Session : 사용자별 임시 저장 공간

Request를 통하여 접근 가능

사용자별의 기준 = IP, 브라우저, MAC Address

로그인 : session에 사용자 정보를 보관.

-AOP(Aspect Oriented Programming) : 관점지향 프로그래밍. 중간에 끼어드는 프로그램

인터셉터, AOP, ......

로그인 체크, 권한 체크하는데 주로 사용

AOP

1. Pointcut으로 범위 설정

2. JoinPoint로 Pointcut 대상 수행 구현

Around(전부다)

============================================================

before after after-returning

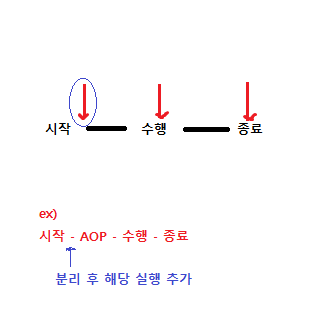
시작 - 수행 - 종료 - 값 리턴

ㄴ 예외

after-throwing

============================================================

주로 around를 많이 쓰고, 상황에 따라 before로 구현



|  |
| --- |
| package com.spring.sample.common.controller;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  import org.aspectj.lang.ProceedingJoinPoint;  import org.aspectj.lang.annotation.Around;  import org.aspectj.lang.annotation.Aspect;  import org.aspectj.lang.annotation.Pointcut;  import org.springframework.web.context.request.RequestContextHolder;  import org.springframework.web.context.request.ServletRequestAttributes;  import org.springframework.web.servlet.ModelAndView;  @Aspect  public class CommonAOP {  //Pointcut -> 적용범위  //@Pointcut(범위설정)  /\*  \* 범위  \* execution -> include필터  \* !execution -> exclude필터  \* \* -> 모든것  \* \*(..) -> 모든 메소드  \* .. -> 모든 경로  \* && -> 필터 추가  \*/  // TestController 밑에 있는 b로 시작하는 메소드에 걸겠다.  @Pointcut("execution(\* com.spring.sample..TestController.b\*(..))"  + "&&!execution(\* com.spring.sample..TestController.\*Log\*(..))"  + "&&!execution(\* com.spring.sample..TestController.\*Join(..))"  + "&&!execution(\* com.spring.sample..TestController.\*s(..))")  public void testAOP() {}    //ProceedingJoinPoint -> 대상 적용 이벤트 필터  /\*  \* @Before -> 메소드 실행 전  \* @After -> 메소드 실행 후  \* @After-returning -> 메소드 정상실행 후  \* @After-throwing -> 메소드 예외 발생 후  \* @Around -> 모든 동작시점  \*/  @Around("testAOP()") // @Around(포인트컷메소드) : 포인트컷에 할당된 메소드의 모든 동작시점에 적용  public ModelAndView testAOP(ProceedingJoinPoint joinPoint)  throws Throwable {  ModelAndView mav = new ModelAndView();    //Request 객체 취득  HttpServletRequest request  = ((ServletRequestAttributes)RequestContextHolder.currentRequestAttributes()).getRequest();    HttpSession session = request.getSession(); // Session 취득  if(session.getAttribute("sBmNo") != null) {  mav = (ModelAndView) joinPoint.proceed(); //기존 이벤트 처리 행위를 이어서 진행  } else {  mav.setViewName("redirect:bLogin");  }      System.out.println("------- testAOP 실행됨 ------");    return mav;  }  } |

|  |
| --- |
| @RequestMapping(value = "/bJoin")  public ModelAndView bJoin(ModelAndView mav) {    mav.setViewName("test/bJoin");    return mav;  }    @RequestMapping(value = "/bJoins")  public ModelAndView bJoins(@RequestParam HashMap<String, String> params, ModelAndView mav) {    int res = 0;    try {  int cnt = iTestService.getBmIdCheck(params);    if(cnt > 0) {  res = 1;  } else {  iTestService.insertJoin(params);  }  } catch (Throwable e) {  res = 2;  e.printStackTrace();  }  mav.addObject("res", res);  mav.setViewName("test/bJoins");  return mav;  }    @RequestMapping(value = "/bLogin")  public ModelAndView bLogin(HttpSession session, ModelAndView mav) {  if(session.getAttribute("sBmNo") != null) {  mav.setViewName("redirect:bMain");  } else {  mav.setViewName("test/bLogin");  }  return mav;  }    @RequestMapping(value = "/bLogins")  public ModelAndView bLogins(//HttpServletRequest req, // 세션 취득(옛날 방식)  HttpSession session,  @RequestParam HashMap<String,String> params, ModelAndView mav) throws Throwable {    HashMap<String,String> data = iTestService.getBm(params);    if(data != null) { // 해당 사용자 정보가 있을 때  //HttpSession session = req.getSession(); // 세션 정보 취득(옛날 방식)  //세션 정보 할당(충돌 방지하기 위해 앞에 s를 달아줌)  session.setAttribute("sBmNo", data.get("BM\_NO"));  session.setAttribute("sBmId", data.get("BM\_ID"));  session.setAttribute("sBmNm", data.get("BM\_NM"));  //세션 정보 취득  System.out.println(session.getAttribute("sBmNm"));    mav.setViewName("redirect:bMain");    } else { // 사용자 정보가 없을 때  mav.setViewName("test/bLogins");  }  return mav;  }    @RequestMapping(value = "/bMain")  public ModelAndView bMain(HttpSession session, ModelAndView mav) {    mav.setViewName("test/bMain");    return mav;  }    @RequestMapping(value = "/bLogout")  public ModelAndView bLogout(HttpSession session, ModelAndView mav) {  session.invalidate(); // 세션 초기화    mav.setViewName("redirect:bLogin");    return mav;  } |

|  |
| --- |
| <insert id="insertJoin" parameterType="hashmap">  INSERT INTO BM(BM\_NO, BM\_ID, BM\_PW, BM\_NM)  VALUES(BM\_SEQ.NEXTVAL, #{bmId}, #{bmPw}, #{bmNm})  </insert>    <select id="getBmIdCheck" parameterType="hashmap" resultType="Integer">  SELECT COUNT(BM\_ID)  FROM BM  WHERE BM\_ID = #{bmId}  </select>    <select id="getBm" parameterType="hashmap" resultType="hashmap">  SELECT BM\_NO, BM\_ID, BM\_NM  FROM BM  WHERE BM\_ID = #{bmId}  AND BM\_PW = #{bmPw}  </select> |