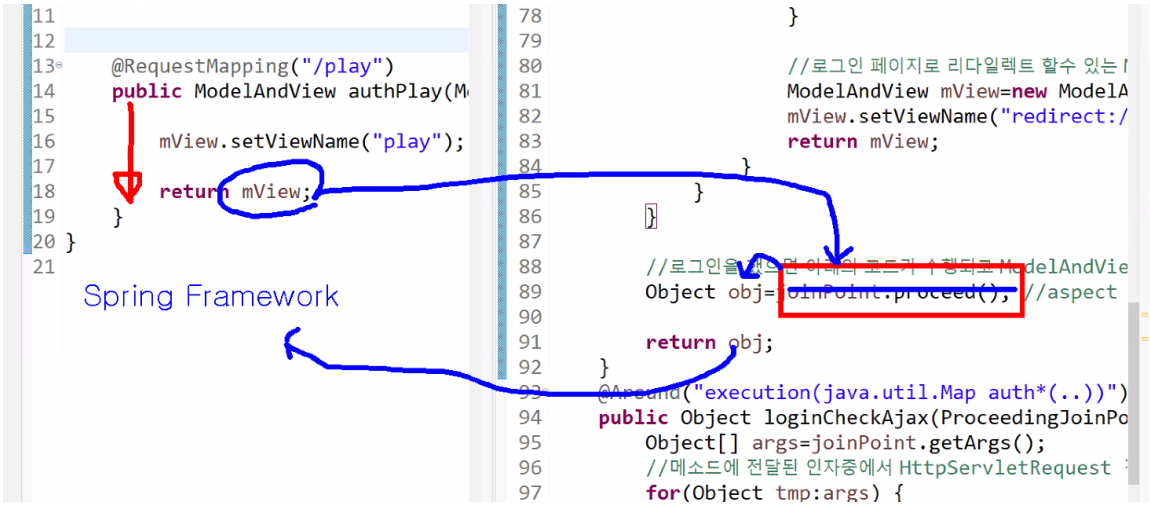
```
               HttpServletRequest request=(HttpServletRequest)tmp;
               //HttpSession 객체의 참조값 얻어와서 로그인 여부를 알아낸다.
               String id=(String)request.getSession().getAttribute("id");
               if(id == null) {//만일 로그인을 하지 않았으면
                  //원래 가려던 url 정보 읽어오기
                  String url=request.getRequestURI();
                  //GET 방식 전송 파라미터를 query 문자열로 읽어오기 ( a=xxx&b=xxx&c=xxx )
                  String query=request.getQueryString();
                  //특수 문자는 인코딩을 해야한다.
                  String encodedUrl=null;
                  if(query==null) {//전송 파라미터가 없다면
                     encodedUrl=URLEncoder.encode(url);
                  }else {
                     // 원래 목적지가 /test/xxx.jsp 라고 가정하면 아래와 같은 형식의 문자열을 만든다.
                     // "/test/xxx.jsp?a=xxx&b=xxx ..."
                     encodedUrl=URLEncoder.encode(url+"?"+query);
                  }

                  //로그인 페이지로 리다일렉트 할수 있는 ModelAndView 객체를 생성해서
                  ModelAndView mView=new ModelAndView();
                  mView.setViewName("redirect:/users/loginform.do?url="+encodedUrl);
                  return mView;
               }
            }
         }

         //로그인을 했으면 아래의 코드가 수행되고 ModelAndView 객체가 Object type 으로 리턴된다.
         Object obj=joinPoint.proceed();

         return obj;
      }
      @Around("execution(java.util.Map auth*(..))")
      public Object loginCheckAjax(ProceedingJoinPoint joinPoint) throws Throwable {
         Object[] args=joinPoint.getArgs();
         //메소드에 전달된 인자중에서 HttpServletRequest 객체를 찾는다.
         for(Object tmp:args) {
            if(tmp instanceof HttpServletRequest) {
               //찾았으면 원래 type 으로 casting
               HttpServletRequest request=(HttpServletRequest)tmp;
               //HttpSession 객체의 참조값 얻어와서 로그인 여부를 알아낸다.
               String id=(String)request.getSession().getAttribute("id");
               if(id == null) {//만일 로그인을 하지 않았으면
                  //예외를 발생 시켜서 정상적인 응답을 받을수 없도록 한다.
                  throw new RuntimeException("로그인이 필요 합니다.");
               }
            }
         }

         //로그인을 했으면 아래의 코드가 수행되고 ModelAndView 객체가 Object type 으로 리턴된다.
         Object obj=joinPoint.proceed();

         return obj;
      }
}
```

(..) 인자가 있든 없든 관여하지 않겠다

```
execution(MedelAndView auth*(..)
```

▼ PlayController.java

```
package com.livelikesloth.step03;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class PlayController {

  @RequestMapping("/play")
  public ModelAndView authPlay(ModelAndView mView, HttpServletRequest request) {
    mView.setViewName("play");

    return mView;
  }
}
```



filter, aspect 차이점

- filter

  클라이언트의 요청이 서블릿 혹은 jsp 페이지에 도달하기 전에 요청경로를 확인해서 동작

- aspect

  spring framework 가 동작하는 도중에 특정 pattern의 메소드가 호출될 때 동작

spring framework가 동작하는 도중 →
xxx.do요청을 해서 DispatcherServlet을 거친 이후 →

controller or service or dao 의 메소드 주에 어디에든 적용시킬 수 있다.

파일 업로드 다운로드

▼ servlet-context.xml

```
<!-- Multipart 폼 전송 처리를 위한 bean 설정 -->
   <beans:bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
      <beans:property name="maxUploadSize" value="102400000"/>
   </beans:bean>
```

▼ home.jap

```
<h1>파일 업로드 폼</h1>
    <form action="upload.do" method="post" enctype="multipart/form-data">
      <input type="text" name="title" placeholder="설명입력" /><br />
      <input type="file" name="myFile" /><br />
      <button type="submit">업로드</button>
    </form>
```

▼ FileController.java

```
package com.livelikesloth.step03;

import java.io.File;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class FileController {
   /*
    *  <input type="text" name="title" placeholder="설명입력..."/><br />
       <input type="file" name="myFile"/><br />

         위에서  name 속성의 값인 title 과 myFile 이라는 이름으로 매개변수명을 정하면
         전송된  text 는  title 변수에 담기고
         전송된  파일에 대한 정보(파일명, 파일의크기, 전송된 파일객체)는 myFile 객체에 담긴다.
    */
```

```
    @RequestMapping("/file/upload")
    public ModelAndView upload(String title, MultipartFile myFile,
          HttpServletRequest request) {

       //원본 파일명
       String orgFileName=myFile.getOriginalFilename();
       //파일 사이즈
       long fileSize=myFile.getSize();

       //upload 폴더에 저장할 파일명을 직접구성한다.
       // 1234123424343xxx.jpg
       String saveFileName=System.currentTimeMillis()+orgFileName;

       // webapp/upload 폴더까지의 실제 경로 얻어내기
       String realPath=request.getServletContext().getRealPath("/upload");
       // upload 폴더가 존재하지 않을경우 만들기 위한 File 객체 생성
       File upload=new File(realPath);
       if(!upload.exists()) {//만일 존재 하지 않으면
          upload.mkdir(); //만들어준다.
       }
       //파일을 저장할 전체 경로를 구성한다.
       String savePath=realPath+File.separator+saveFileName;
       try {
          //임시폴더에 업로드된 파일을 원하는 파일을 저장할 경로에 전송한다.
          myFile.transferTo(new File(savePath));
       }catch(Exception e) {
          e.printStackTrace();
       }

       //업로드된 파일의 정보를 원래는 db에 저장해야 하지만
       //테스트중이니 그냥 MedelndView에 담고 view Page에서 확인만 해보기
       ModelAndView mView=new ModelAndView();
       mView.addObject("title", title);
       mView.addObject("fileSize", fileSize);
       mView.addObject("orgFileName", orgFileName);
       mView.addObject("saveFileName", saveFileName);
       mView.addObject("savePath", savePath);
       mView.setViewName("file/upload");

       return mView;
    }
}
```

▼

db 연동해서 파일 업로드 다운로드 spring 기반으로 만들기

자료실 만들기

다음주 월요일까지