

1.1:응용소프트웨어 개발에 필요한 하드웨어 및 소프트웨어의 필요 사항을 검토하고 이에따라, 개발환경에 필요한 준비를 수행할수있다.

= [Java 설치](#) , [Eclipse 다운로드](#) , [Maven 설치](#) , [apache tomcat 설치](#)

1.2응용소프트웨어 개발에 필요한 하드웨어 및 소프트웨어를 설치하고 설정하여 개발환경을 구축할수있다.

Java 설치

무료 Java 다운로드

지금 데스크톱 컴퓨터용 Java를 다운로드하십시오!

Version 8 Update 161

릴리스 날짜: 2018년 1월 16일



» [Java란 무엇입니까?](#) » [Java가 설치되어 있습니까?](#) » [도움말이 필요하십니까?](#)

Eclipse 다운로드




Apache Tomcat 설치 완료 화면

192.168.2.56:8080


Insert title here V3_관리자 Insert title here Spring

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/7.0.84

 SOFTWARE FOUNDATION
http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

 **Recommended Reading:**
[Security Considerations HOW-TO](#)
[Manager Application HOW-TO](#)
[Clustering/Session Replication HOW-TO](#)

Server Status
Manager App
Host Manager

Developer Quick Start

[Tomcat Setup](#)
[First Web Application](#)

[Realms & AAA](#)
[JDBC DataSources](#)

[Examples](#)

[Servlet Specifications](#)
[Tomcat Versions](#)

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 7.0 access to the manager application is split between different users.
[Read more...](#)

[Release Notes](#)
[Changelog](#)
[Migration Guide](#)
[Security Notices](#)

Documentation

[Tomcat 7.0 Documentation](#)
[Tomcat 7.0 Configuration](#)
[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

[Tomcat 7.0 Bug Database](#)
[Tomcat 7.0 JavaDocs](#)
[Tomcat 7.0 SVN Repository](#)

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).

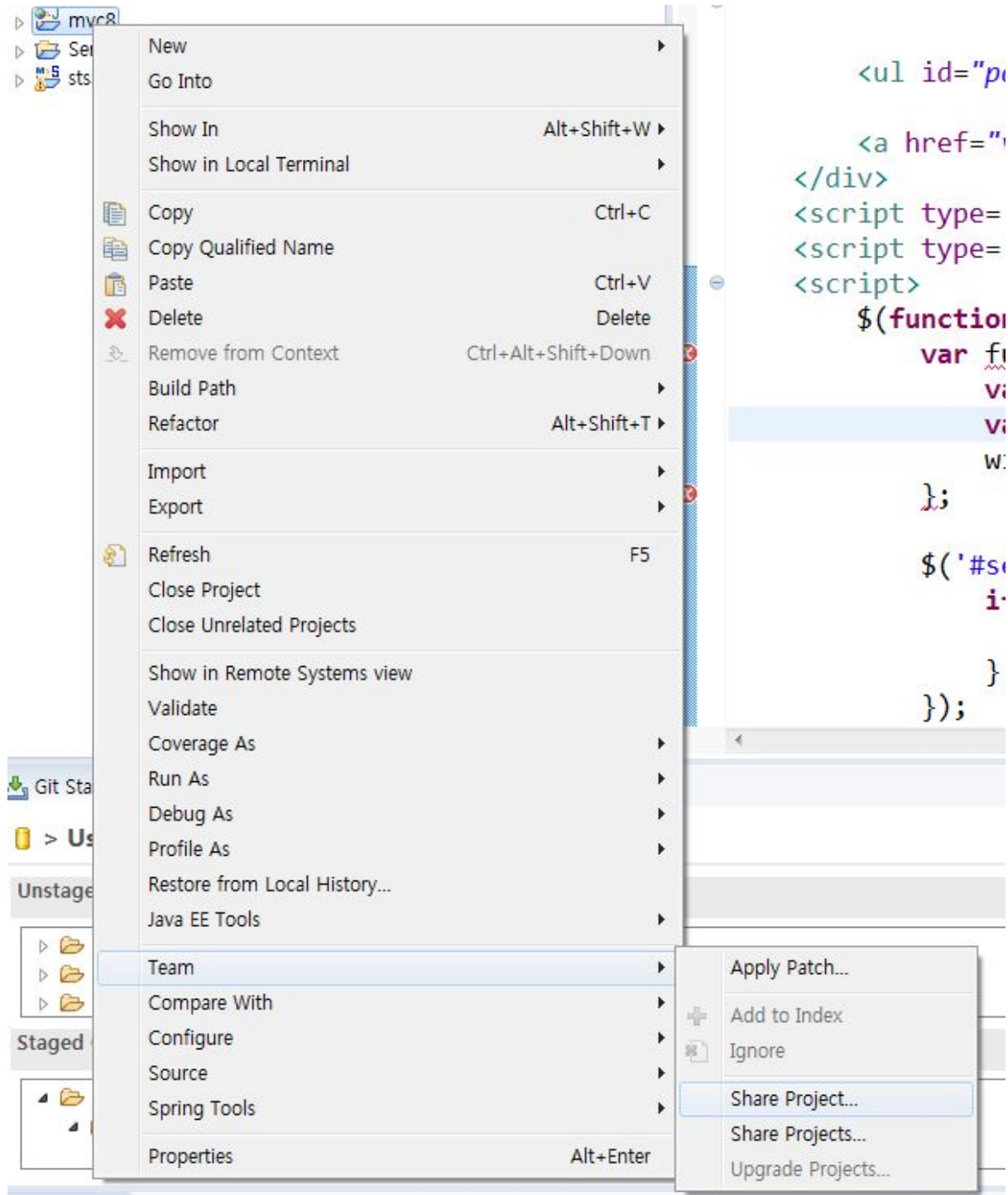
[tomcat-users](#)
User support and discussion

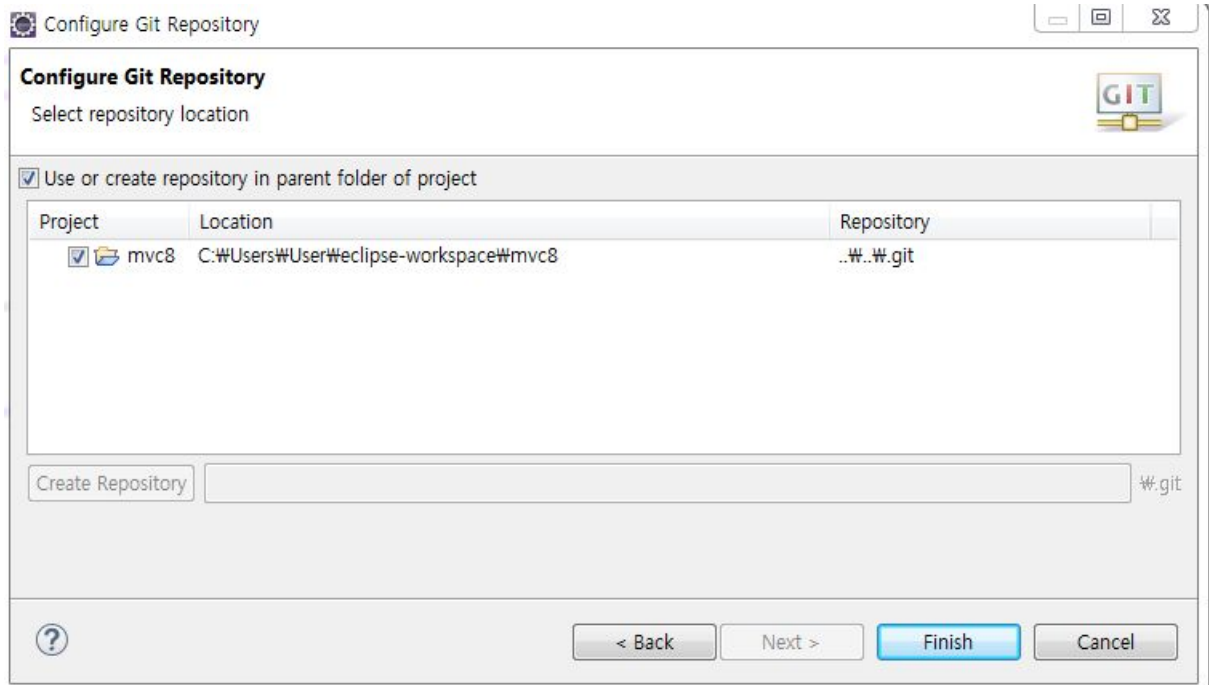
[taglibs-user](#)
User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)
Development mailing list, including commit messages

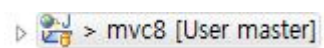
1.3:사전에 수립된 형상관리 방침에 따라, 운영정책에 부합하는 형상관리 환경을 구축할 수 있다.

형상관리도구 : Git 사용





완료된 모습



2.1 공통 모듈의 상세 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 업무 프로세스 및 서비스의 구현에 필요한 공통 모듈을 작성할수있다.

VO

```
public class V7_Member {
    private String mem_id = null;
    private String mem_pw = null;
    private String mem_name = null;
    private String mem_tel = null;
    private String mem_email = null;
    private String mem_date = null;
    private boolean mem_auth = false;

    public boolean isMem_auth() {
        return mem_auth;
    }
    public void setMem_auth(boolean mem_auth) {
        this.mem_auth = mem_auth;
    }
    public String getMem_id() {
        return mem_id;
    }
    public void setMem_id(String mem_id) {
        this.mem_id = mem_id;
    }
    public String getMem_pw() {
        return mem_pw;
    }
    public void setMem_pw(String mem_pw) {
        this.mem_pw = mem_pw;
    }
    public String getMem_name() {
        return mem_name;
    }
    public void setMem_name(String mem_name) {
        this.mem_name = mem_name;
    }
}
```

Controller

```
@RequestMapping(value="/v7_memberjoin.do",method = RequestMethod.GET)
public String memberjoin(Model model,V7_Member obj) {
    model.addAttribute("obj",obj);
    return "v7_memberjoin";
}

@RequestMapping(value="/v7_memberjoin.do",method = RequestMethod.POST)
public String memberjoin(@ModelAttribute("obj")V7_Member obj) {
    mDAO.memberJoin(obj);
    return "redirect:v7_main.do";
}
```

DAO

```

@Insert("INSERT INTO V7_MEMBER(mem_id, mem_pw, mem_name, mem_email, mem_tel,mem_date)"
      + "VALUES("
      + "#{vo.mem_id},"
      + "#{vo.mem_pw},"
      + "#{vo.mem_name},"
      + "#{vo.mem_email},"
      + "#{vo.mem_tel},"
      + "SYSDATE)")
public int memberJoin(@Param("vo") V7_Member vo);

```

2.2 소프트웨어 측정지표 중 모듈간의 결합도는 줄이고 개별 모듈들의 내부 응집도를 높인 공통 모듈을 구현할 수 있다.

VO

```

public class V7_Member {
    private String mem_id = null;
    private String mem_pw = null;
    private String mem_name = null;
    private String mem_tel = null;
    private String mem_email = null;
    private String mem_date = null;
    private boolean mem_auth = false;

    public boolean isMem_auth() {
        return mem_auth;
    }

    public void setMem_auth(boolean mem_auth) {
        this.mem_auth = mem_auth;
    }

    public String getMem_id() {
        return mem_id;
    }

    public void setMem_id(String mem_id) {
        this.mem_id = mem_id;
    }

    public String getMem_pw() {
        return mem_pw;
    }

    public void setMem_pw(String mem_pw) {
        this.mem_pw = mem_pw;
    }

    public String getMem_name() {
        return mem_name;
    }

    public void setMem_name(String mem_name) {
        this.mem_name = mem_name;
    }
}

```

Controller


```

@RequestMapping(value = "/v7_memberedit.do", method = RequestMethod.GET)
public String memberEdit(@ModelAttribute("vo") V7_Member vo, Model model) {
    if(vo != null) {
        if(vo.isMem_auth() == false) {
            return "redirect:v7_memberauth.do";
        }
        V7_Member vo1 = mDAO.selectMemberOne(vo);
        model.addAttribute("obj", vo1);
        return "v7_memberedit";
    }
    else {
        return "redirect:v7_memberlogin.do";
    }
}

@RequestMapping(value="/v7_memberedit.do", method = RequestMethod.POST)
public String memberEdit(@ModelAttribute("obj") V7_Member vo) {
    int ret = mDAO.updateMemberOne(vo);
    if(ret > 0) {
        return "redirect:v7_main.do";
    }
    return "redirect:v7_memberedit.do";
}

```

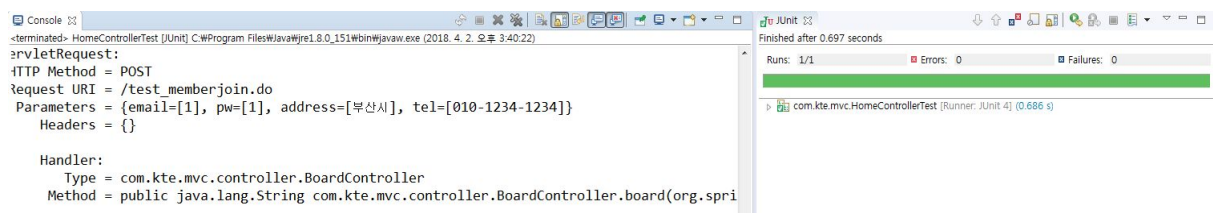
DAO

```

@Update("UPDATE V7_MEMBER SET mem_name=#{vo.mem_name}, mem_tel=#{vo.mem_tel}, mem_email=#{vo.mem_email} WHERE mem_id=#{vo.mem_id}")
public int updateMemberOne(@Param("vo") V7_Member vo);

```

2.3 개발된 공통 모듈의 내부기능과 제공하는 인터페이스에 대해 테스트할수있는 테스트케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.



3.1 애플리케이션 설계를 기반으로, 프로그래밍 언어와 도구를 활용하여 서버 프로그램 구현 기술에 부합하는 서버 프로그램을 개발할 수 있다.

VO

```
public class V7_Board {
    private int brd_no = 0;
    private String brd_title = null;
    private String brd_content = null;
    private int brd_hit = 0;
    private String brd_date = null;
    private String mem_id = null;
    private int brd_cd_no = 0;

    public int getBrd_no() {
        return brd_no;
    }
    public void setBrd_no(int brd_no) {
        this.brd_no = brd_no;
    }
    public String getBrd_title() {
        return brd_title;
    }
    public void setBrd_title(String brd_title) {
        this.brd_title = brd_title;
    }
    public String getBrd_content() {
        return brd_content;
    }
    public void setBrd_content(String brd_content) {
        this.brd_content = brd_content;
    }
    public int getBrd_hit() {
        return brd_hit;
    }
    public void setBrd_hit(int brd_hit) {
        this.brd_hit = brd_hit;
    }
}
```

Controller


```

@RequestMapping(value = "/v7_boardw.do", method = RequestMethod.GET)
public String boardWrite(Model model, @RequestParam(value="code", defaultValue="1") int no) {
    int max_no = bDAO.selectBoardMaxNo();
    V7_Board vo = new V7_Board();
    vo.setBrd_no(max_no + 1);
    //vo.setBrd_cd_no(no);
    vo.setMem_id("123");
    model.addAttribute("vo",vo);
    return "v7_boardw";
}

@RequestMapping(value = "/v7_boardw.do", method = RequestMethod.POST)
public String boardWrite(Model model,
    @RequestParam("code") int code,
    @ModelAttribute("vo")V7_Board vo,
    MultipartHttpServletRequest request) {
    try {
        //input type="file"태그의 첨부된 파일을 map으로 받음
        Map<String, MultipartFile> map
            = request.getFileMap();

        V7_BoardImg imgVo = new V7_BoardImg();
        //V7_Board에서 꺼내어 V7_BoardImg에 번호 넣음
        imgVo.setBrd_no( vo.getBrd_no() );

        //input type="file" 개수 만큼 반복
        for(int i=0;i<map.size();i++) {
            //1개의 파일을 가져옴.
            MultipartFile tfile = map.get("img"+(i+1));
            //파일의 첨부유무 확인
            if(tfile != null
                && !tfile.getOriginalFilename().equals("")) {
                if(i==0) {
                    imgVo.setBrd_img_1( tfile.getBytes() );
                }
                if(i==1) {
                    imgVo.setBrd_img_2( tfile.getBytes() );
                }
                if(i==2) {
                    imgVo.setBrd_img_3( tfile.getBytes() );
                }
            }
        }

        vo.setBrd_cd_no(code);
        bDAO.insertBoard(vo, imgVo);
        return "redirect:v7_board.do?code="+code;
    }
}

```

DAO

```

<div class="container">
    <form:form action="v7_boardw.do?code=${param.code}" method="post"
        modelAttribute="vo" enctype="multipart/form-data">
        <form:input type="text" path="brd_no" readonly="true"/><br />
        <form:input type="text" path="brd_title" /><br />
        <form:textarea path="brd_content"></form:textarea><br />
        <form:input type="text" path="mem_id" readonly="true" /><br />
        <input type="file" name="img1" /><br />
        <input type="file" name="img2" /><br />
        <input type="file" name="img3" /><br />
        <input type="submit" value="글쓰기" />
        <a href="v7_main.do">메인</a><br />
    </form:form>

```

3.2 클라이언트 프로그램에 대한 종속도를 낮출 수 있고 쉽게 연동할 수 있는 서버 프로그램을 개발할 수 있다.

VO

```
public class V7_Board {
    private int brd_no = 0;
    private String brd_title = null;
    private String brd_content = null;
    private int brd_hit = 0;
    private String brd_date = null;
    private String mem_id = null;
    private int brd_cd_no = 0;

    public int getBrd_no() {
        return brd_no;
    }
    public void setBrd_no(int brd_no) {
        this.brd_no = brd_no;
    }
    public String getBrd_title() {
        return brd_title;
    }
    public void setBrd_title(String brd_title) {
        this.brd_title = brd_title;
    }
    public String getBrd_content() {
        return brd_content;
    }
    public void setBrd_content(String brd_content) {
        this.brd_content = brd_content;
    }
    public int getBrd_hit() {
        return brd_hit;
    }
    public void setBrd_hit(int brd_hit) {
        this.brd_hit = brd_hit;
    }
}
```

Controller

```

@RequestMapping(value = "/v7_board.do", method = RequestMethod.GET)
public String boardList(Model model,
    @RequestParam(value="code", defaultValue="1") int no,
    @RequestParam(value="page", defaultValue="1") int page,
    @RequestParam(value="type", defaultValue="brd_title")String type,
    @RequestParam(value="text", defaultValue="")String text) {
    if(no == 0) {
        return "redirect:v7_board.do?code=1";
    }
    List<V7_BoardCode> code = bDAO.selectBoardCode();
    model.addAttribute("code", code);

    //int totPage = bDAO.selectBoardTotPage(no);
    int totPage = bDAO.selectBoardTotPage2(no,type,text);
    model.addAttribute("totPage",(totPage-1)/10 +1);
    //1 = 1
    //2 = 11
    //3 = 21
    /*List<V7_Board> list = bDAO.selectBoardList(no);*/
    List<V7_Board> list = bDAO.selectBoardList2(no, page*10-9,type,text);
    model.addAttribute("list",list);
    return "v7_board";
}

```

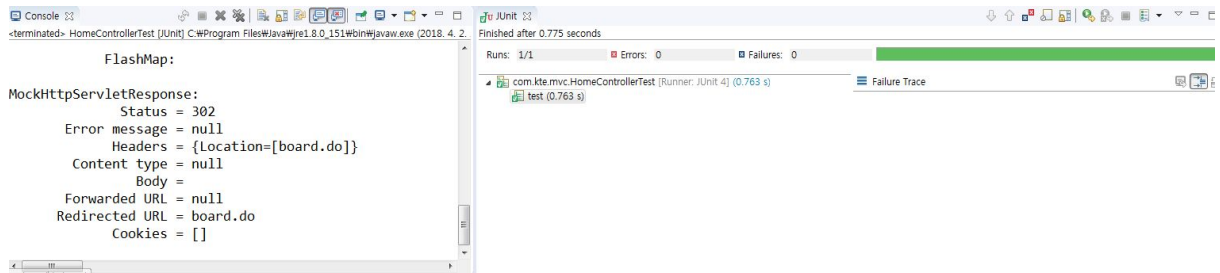
SQL

```

@Select("SELECT * FROM ( " +
    "SELECT brd_no, brd_title, brd_hit, mem_id, brd_date, ROW_NUMBER() OVER (ORDER BY brd_no DESC) rown " +
    "FROM V7_BOARD " +
    "WHERE brd_cd_no = #{code} AND ${type} LIKE '%' || #{text} || '%" +
    "WHERE rown BETWEEN #{page} and #{page}+9")
public List<V7_Board> selectBoardList2(
    @Param("code")int a,
    @Param("page") int b,
    @Param("type") String ty,
    @Param("text") String tx);

```

3.3 개발된 서버 프로그램 내부 기능과 제공하는 인터페이스에 대해 테스트할 수 있는 테스트 케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건명을 시할 수 있다.



4.1 애플리케이션 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 배치 프로그램 구현 기술에 부합하는 배치 프로그램을 개발할 수 있다.

```
<insert id="insertItemList" parameterType=com.kte.mvc.vo.V6_Item">
BEGIN
FOR i IN 1..#{count} LOOP
INSERT INTO NO,TITLE,CONTENT,WRITER,DATE FROM V6_ITEM VALUES(#{no},#{title},#{content},#{writer},SYSDATE)
END LOOP
</insert>
```

4.2 목표 시스템을 구성하는 하위 시스템간의 연동 시, 안정적이고 안전하게 동작할 수 있는 배치 프로그램을 개발할 수 있다.

```
<delete id="deleteItemList" parameterType=com.kte.mvc.vo.V6_Item">
BEGIN
FOR i IN 1..#{count} LOOP
DELETE * FROM V6_ITEM WHERE NO = #{no}
END LOOP
</delete>
```

4.3 개발하고자 하는 목표 시스템 잠재적 보안 취약성이 제거될 수 있도록 배치 프로그램을 개발할 수 있다 .

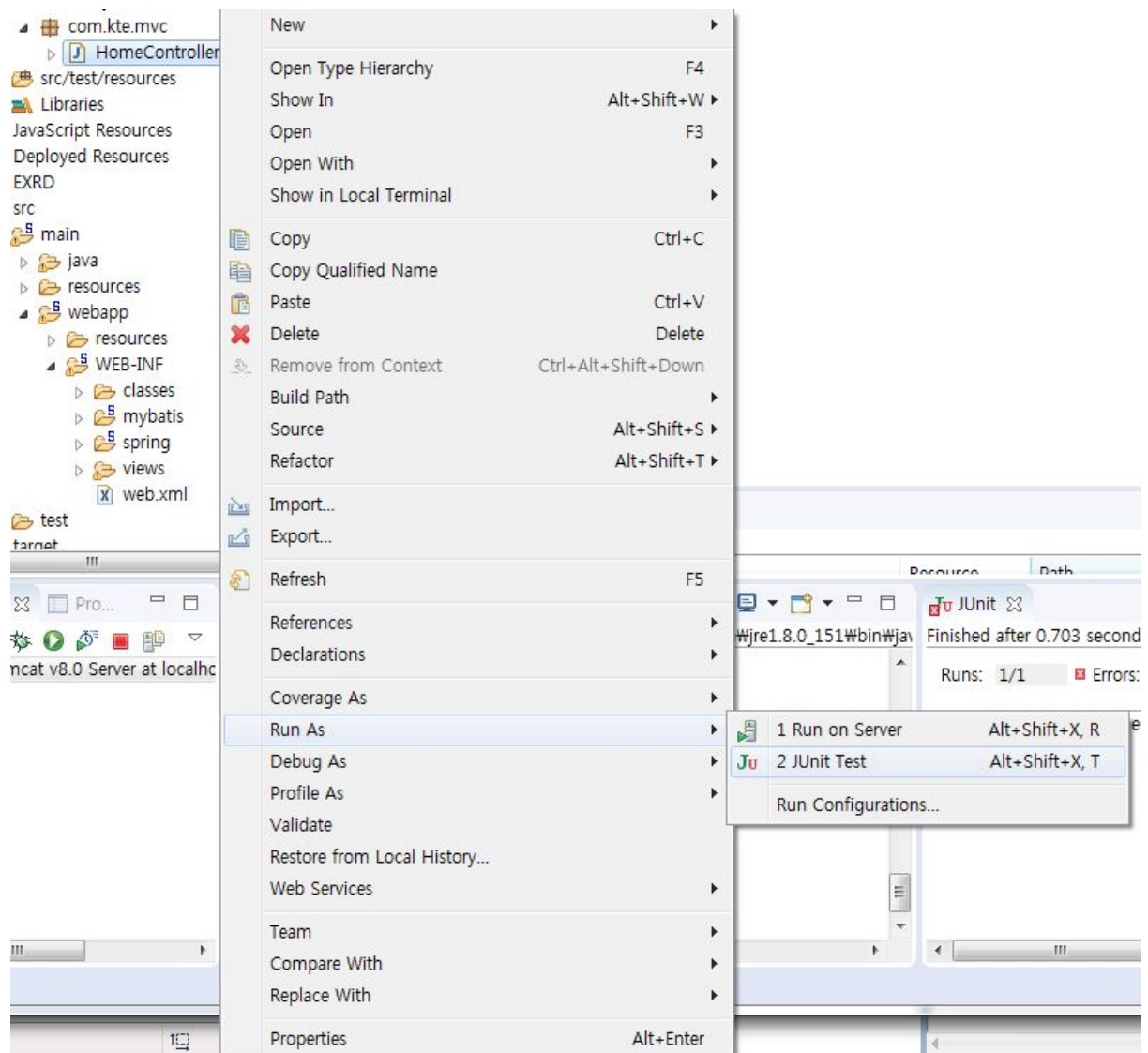
```
<update id="updateItemList" parameterType=com.kte.mvc.vo.V6_Item">
BEGIN
FOR i IN 1..#{count} LOOP
UPDATE NO,TITLE,CONTENT,WRITER,DATE FROM V6_ITEM VALUES(#{no},#{title},#{content},#{writer},SYSDATE)
END LOOP
</update>
```

4.4 개발된 배치 프로그램을 테스트할 수 있는 테스트 케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.

log4j.xml 결과 화면

BRD_NO	BRD_TITLE	BRD_HIT	MEM_ID	BRD_DATE	ROWN
2	2	1231	123	2018-03-29 00:00:00	[unread]
1	1	1	123	[null]	[unread]

5.1 구현한 응용소프트웨어 단위가 설계 내용을 반영하는지 여부를 판단하기 위한 단위테스트의 표준, 절차, 기법 등을 정의할 수 있다.



5.2 기능요구사항을 분석하여 단위테스트 계획을 수립하고, 단위 테스트 계획대로 단위 모듈/컴포넌트 별로 테스트를 수행할 수 있다.

작성자	남동현		작성일자	2018.4.02
식별번호	구분	사전조건	수행절차	기대결과
join1.1	ID중복체크	1234,###,abc,가나다 ,(공백)	회원가입 페이지에서 중복확인 버튼 클릭	영문 id에 를 제외한 데이터에 오류 메세지 출력
join1.2	ID값	123456,1234,#####	아이디값 입력 후 회원가입 버튼 클릭	123456 → 성공 1234→실패 (길이짧음) #####→실패 (특수문자) 실패시 오류메세지 출력
join1.3	PW값	a,aaaa1212,#####	암호값 입력 후 회원가입 버튼 클릭	aaaa1212→성공 a→길이짧음 #####→실패 (특수문자) 실패시 오류메세지 출력
join1.4	PW확인	aaaa1212,#####	암호 재확인 입력 후 회원가입 버튼 클릭	aaaa1212→성공 (동일한데이터) #####→실패 (pw값과다른 데이터) 실패시 오류메세지 출력
join1.5	이름값	홍길동,외자,가나다라,123 4,####	이름값 입력 후 회원가입 버튼 클릭	홍길동→성공 외자→성공 가나다라→성공 1234→실패 (숫자입력) ####→실패 (특수문자) 실패시 오류메세지 출력
join1.6	생년월일값	20010101,200101,12345 678,#####	생년월일값 입력 후 회원가입 버튼 클릭	20010101→성공 200101→실패 (8자리입력) 12345678→실패 (월,일값이 맞지않음) #####→실패 (특수문자) 실패시 오류 메세지 출력
join1.7	전화번호값	055-1234-5678,053-####- ####,02-가나다라-마바사 아	전화번호값 입력후 회원가입 버튼 클릭	055-1234-5678→ 성공 053-####-####→

				실패 (특수문자) 02-가나다라-마바 사아→실패 (문자) 실패시 오류메세지 출력
join1.8	핸드폰번호값	010-1234-5678,010-####- ####,010-가나다라-마바 사아	핸드폰번호값 입력후 회원가입 버튼 클릭	010-1234-5678→ 성공 010-####-####→ 실패 (특수문자) 010-가나다라-마바 사아→실패 (문자) 실패시 오류메세지 출력
join1.9	우편번호값	12345,가나다라마,#####, 666666	우편번호값 입력후 회원가입 버튼 클릭	12345→성공 666666→성공 (5 또는 6자리) 가나다라마→실패 (문자)
join2.0	주소값	부산시 미남역,@@@@@,123 123	주소값 입력후 회원가입 버튼 클릭	부산시 미남역→성공 @@@@@→성 공 123123→성공

5.3 단위 모듈/컴포넌트가 설계 내용을 만족하는지 여부를 계획한 단위 테스트 케이스에 따라 검증할 수 있다.

작성자	남동현		작성일자	2018.4.02
식별번호	구분	사전조건	수행절차	기대결과
write1.0	TITLE	1234,###,abc,가나다 ,(공백)	제목 입력후 글쓰기 버튼 클릭	abc,가나다,1234 =성공 ###,(공백) =실패 실패시 오류메세지 출력
write1.1	CONTENT	1234,###,abc,가나다 ,(공백)	내용 입력후 글쓰기 버튼 클릭	abc,가나다,1234,# ## =성공 (공백) =실패 실패시 오류메세지

				출력
write1.2	FILE	aaa.jpg, bbb.png, (공백), (영상)	파일 첨부후 글쓰기 버튼 클릭	aaa.jpg,bbb.png, (공백) =성공 (영상) =실패 실패시 오류메세지 출력

5.4 단위 테스트 결과 발견된 결함과 이슈를 식별하고, 단위 테스트 결과 분석을 통하여 테스트의 충분성 여부를 검증할 수 있다.

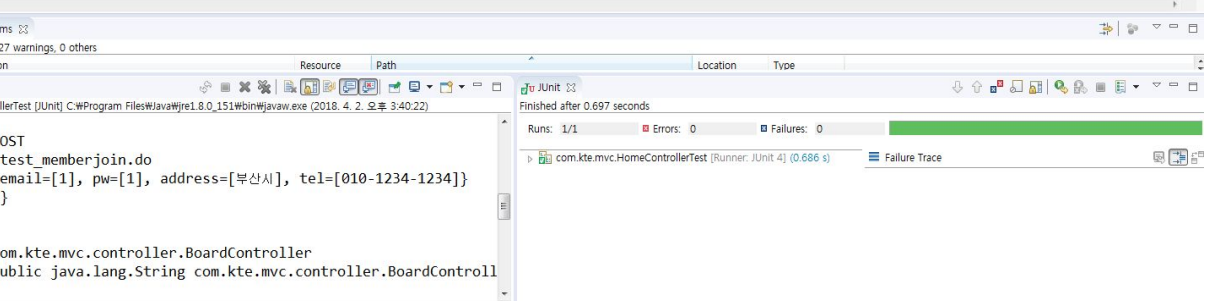
```
private MockMvc mockMvc = null;

@Before // 테스트 실행 전
public void setup() {
    // 테스트 하고자 하는 controller객체 만들
    mockMvc = MockMvcBuilders.standaloneSetup(new BoardController()).build();
}

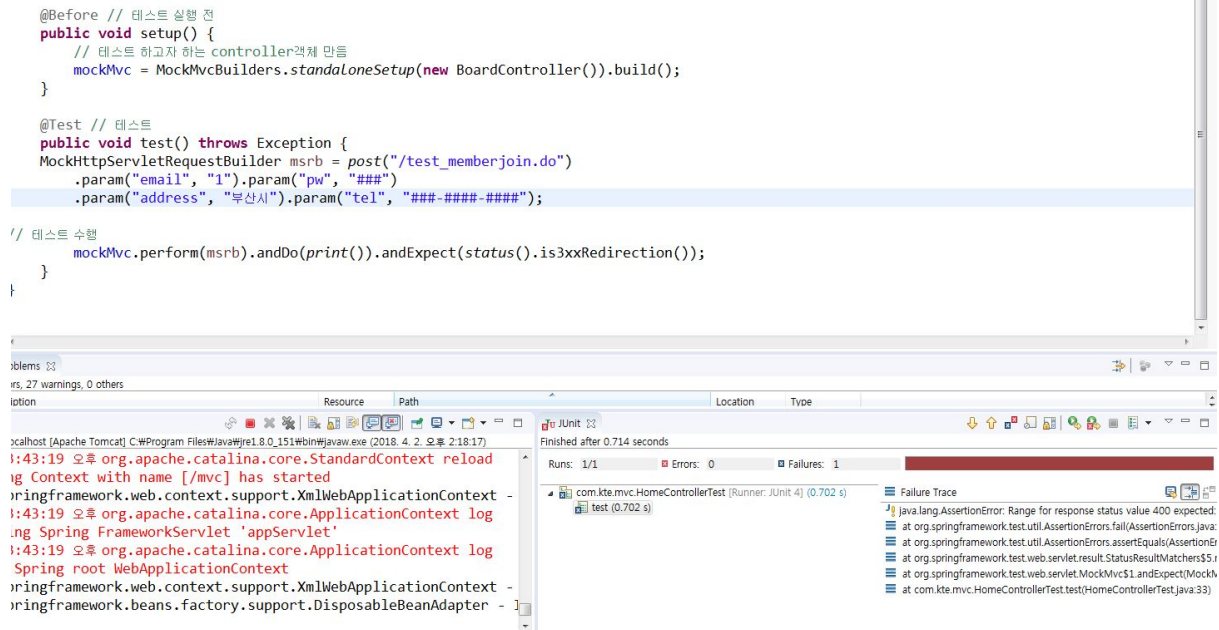
@Test // 테스트
public void test() throws Exception {
    MockHttpServletRequestBuilder msrb = post("/test_memberjoin.do")
        .param("email", "1").param("pw", "1")
        .param("address", "부산시").param("tel", "010-1234-1234");

    mockMvc.perform(msrb).andDo(print()).andExpect(status().is3xxRedirection());
}

// 테스트 수행
}
```



5.5 단위 테스트 결과 발견된 결함에 대한 개선의 시스템 반영 여부를 검증하고, 필요할 경우 시정조치를 실시할 수 있다.



6.1 실 데이터를 기반으로 테스트를 수행하여 애플리케이션의 성능을 확인하고 목표 성능이 충족되도록 개선할 수 있다.

```

@RequestMapping(value="/v7_memberedit.do",method = RequestMethod.POST)
public String memberEdit(@ModelAttribute("obj") V7_Member vo) {
    try {
        int ret = mDAO.updateMemberOne(vo);
        if(ret > 0) {
            return "redirect:v7_main.do";
        }
        return "redirect:v7_memberedit.do";
    }
    catch(Exception e) {
        System.out.println(e.getMessage());
        return "redirect:v7_memberedit.do";
    }
}

```

5.2 애플리케이션 성능을 개선하기 위해, 기 정의된 프로그래밍 언어 표준 가이드라인에 따른 코드 품질 매트릭을 이해하고 적용할 수 있다.

불러온 데이터의 이미지가 없을때의 기본 이미지 처리

```

if(imgData == null) {
    InputStream in = request.getSession().getServletContext().getResourceAsStream("/resources/imgs/default-image.jpg");
    imgData = IOUtils.toByteArray(in);
}

```

6.3 애플리케이션 성능을 개선하기 위해, 프로그래밍 언어와 이의 표준에 대한 이해를 바탕으로 소스코드에 내재된 품질 수준을 분석하기 위한 도구를 활용할 수 있다.

```
@Controller
public class Test_Controller1 {
    public static class testtest{
        protected int a = 1;

        public int getA() {
            return a;
        }

        public void setA(int a) {
            this.a = a;
        }
    }
}

import com.kte.mvc.controller.Test_Controller1.testtest;

@Controller
public class Test_Controller2 extends testtest{
    private void a() {
        System.out.println(a);
    }
}
```