

SPRING의 MVC2 패턴을 적용한 부동산 홈페이지 구현

강 지 수 (참여 인원 5명)

목차

01. 프로젝트 개요

02. 시스템 구성

03. 담당 파트 소개 - 마이페이지

04. 프로젝트 소감

01. 프로젝트 개요

○ 목 적 : MVC2 등을 적용하여 부동산 거래 웹사이트 생성
(회원가입, 부동산 매물 정보, 매물 등록 및 관리, 마이페이지, 게시판 생성)

○ 기 간 : 2021.03.09. ~ 04.15.(4주간)

○ 기대효과

- ▲ 프로그래밍 언어 숙달
- ▲ MVC2 구조 습득
- ▲ 정보 추천 알고리즘 방식 이해

○ 결과 : 부동산 거래 홈페이지 구축 완료

01. 프로젝트 개요

프로젝트 개발 기간 (WBS)

SPRING 의 MVC2 패턴을 적용한 부동산 홈페이지 구현

사전 준비

03.09~03.10

팀 구성 및 프로젝트 아
이디어 공유

기획

03.15~03.16

주제 선정 및 구체화

요구 분석

03.17~03.18

요구사항 정의서 및
테이블 흐름도 작성

설계

03.19~03.22

각 페이지 초안 및 DB
테이블/변수명 선정

구현 시작

03.23~04.06

기본기능 중심 코드 구현
및 DB 연동

구현 완료

04.07~04.09

디자인 적용(부트스트랩,
CSS)
상세 주석 수정

통합 및 테스트

04.12~04.14

각 파트 통합 및 오류 개선

마무리

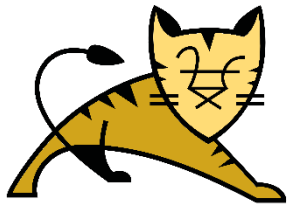
04.14~04.15

발표자료 작성 및 제출

02. 시스템 구성 - 사용 프로그램



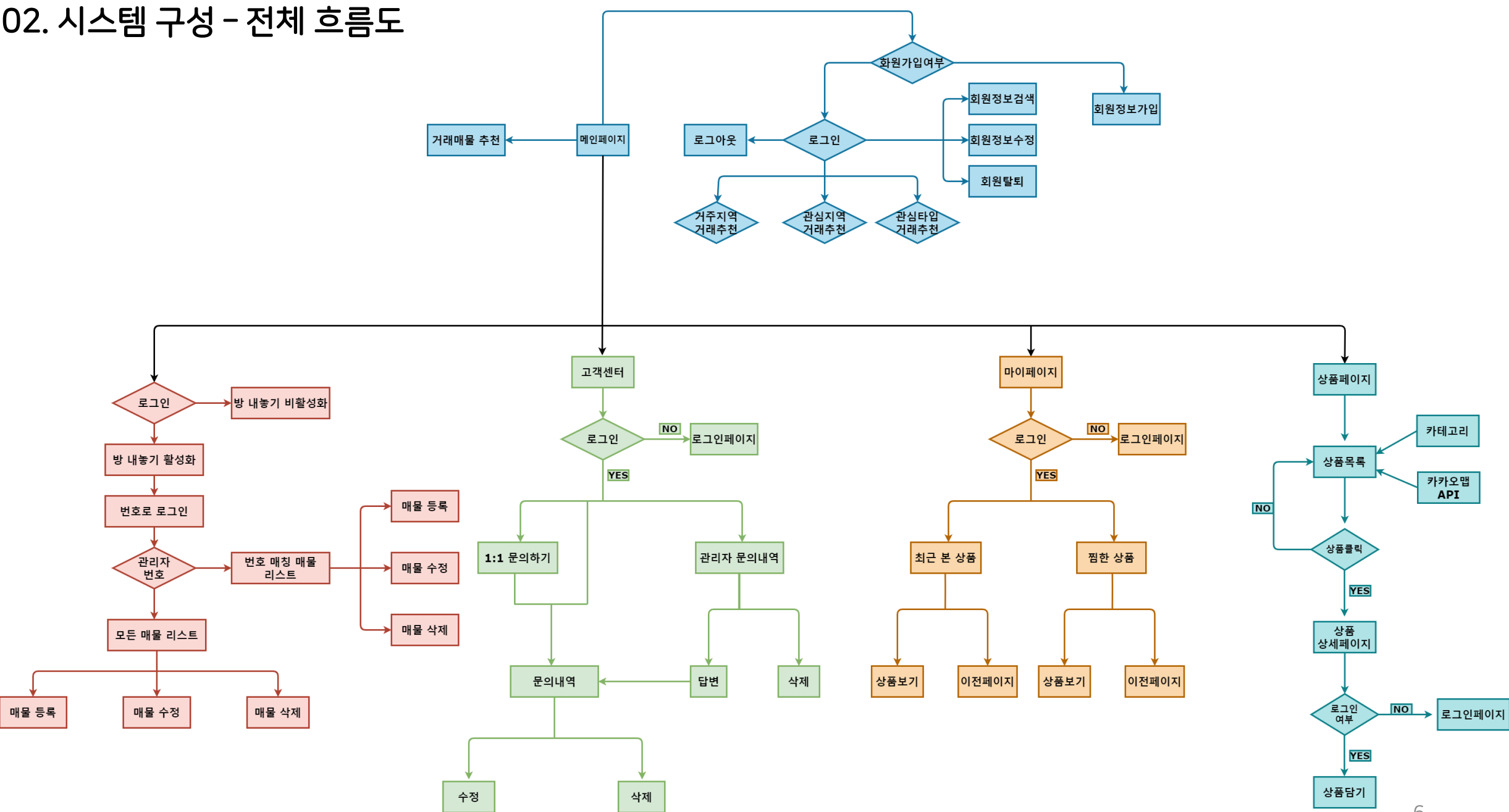
Kakao Developers Open API



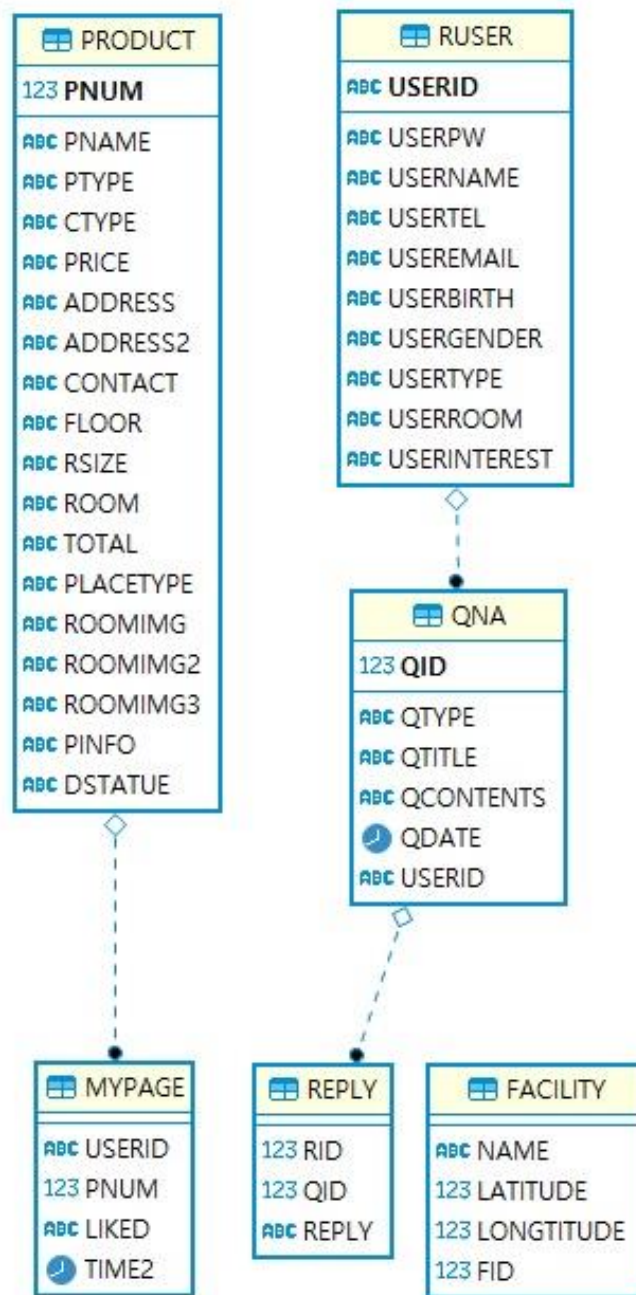
Apache Tomcat



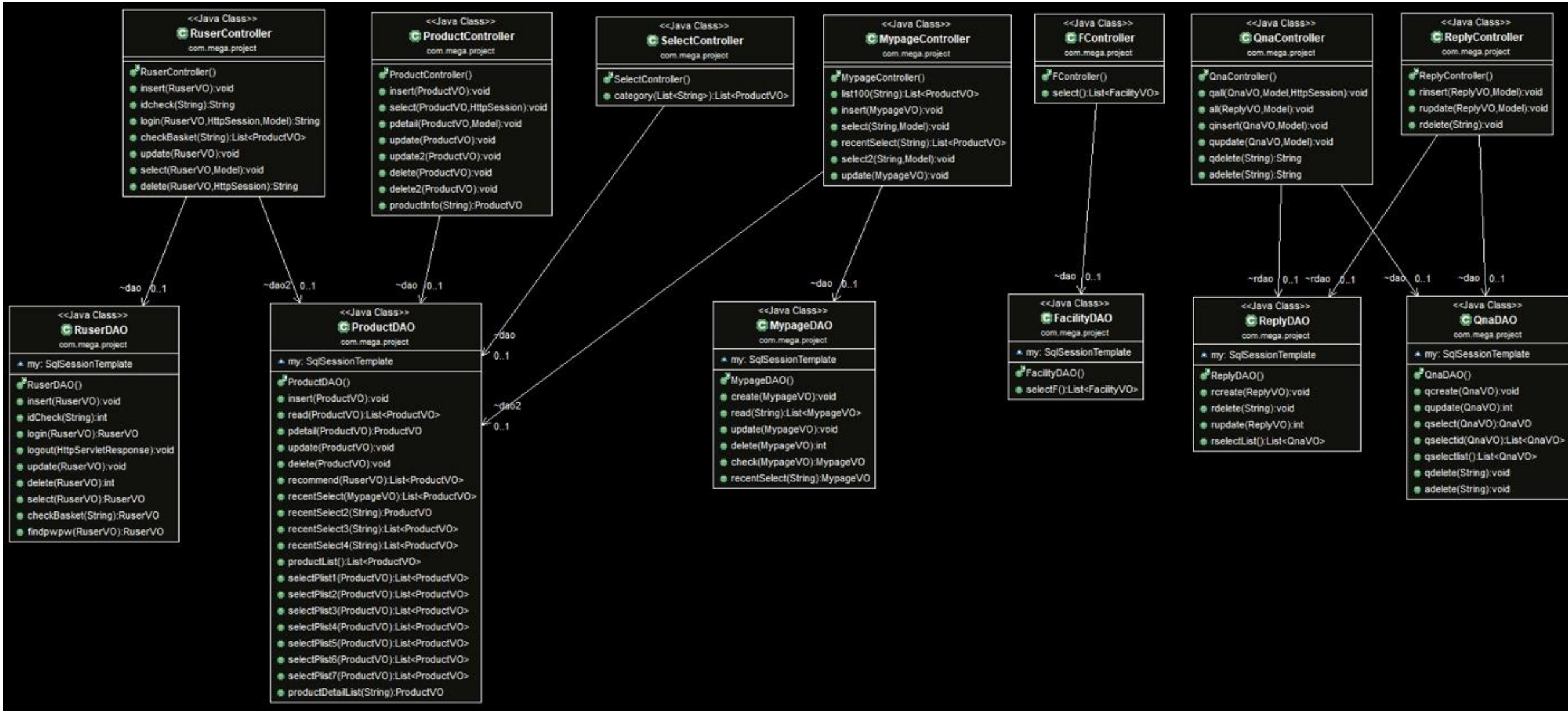
02. 시스템 구성 - 전체 흐름도



02. 시스템 구성 - 전체 ERD



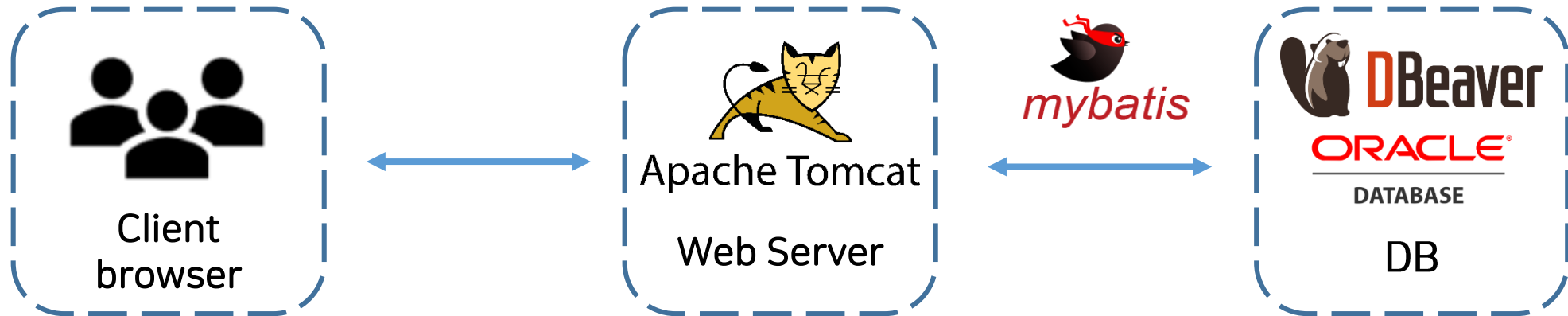
02. 시스템 구성 - 전체 UML Diagram



02. 시스템 구성 - 시스템 구성도



Development Tool



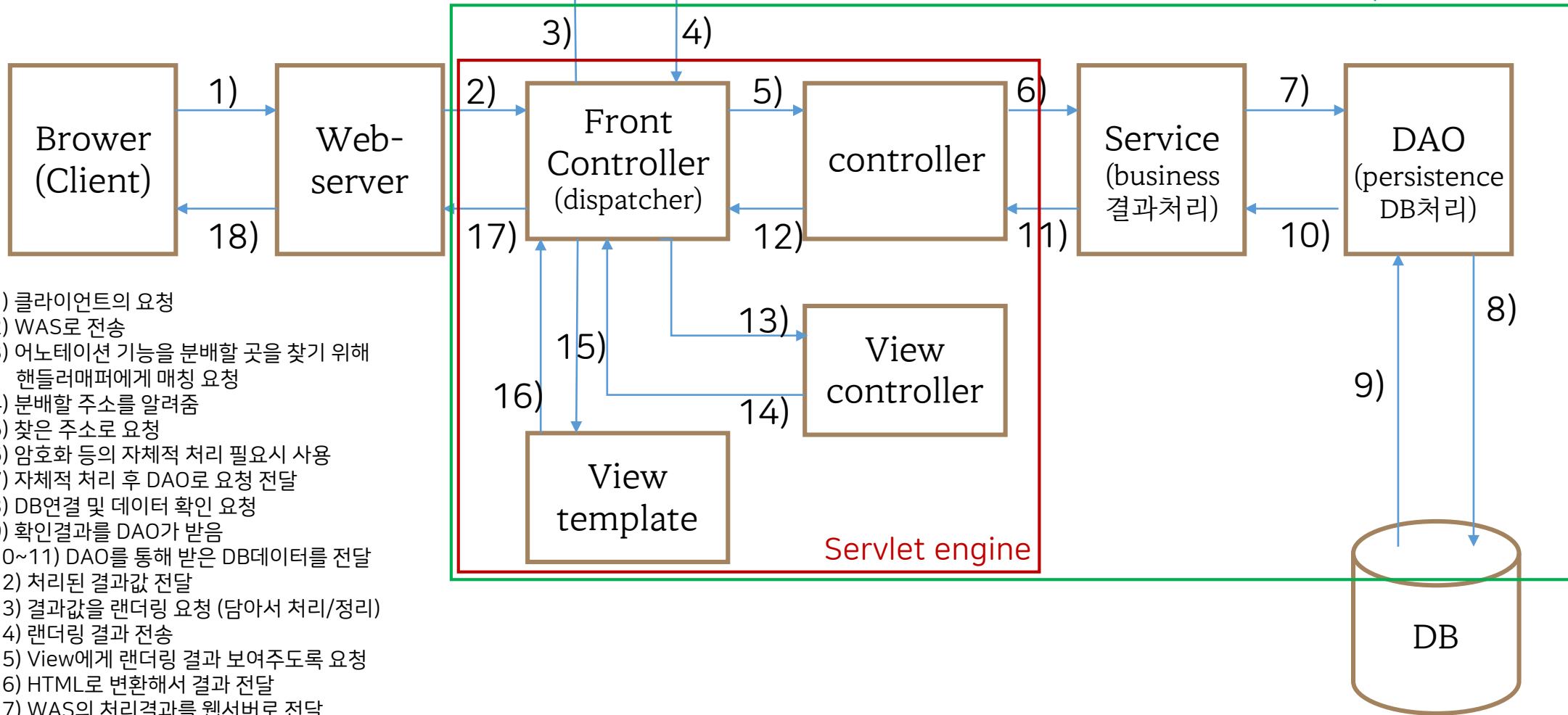
MVC2 모델 처리 과정

| | |
|--|--|
| | |
| | |
| | |

<핸들러매퍼 클래스>

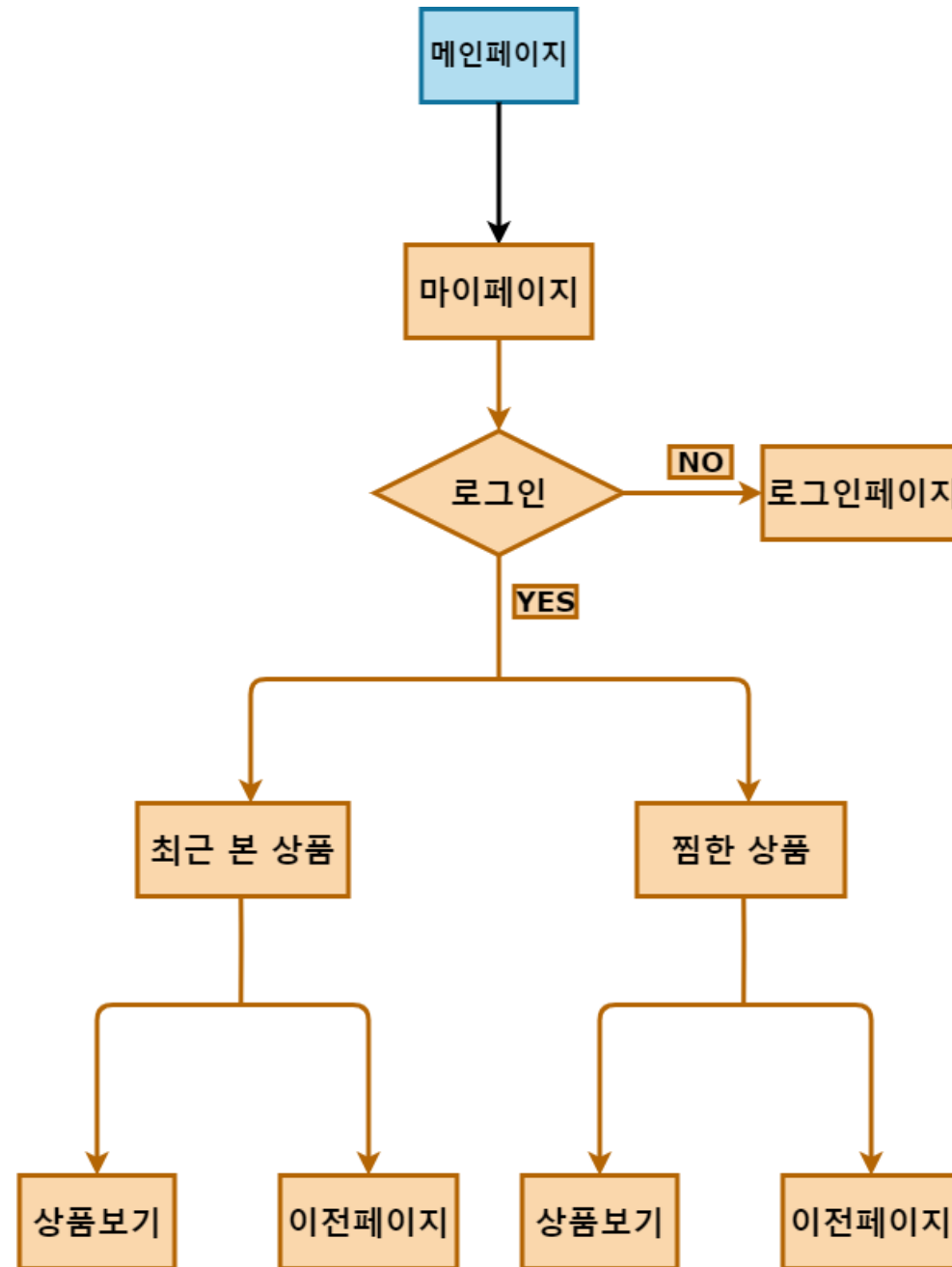
⇒ 프론트 컨트롤러가 요청 받은 주소를 처리할 객체의 메소드를 핸들러 매퍼에게 구해온다.
요청주소(key) : 컨트롤러 객체의 메소드(value)

WAS(WEB APP SERVER)



- 1) 클라이언트의 요청
- 2) WAS로 전송
- 3) 어노테이션 기능을 분배할 곳을 찾기 위해 핸들러매퍼에게 매칭 요청
- 4) 분배할 주소를 알려줌
- 5) 찾은 주소로 요청
- 6) 암호화 등의 자체적 처리 필요시 사용
- 7) 자체적 처리 후 DAO로 요청 전달
- 8) DB연결 및 데이터 확인 요청
- 9) 확인결과를 DAO가 받음
- 10~11) DAO를 통해 받은 DB데이터를 전달
- 12) 처리된 결과값 전달
- 13) 결과값을 랜더링 요청 (담아서 처리/정리)
- 14) 랜더링 결과 전송
- 15) View에게 랜더링 결과 보여주도록 요청
- 16) HTML로 변환해서 결과 전달
- 17) WAS의 처리결과를 웹서버로 전달
- 18) 최종 결과를 브라우저(클라이언트)에 보여줌

03. 담당 파트 소개 - 마이페이지



마이페이지 - DB 테이블 구성

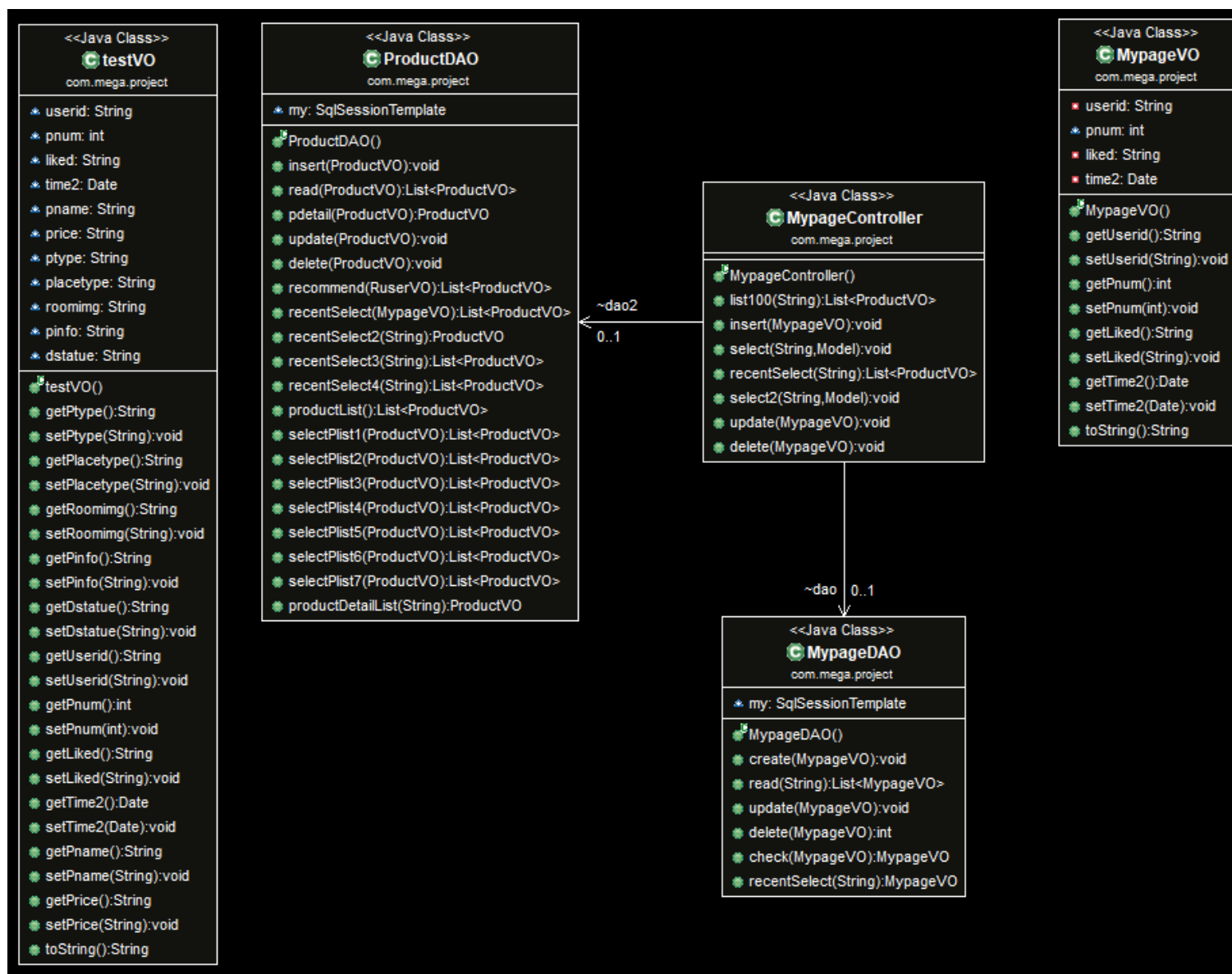
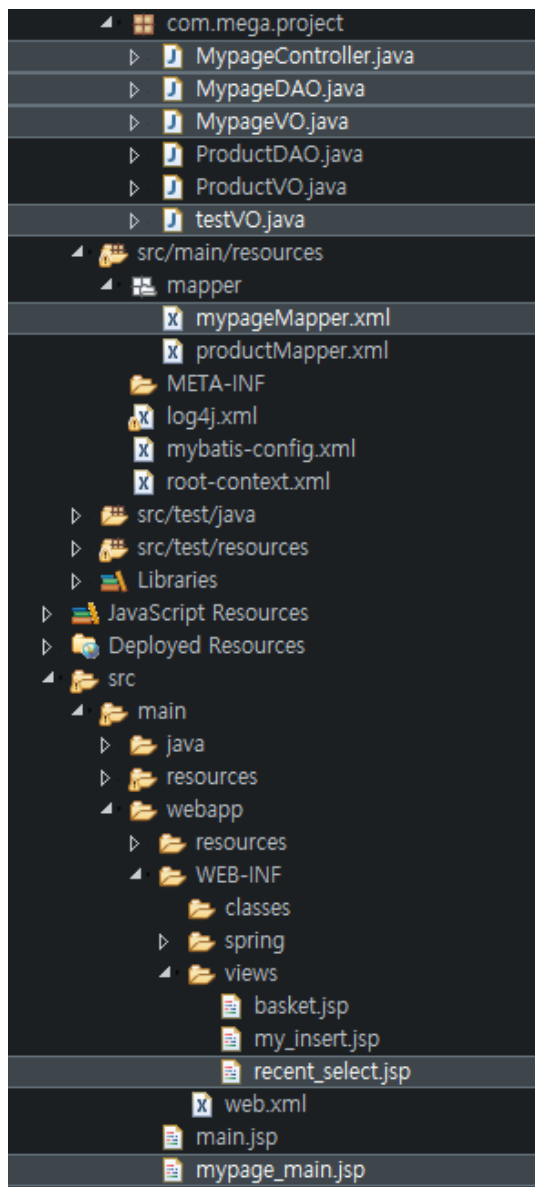
| | ABC USERID ↕ | 123 PNUM ↕ | ABC LIKED ↕ | 🕒 TIME2 ↕ |
|----|--------------|------------|-------------|---------------------|
| 1 | apple | 40 | 0 | 2021-04-19 04:44:04 |
| 2 | apple | 48 | 1 | 2021-04-19 04:40:00 |
| 3 | apple | 45 | 0 | 2021-04-19 04:38:40 |
| 4 | apple | 63 | 0 | 2021-04-19 04:30:49 |
| 5 | apple | 19 | 1 | 2021-04-19 03:13:27 |
| 6 | apple | 89 | 0 | 2021-04-19 03:12:07 |
| 7 | apple | 51 | 1 | 2021-04-16 18:43:48 |
| 8 | banana | 40 | 0 | 2021-04-16 18:41:18 |
| 9 | banana | 42 | 1 | 2021-04-16 16:11:07 |
| 10 | banana | 46 | 0 | 2021-04-16 16:05:06 |
| 11 | banana | 60 | 1 | 2021-04-16 16:04:42 |

| MYPAGE | 컬럼명 | # | Type | Type Mod | Not Null |
|------------|------------|---|--------------|----------|----------|
| ABC USERID | ABC USERID | 1 | VARCHAR2(20) | | [V] |
| 123 PNUM | 123 PNUM | 2 | NUMBER(38,0) | | [V] |
| ABC LIKED | ABC LIKED | 3 | CHAR(1) | | [V] |
| 🕒 TIME2 | 🕒 TIME2 | 4 | DATE | | [] |

```
public class MypageVO {  
    private String userid; //유저 아이디  
    int pnum; //최근본상품 (상품번호)  
    private String liked; //찜 (T/F) => (1/0)  
    private Date time2; //최근 본 상품을 시간순으로 정렬하기 위한 time
```

```
public class testVO {  
    String userid;  
    int pnum;  
    String liked;  
    java.util.Date time2; //최근 본 시간  
    String pname; //이름  
    String price; //가격  
    String ptype; //매물종류  
    String placetype; //땡세권  
    String roomimg; //메인이미지  
    String pinfo; //정보  
    String dstatue; //거래상태
```

프로젝트 트리 & UML 다이어그램



마이페이지 - 기능 상세 - 최근 본 상품 중복 체크 (최초 조회 정보만 카운트)

```
// 최근 본 상품 넣기
@RequestMapping("my_insert") //요청에 대해 어떤 컨트롤러와 메서드가 처리를 할지 매핑
//클라이언트가 가상주소;URL(my_insert)로 입력값을 보내면, 받아서 아래 메서드로 처리함.
//메서드 내에 viewName을 별도로 설정하지 않으면, URL을 viewName으로 인식
public void insert(MypageVO mypageVO) throws Exception {
    //입력값을 받아서 VO에 넣는 경우, 클래스 이름 첫글자를 소문자로 바꿔 객체를 생성하면,
    // views아래 my_insert.jsp에서 사용할 수 있다. => MypageVO mypageVO
    MypageVO check = dao.check(mypageVO); //최근 본 상품에 이미 있는지 확인
    if (check == null) { //id & pnum에 해당하는 row가 DB에 없으면(null), create실행
        dao.create(mypageVO); //입력값(mypageVO)을 dao의 create로 보냄
        System.out.println("my_insert!!");
    } else { //id & pnum에 해당하는 row가 DB에 있으면, 이미 추가된게 있으므로 아무 작업 하지않음.
        System.out.println("이미 추가됨 (중복)");
    }
}
```

MypageController.java

1) 상품 상세페이지 접근 시,
ajax를 통해 MypageController로
넘겨준 값들을 이용한다.

2) check 메서드는 로그인 된 id로
해당 상품을 조회한 적이 없으면,
(userid와 pnum이 동일한 row 존재)
create 메서드를 실행한다.

```
//중복체크(최근 본 상품 -> 이미 조회한 적 있으면, 업데이트 하지 않음)
public MypageVO check(MypageVO bag) {
    //리턴 타입이 MypageVO
    return my.selectOne("mypage.check", bag);
}
```

MypageDAO.java

```
<!-- 최근 본 상품 중복 체크 -->
<select id="check" parameterType="mypageVO" resultType="mypageVO">
    _select * from MYPAGE where userid = #{userid} and pnum = #{pnum}
</select>
```

mypageMapper.xml

```
//최근 본 상품 추가 (create)
public void create(MypageVO bag) throws Exception {
    my.insert("mypage.insert", bag);
    //myBatis의 insert 기능
    //namespace가 mypage인 매퍼(mapper)에서, id가 'insert'인 기능 수행
}
```

MypageDAO.java

```
<mapper namespace="mypage">
    <insert id="insert" parameterType="mypageVO">
        insert into MYPAGE (USERID, PNUM, LIKED, TIME2)
        values (#{userid}, #{pnum}, #{liked}, sysdate)
    </insert>
```

마이페이지 - 기능 상세 - 최근 본 상품 조회 (controller)

```
// 최근 본 상품 조회
@RequestMapping("recent_select") //요청에 대해 어떤 컨트롤러와 메서드가 처리를 할지 매핑
//클라이언트가 가상주소;URL(recent_select)로 입력값을 보내면, 받아서 아래 메서드로 처리함
public void select(String userid, Model model) throws Exception {
    System.out.println(userid); //아이디 확인

    //아이디에 해당하는 방문페이지 정보(페이지 넘버, 찜, 방문날짜) 출력
    //1. 읽어온 값(userid, pnum, liked, time2)의 리턴타입이 List이므로 mp리스트로 받아준다.
    List<MypageVO> mp = dao.read(userid);

    //2. mp리스트에 있는 pnum값들을 담은 pnumList라는 리스트 객체 생성 >> 넘버에 해당하는 상품 정보를 추출하기 위한 key
    List<String> pnumList = new ArrayList<String>(); //From List to Array

    for(int i = 0; i<mp.size(); i++) {
        pnumList.add(Integer.toString(mp.get(i).getPnum()));
    }

    //위에서 구한 pnumList를 통해 상품DAO에 접근하여 상품정보들을 출력한다.
    //3. pnumList에 들어있는 값들로 상품테이블에서 정보 가져와서
    // sp리스트에 담아준다.
    List<ProductVO> sp = new ArrayList<ProductVO>();
    for (int i = 0; i < mp.size(); i++) {
        sp.add(dao2.recentSelect2(pnumList.get(i))) ;
        //리턴값이 ProductVO
    }
}
```

MypageController.java

1) userid를 통해서 읽어온 값을 '리스트'로 받아준다. (MypageVO)

2) '리스트'에서 pnum만 뽑아서 ArrayList에 담는다. (for문)

3) pnumList를 통해 상품DAO에 접근하여, 상품data를 ArrayList로 받아온다. (ProductVO)

MypageController.java

```
//방문페이지들의 정보를 담은 리스트 생성
```

```
List<testVO> list2 = new ArrayList<testVO>();
```

```
//1. testVO타입의 변수 list를 생성하고,
```

```
// 방문페이지 정보를 담고있는 리스트 mp에서 필요한 데이터와 상품정보를 담고있는 리스트 sp에서 필요한 데이터 추출
```

```
//2. 추출한 데이터를 담고있는 bag(list)을 리스트(List<testVO> list2 = new ArrayList<testVO>();)에 담는다.
```

```
for (int i = 0; i < pnumList.size(); i++) {
```

```
    testVO list = new testVO();
```

```
    list.setUserId(mp.get(i).getUserId()); //userid
```

```
    list.setTime2((java.util.Date)mp.get(i).getTime2()); //time2
```

```
    list.setPnum(mp.get(i).getPnum()); //pnum
```

```
    list.setPname(sp.get(i).getPname());
```

```
    list.setPrice(sp.get(i).getPrice());
```

```
    list.setPtype(sp.get(i).getPtype());
```

```
    list.setPlacetype(sp.get(i).getPlacetype());
```

```
    list.setRooming(sp.get(i).getRooming());
```

```
    list.setPinfo(sp.get(i).getPinfo());
```

```
    list.setDstatue(sp.get(i).getDstatue());
```

```
    list2.add(list);
```

```
}
```

```
model.addAttribute("list2",list2); // 방문페이지들의 정보를 담고 있는 리스트를 모델로 전송
```

```
}
```

- 1) MypageVO로 가져온 값과 ProductVO로 가져온 값 중 필요한 정보들을 담은 testVO로 ArrayList 생성
- 2) 모델로 전송

마이페이지 - 기능 상세 - 최근 본 상품 조회 (dao, mapper)

```
//최근 본 상품 조회 (read) => userid에 해당하는 pnum, liked, time2
public List<MyPageVO> read(String userid){
    //리턴타입이 List
    return my.selectList("mypage.select", userid);
    //myBatis의 selectList 기능
    // namespace가 mypage인 매퍼의 id가 select인 기능 수행
}
```

MypageDAO.java

```
<!-- 최근 본 상품 조회 -->
<select id="select" parameterType="String" resultType="mypageVO">
    select * from MYPAGE where userid = #{userid}
</select>
```

mypageMapper.xml

```
public ProductVO recentSelect2(String pnum) {
    return my.selectOne("product.recentselect2", pnum);
}
```

ProductDAO.java

```
<select id="recentselect2" parameterType="String" resultType="productVO">
    select * from PRODUCT where pnum = #{pnum}
</select>
```

productMapper.xml

내 집은 신촌에 있나방

방찾기 마이페이지 방내놓기 고객센터 회원정보수정 회원탈퇴 apple님 환영합니다. 로그아웃

마이페이지

최근 본 상품 찜한 상품

apple 님의 최근 본 상품



>> 상품 보기

이름: 슈테리움 오피스텔
타입: 오피스텔
특징: 편세권
가격: 16000
거래상태: 거래가능
조회시간: Fri Apr 16 18:43:48 KST 2021



>> 상품 보기

이름: 스페이스엠
타입: 오피스텔
특징: 맥세권
가격: 10000
거래상태: 거래가능
조회시간: Mon Apr 19 04:40:00 KST 2021



>> 상품 보기

이름: (90-58)
타입: 오피스텔
특징: 역세권
가격: 14500
거래상태: 거래가능
조회시간: Mon Apr 19 04:44:04 KST 2021



>> 상품 보기

이름: 다가구
타입: 단독/다가구
특징: 스세권
가격: 5000
거래상태: 계약완료
조회시간: Mon Apr 19 03:12:07 KST 2021

이전 페이지로

마이페이지 - 기능 상세 - 최근 본 상품 조회 (view)

```
${userid} 님의 최근 본 상품 <br>
<hr color="blue">
<!-- 화면에 찍어줄 것: 상품번호를 통해 얻어온 상품 정보들 -->
<c:forEach var="list2" items="${list2}">
  <%-- 아이디: ${list2.userid} <!-- 표현언어(Expression Languages:EL), 속성프린트 --><br>
  상품번호: ${list2.pnum} --%>
  <img src = "resources/rimg/${list2.roomimg}" height = 100px>
  <a href = "productdetail.jsp?pnum=${list2.pnum}"><button style="background-color:R
이름 : ${list2.pname}<br> <!-- 상품 이름 -->
타입 : ${list2.ptype}<br> <!-- 상품 종류 -->
특징 : ${list2.placetype}<br> <!-- 땡세권 -->
가격 : ${list2.price}<br> <!-- 상품 가격 -->
거래상태 : ${list2.dstatue}<br> <!-- 거래가능/협약중/계약완료 -->
조회시간: ${list2.time2}<br> <!-- 상품 조회 시간 -->
<br>
</c:forEach>
<button onclick="goBack()" style="background-color:RGB(23, 108, 249); color: white
<script>
  function goBack() { /* 버튼을 누르면 이전 페이지로 돌아가기 */
    window.history.back();
  }
</script>
<hr color="blue">
</body>
</html>
```

```
model.addAttribute("list2",list2);
```



DMC라미안e편한세상

매매 134000

상품담기

상품상세설명

```
// 찜한상품 업데이트 (liked = 1)
@RequestMapping("basket")
public void update(MypageVO mypageVO) throws Exception {
    System.out.println("담기성공");
    dao.update(mypageVO);
}
MypageController.java
```

```
//찜한 상품 업데이트 (update)//
public void update(MypageVO bag) throws Exception {
    my.update("mypage.update", bag);
    // namespace가 mypage인 매퍼의 id가 update인 기능 수행
}
MypageDAO.java
```

```
<!-- 찜한 상품 추가 (liked = 1) -->
<update id="update" parameterType="mypageVO">
    UPDATE MYPAGE SET liked = #{liked} where userid = #{userid} AND pnum = #{pnum}
</update>
mypageMapper.xml
```

- 1) 상품 상세 페이지에 접근 시 초기값 liked = 0
- 2) "상품담기"를 클릭 시 liked = 1로 업데이트

→ 로그인 된 아이디의 해당 페이지
상품 번호에 해당하는 row 업데이트

마이페이지 - 기능 상세 - 찜한 상품 조회 (controller)

```
// 찜한 상품 조회 => 최근방문이랑 비슷한 방법
@RequestMapping("liked_select")
public void select2(String userid, Model model) throws Exception{
    System.out.println(userid);
    //아이디로 방문페이지 정보(페이지 넘버, 찜, 방문날짜) 출력
    List<MypageVO> mp = dao.read(userid); //방문페이지 정보(userid, pnum, liked, time2)를 담고

    //상품정보의 내용(아래 리스트 sp에서 pnum으로 사용됨)을 추출하기 위해 필요한 pnum값들을 담은 리스트
    List<String> pnumList = new ArrayList<String>();
    for(int i = 0; i<mp.size(); i++) {
        pnumList.add(Integer.toString(mp.get(i).getPnum()));
    }

    //위에서 구한 pnum리스트를 통해 상품DAO로 접근하여 상품정보들을 출력한다.
    List<ProductVO> sp = new ArrayList<ProductVO>();
    for (int i = 0; i < mp.size(); i++) {
        sp.add(dao2.recentSelect2(pnumList.get(i))) ; //상품페이지에 해당하는 상품정보 추출
    }
}
```

1) userid를 통해서 읽어온 값을 '리스트'로 받아준다. (MypageVO)

2) '리스트'에서 pnum만 뽑아서 ArrayList에 담는다. (for문)

3) pnumList를 통해 상품DAO에 접근하여, 상품data를 ArrayList로 받아온다. (ProductVO)

MypageController.java

마이페이지 - 기능 상세 - 찜한 상품 조회 (controller)

```
//찜한상품 확인
List<testVO> list3 = new ArrayList<testVO>();

for (int i = 0; i < pnumList.size(); i++) {
    testVO list = new testVO();
    //조건 필요!!
    //찜이 1인 상품만 담을 수 있도록 설정
    if(mp.get(i).getLiked().equals("1")) {
        list.setUserid(mp.get(i).getUserid());
        list.setTime2((java.util.Date)mp.get(i).getTime2());
        list.setPnum(mp.get(i).getPnum());
        list.setLiked(mp.get(i).getLiked());
        list.setPname(sp.get(i).getPname());
        list.setPrice(sp.get(i).getPrice());
        list.setPtype(sp.get(i).getPtype());
        list.setPlacetype(sp.get(i).getPlacetype());
        list.setRoomimg(sp.get(i).getRoomimg());
        list.setPinfo(sp.get(i).getPinfo());
        list.setDstatue(sp.get(i).getDstatue());
        list3.add(list);
    }
}
//list는 vo에서 받아올 수 없어서 model로 받아줌
model.addAttribute("list3", list3);
}
```

- 1) MypageVO로 가져온 값과 ProductVO로 가져온 값 중 필요한 정보들을 담을 testVO로 ArrayList 생성
- 2) 모델로 전송

MypageController.java

마이페이지 - 기능 상세 - 최근 본 상품 조회 (dao, mapper)

```
//찜한 상품 조회 (read) => userid에 해당하는 pnum, liked, time2
public List<MyPageVO> read2(String userid){
    //리턴타입이 List
    return my.selectList("mypage.select2", userid);
    //myBatis의 selectList 기능
    // namespace가 mypage인 매퍼의 id가 select인 기능 수행
}
```

MypageDAO.java

```
<!-- 찜한 상품 조회 -->
<select id="select2" parameterType="String" resultType="mypageVO">
    select * from MYPAGE where userid = #{userid} AND liked = '1'
</select>
```

mypageMapper.xml

```
public ProductVO recentSelect2(String pnum) {
    return my.selectOne("product.recentselect2", pnum);
}
```

ProductDAO.java

```
<select id="recentselect2" parameterType="String" resultType="productVO">
    select * from PRODUCT where pnum = #{pnum}
</select>
```

productMapper.xml

내 집은 신촌에 있나방

- 방찾기
- 마이페이지**
- 방내놓기
- 고객센터
- 회원정보수정
- 회원탈퇴
- apple님 환영합니다.
- 로그아웃

마이페이지

최근 본 상품

찜한 상품

apple 님이 찜한 상품



>> 상품 보기

이름 : 슈테리움 오피스텔
타입 : 오피스텔
특징 : 편세권
가격 : 16000
거래상태 : 거래가능
>> 서대문구에 위치한 오피스텔입니다.



>> 상품 보기

이름 : DMC라미안e편한세상
타입 : 아파트
특징 : 맥세권
가격 : 134000
거래상태 : 협의중
>> 서대문구에 위치한 아파트입니다.

| | ABC | USERID | 123 | PNUM | ABC | LIKED | TIME2 |
|----|-----|--------|-----|------|-----|-------|---------------------|
| 1 | | apple | | 40 | 0 | | 2021-04-19 04:44:04 |
| 2 | | apple | | 48 | 1 | | 2021-04-19 04:40:00 |
| 3 | | apple | | 45 | 0 | | 2021-04-19 04:38:40 |
| 4 | | apple | | 63 | 0 | | 2021-04-19 04:30:49 |
| 5 | | apple | | 19 | 1 | | 2021-04-19 03:13:27 |
| 6 | | apple | | 89 | 0 | | 2021-04-19 03:12:07 |
| 7 | | apple | | 51 | 1 | | 2021-04-16 18:43:48 |
| 8 | | banana | | 40 | 0 | | 2021-04-16 18:41:18 |
| 9 | | banana | | 42 | 1 | | 2021-04-16 16:11:07 |
| 10 | | banana | | 46 | 0 | | 2021-04-16 16:05:06 |
| 11 | | banana | | 60 | 1 | | 2021-04-16 16:04:42 |



>> 상품 보기

이름 : 스페이스엠
타입 : 오피스텔
특징 : 맥세권
가격 : 10000
거래상태 : 거래가능
>> 서대문구에 위치한 오피스텔입니다.

이전 페이지로

마이페이지 - 기능 상세 - 찜한 상품 조회 (view)

```

${userid} 님이 찜한 상품 <br>
<hr color="red">
<c:forEach var="list3" items="${list3}">
  <!-- 화면에 찍어줄 것: 상품명버를 통해 얻어온 상품 정보들 + 찜상태 + 최근 본 시간 -->
  <img src = "resources/rimg/${list3.roomimg}" height = 100px>
  <a href = "productdetail.jsp?pnum=${list3.pnum}"><button style="background-color:RGB(23, 108, 24)"
이름 : ${list3.pname}<br> <!-- 상품 이름 -->
타입 : ${list3.ptype}<br> <!-- 매물 종류 -->
특징 : ${list3.placetype}<br> <!-- 땡세권 -->
가격 : ${list3.price}<br> <!-- 가격 -->
거래상태 : ${list3.dstatue}<br> <!-- 거래가능/협약중/거래완료 등 -->
>> ${list3.pinfo}<br> <!-- 상품 한줄 설명 -->
<br>
</c:forEach>
<button onclick="goBack()" style="background-color:RGB(23, 108, 24)"
<script>
  function goBack() {
    window.history.back();
  }
</script>
<hr color="red">
<br>
</body>
</html>

```

liked_select.jsp

내 집은 신촌에 있나방

방찾기 마이페이지 방내놓기 고객센터 회원정보수정 회원탈퇴 apple님 환영합니다. 로그아웃


마이페이지

최근 본 상품

찜한 상품

| | USERID | PNUM | LIKED | TIME2 |
|----|--------|------|-------|---------------------|
| 1 | apple | 40 | 0 | 2021-04-19 04:44:04 |
| 2 | apple | 48 | 1 | 2021-04-19 04:40:00 |
| 3 | apple | 45 | 0 | 2021-04-19 04:38:40 |
| 4 | apple | 63 | 0 | 2021-04-19 04:30:49 |
| 5 | apple | 19 | 1 | 2021-04-19 03:13:27 |
| 6 | apple | 89 | 0 | 2021-04-19 03:12:07 |
| 7 | apple | 51 | 1 | 2021-04-16 18:43:48 |
| 8 | banana | 40 | 0 | 2021-04-16 18:41:18 |
| 9 | banana | 42 | 1 | 2021-04-16 16:11:07 |
| 10 | banana | 46 | 0 | 2021-04-16 16:05:06 |
| 11 | banana | 60 | 1 | 2021-04-16 16:04:42 |

apple 님이 찜한 상품




>> 상품 보기

이름: 슈테리움 오피스텔
타입: 오피스텔
특징: 편세권
가격: 16000
거래상태: 거래가능
>> 서대문구에 위치한 오피스텔입니다.



>> 상품 보기

이름: DMC래미안e편한세상
타입: 아파트
특징: 맥세권
가격: 134000
거래상태: 협의중
>> 서대문구에 위치한 아파트입니다.



>> 상품 보기

이름: 스페이스엠
타입: 오피스텔
특징: 맥세권
가격: 10000
거래상태: 거래가능
>> 서대문구에 위치한 오피스텔입니다.

이전 페이지로

04. 프로젝트 소감

첫 프로젝트에서 CRUD 방식을 이해하고 MVC1 패턴으로 구현했던 것과 달리, 이번 프로젝트에서는 MVC2 패턴을 이용했습니다.

둘의 가장 큰 차이점은 클라이언트의 요청에 따른 각 기능을 모듈화 해서 처리한다는 점인데, 단순 반복적인 코드가 줄어들고 깔끔해져 유지보수, 확장성 측면에서 MVC2가 장점을 가집니다.

그러나, 코드가 짧아져서 간단할 거라고 기대했던 것과 달리, 오히려 Model, View, Controller 등 각 모듈들이 하는 기능과 처리 순서를 고려하지 않으면, 생각지 못한 부분에서 에러가 발생했고, 에러를 찾는 것 또한 어려웠습니다.

여러 시행착오를 거치며 마이페이지 부분을 구현하면서, 스프링의 MVC 패턴 구조에 대해 점차 익숙해질 수 있었고, 프로젝트의 목적에 따라서 어떤 모델을 사용해야 하는지도 이해할 수 있었습니다.

아쉬운 점은 많은 기능을 구현하지 못했다는 점인데, 프로젝트를 마친 후, 추가할 만한 기능들을 보완하고, 팀 내 다른 인원들이 맡았던 페이지의 기능 및 코드를 분석하면서 기획단계부터 설계 - 구축까지의 전체적인 틀을 이해해 볼 예정입니다.

AWS - EC2 연결 과정

Aws - EC2 - 인스턴스 생성

인스턴스 (1) 정보

인스턴스 필터링

| Name | 인스턴스 ID | 인스턴스 상태 | 인스턴스 유형 | 상태 검사 |
|----------|---------------------|---------|----------|-------|
| ubuntu18 | i-03d7d64df5c40bb0f | 중지됨 | t2.micro | - |

'프리 티어만' 체크 후 Window2016 서버 Base 선택

1. AMI 선택
2. 인스턴스 유형 선택
3. 인스턴스 구성
4. 스토리지 추가
5. 태그 추가
6. 보안 그룹 구성
7. 검토

단계 1: Amazon Machine Image(AMI) 선택

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버, 애플리케이션)이 포함된 템플릿입니다. AWS에서 제공하는 AMI를 선택하거나, 자체 AMI 중 하나를 선택할 수도 있습니다.

검색어를 입력하여 AMI를 검색합니다. 예: 'Windows'

빠른 시작

나의 AMI

AWS Marketplace

커뮤니티 AMI

☒ 프리 티어만

Microsoft Windows Server 2016 Base - ami-0f3593d61e59c226b

Windows

프리 티어 사용 가능

Microsoft Windows 2016 Datacenter edition. [English]

루트 디바이스 유형: ebs 가상화 유형: hvm ENA 활성화: 예

64비트(x86)

선택

생성 완료 후 연결(1)

인스턴스 (1/1) 정보

인스턴스 필터링

search: i-0f451b0d712bca25d

필터 지우기

| Name | 인스턴스 ID | 인스턴스 상태 | 인스턴스 유형 | 상태 검사 | 경보 상태 |
|------|---------------------|---------|----------|-------|-------|
| - | i-0f451b0d712bca25d | 실행 중 | t2.micro | 초기화 | 경보 없음 |

암호 해독

EC2 > 인스턴스 > i-0f451b0d712bca25d > 인스턴스에 연결

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-0f451b0d712bca25d에 연결

Session Manager RDP 클라이언트 EC2 직렬 콘솔

원격 데스크톱 파일 다운로드

메시지가 표시되면 다음 세부 정보를 사용하여 인스턴스에 연결합니다.

Public DNS ec2-18-222-151-92.us-east-2.compute.amazonaws.com 사용자 이름 Administrator

암호 암호 가져오기

인스턴스를 디렉터리에 조인한 경우 디렉터리 자격 증명을 사용하여 인스턴스에 연결할 수 있습니다.

암호 암호 복사됨 edXY(*;086Z4fk.

Windows 암호 가져오기 정보

이 인스턴스에 대한 초기 Windows 관리자 암호를 검색하고 해독합니다.

암호를 해독하려면 이 인스턴스에 대한 키 페어가 필요합니다.

이 인스턴스와 연결된 키 페어 jsawskey

키 페어로 이동: Browse

jsawskey.pem 1.7KB

또는 아래 키 페어 내용을 복사하여 붙여 넣습니다.

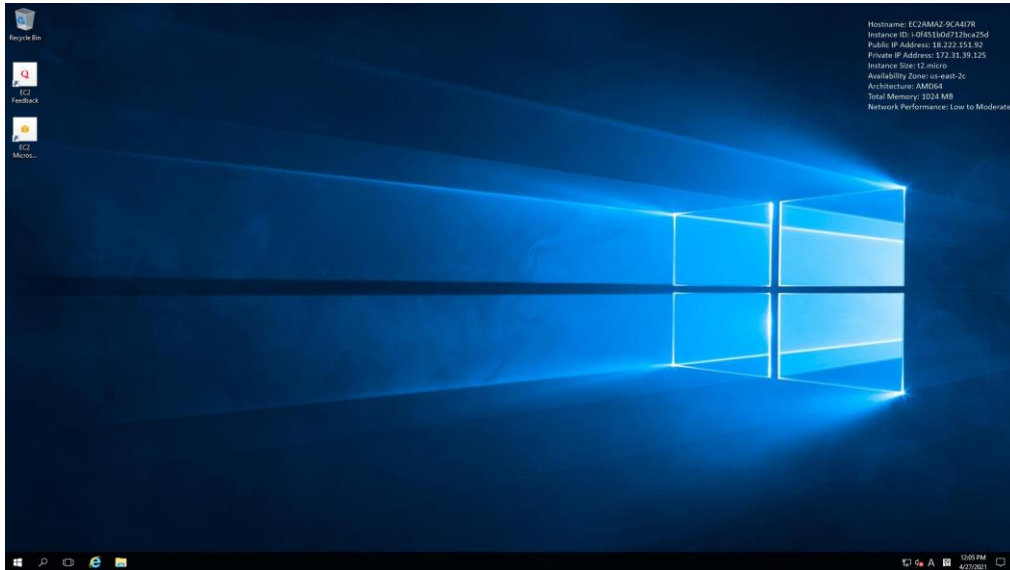
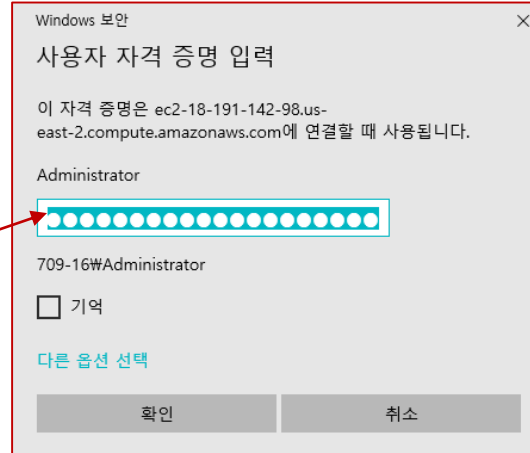
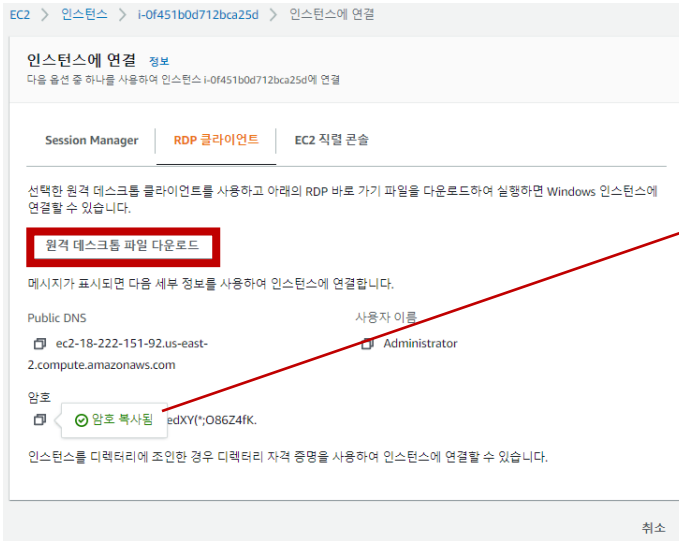
-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEA...

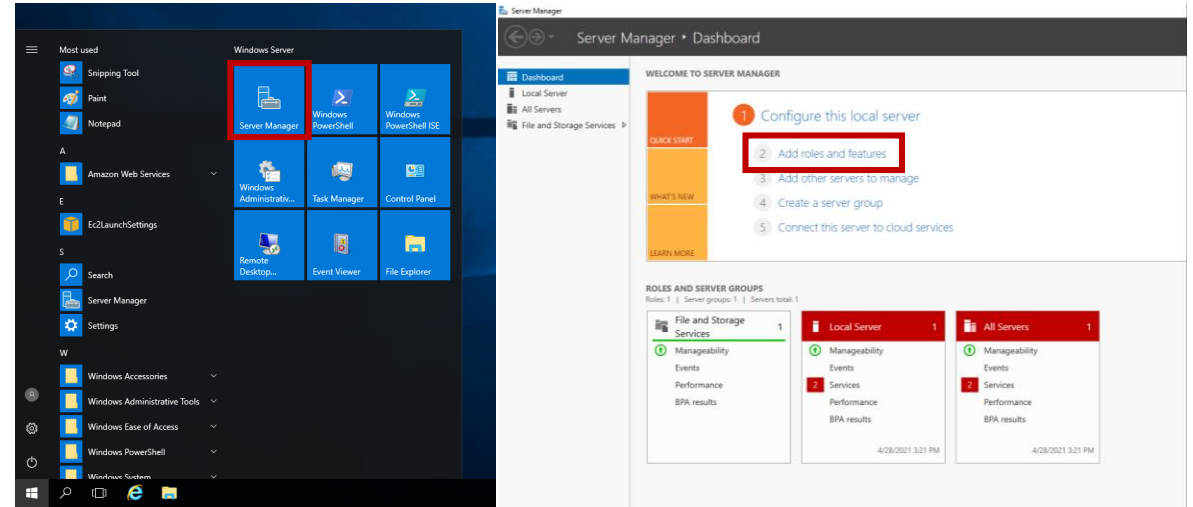
취소 암호 해독

AWS - EC2 연결 과정

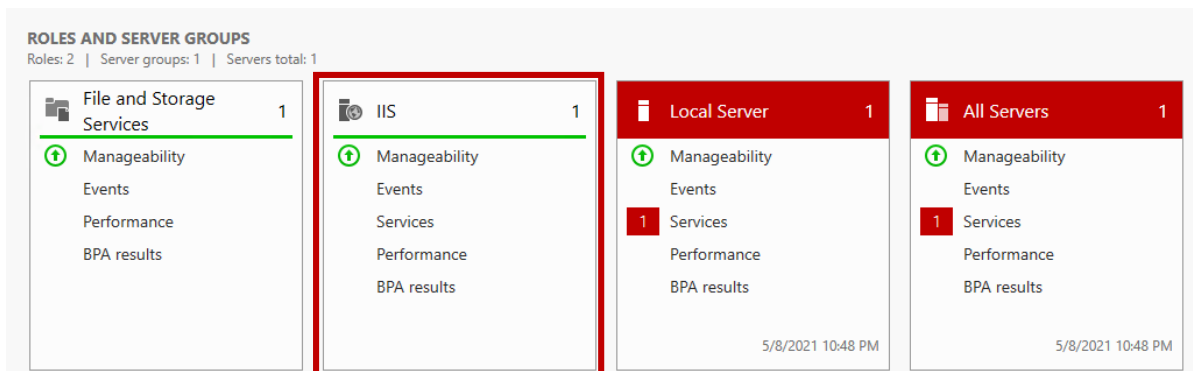
원격 데스크톱 파일 다운로드 후 실행



Server Manager 설정

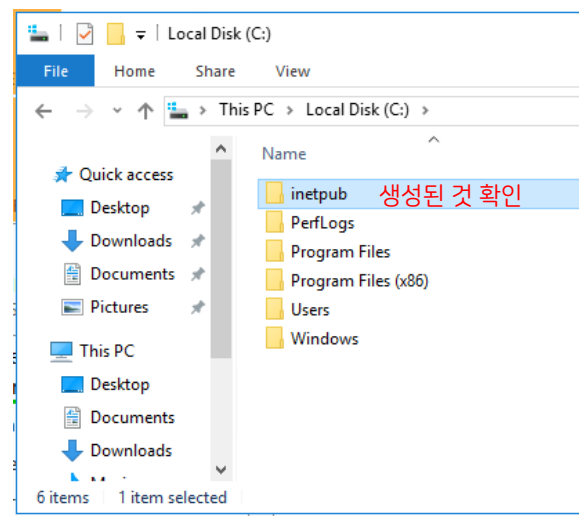
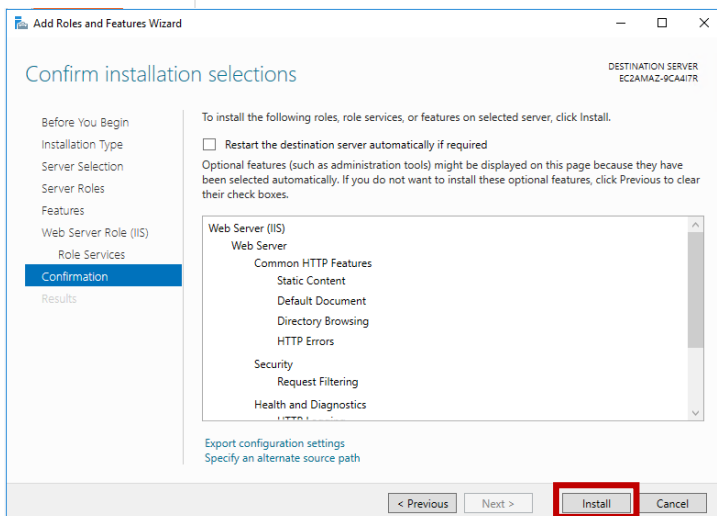
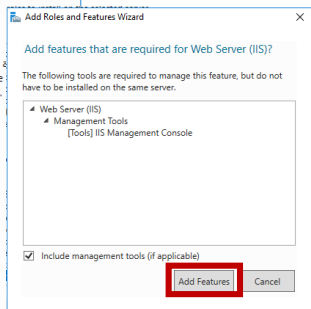
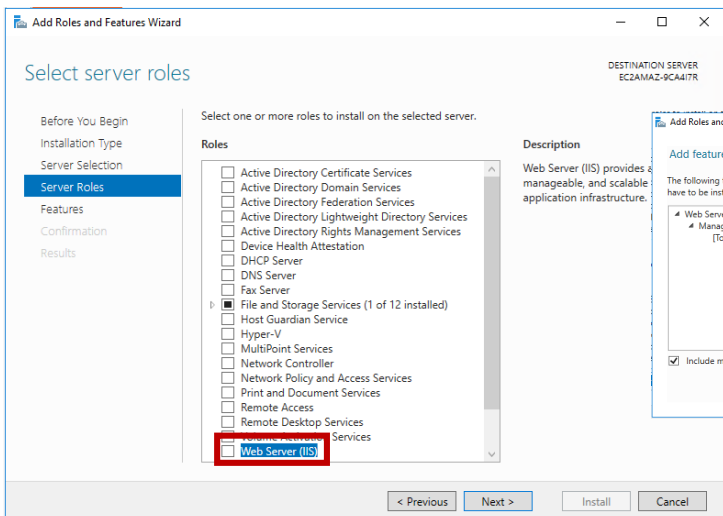
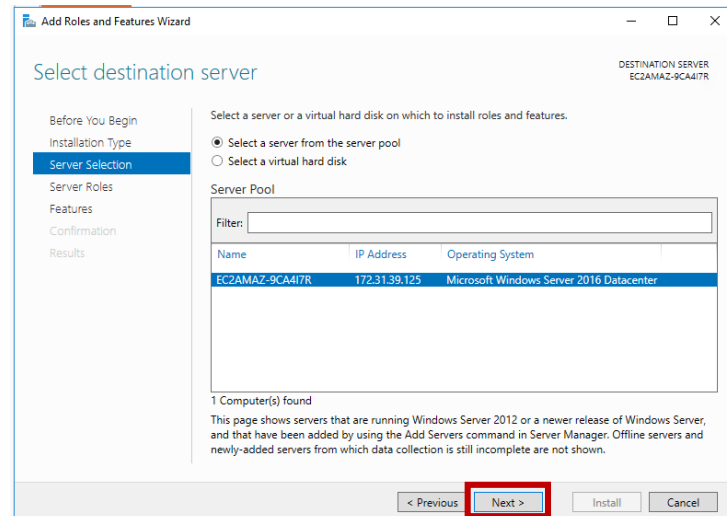
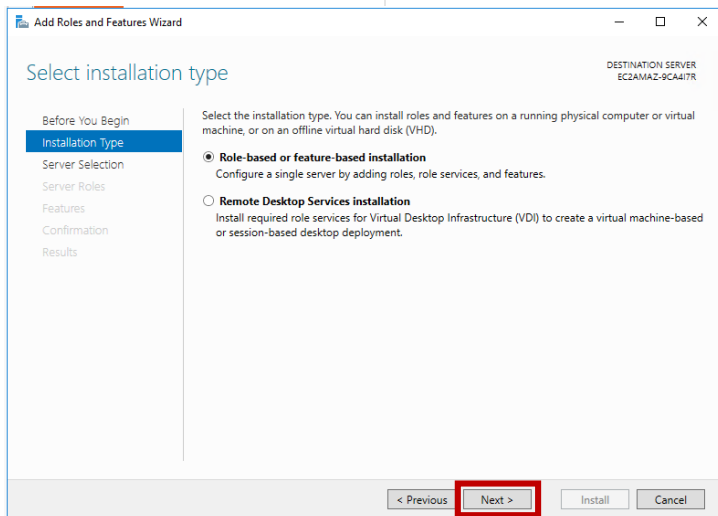
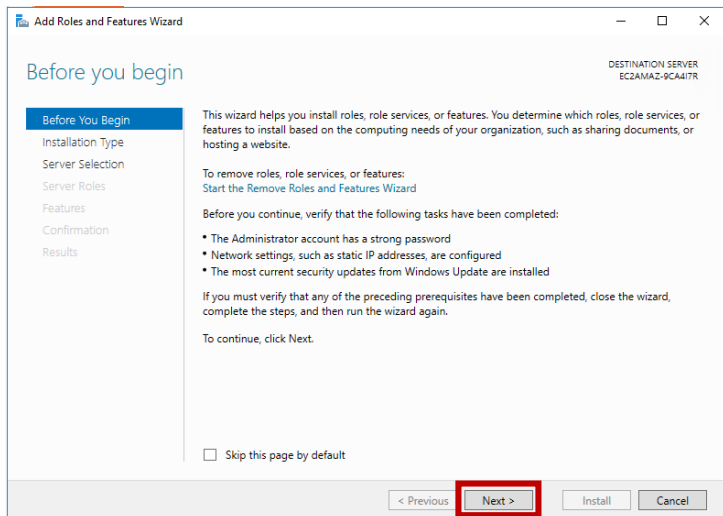


웹서버(IIS) 설치

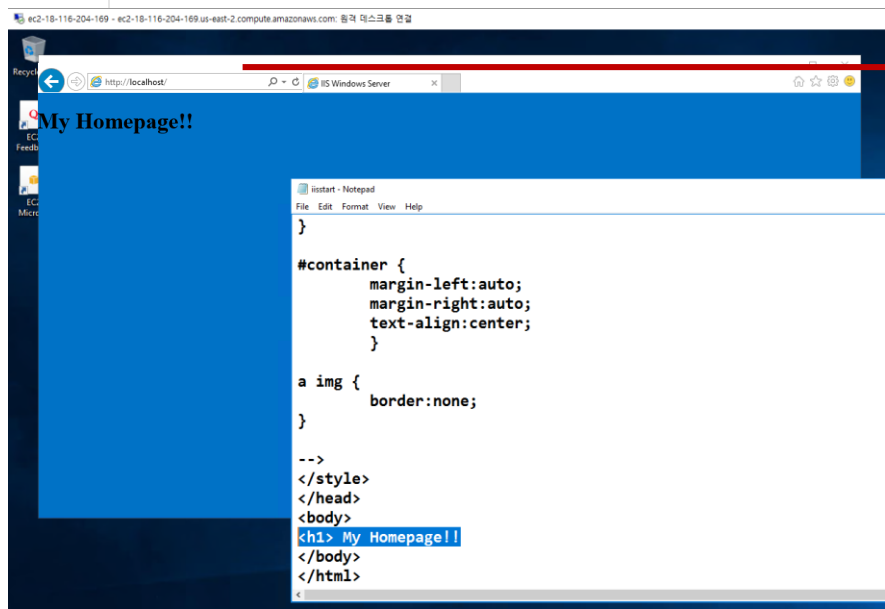
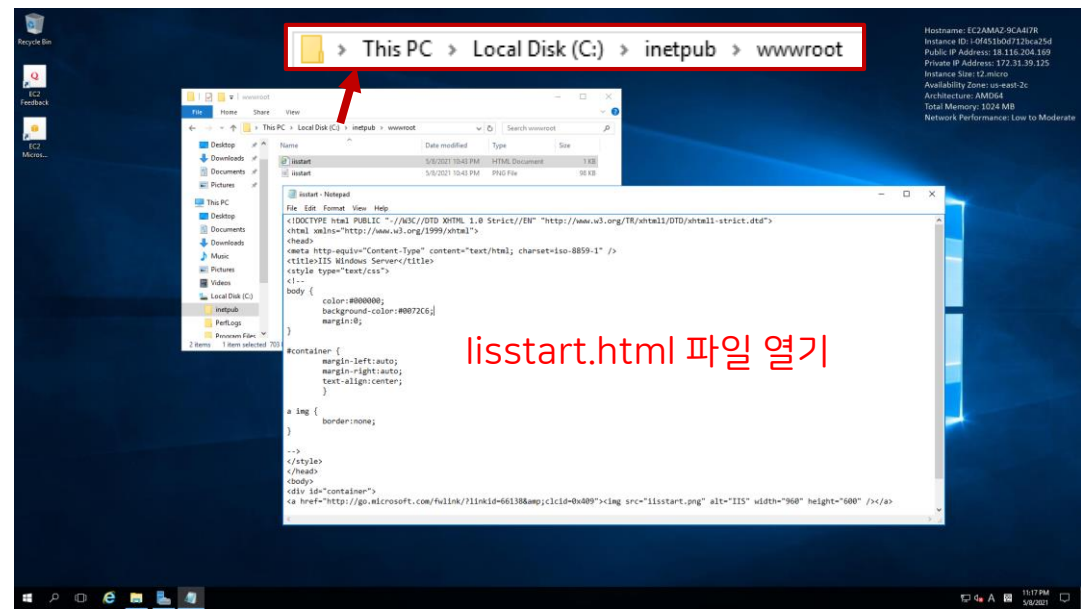


AWS - EC2 연결 과정

next → next → next → Web Server (IIS) 체크 → next → ... → Install



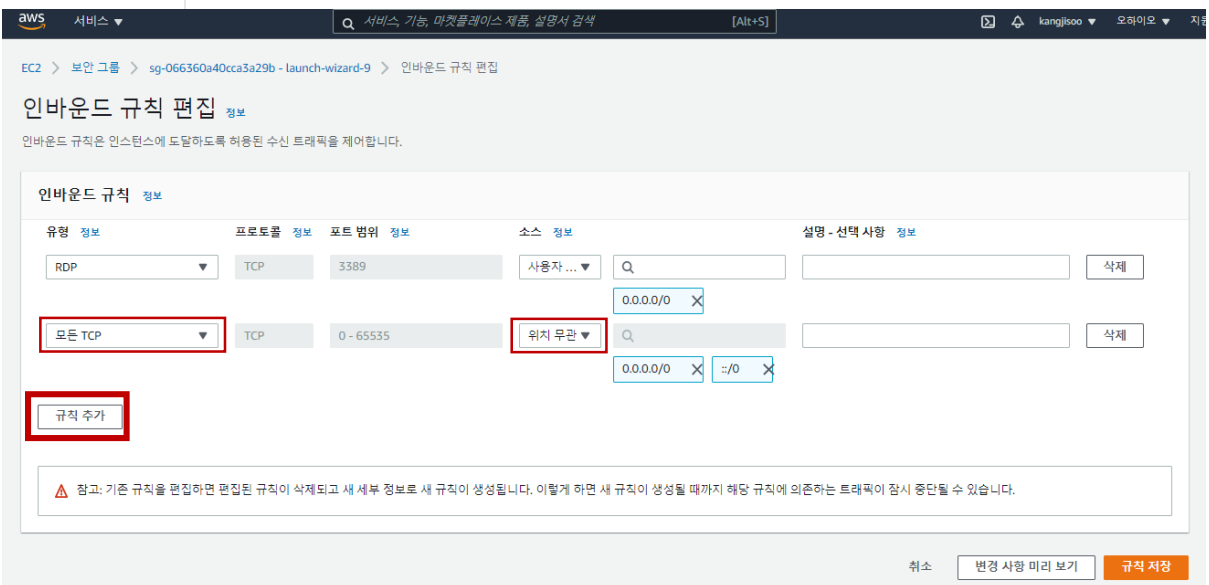
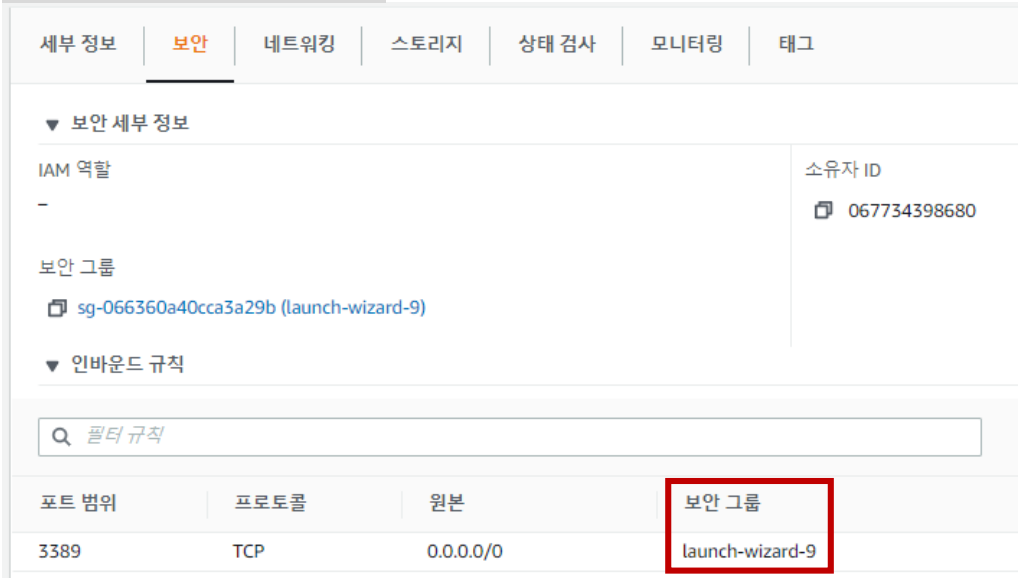
AWS - EC2 연결 과정



- Http://localhost
EC2 (오하이오에 있는 컴퓨터) 에서 접근함

- 내 컴퓨터에서
인스턴스 주소를 통해
접근 가능 하려면,
인바운드 규칙 설정 필요

인바운드 규칙 설정



AWS - EC2 연결 과정

| | Name | 인스턴스 ID | 인스턴스 상태 | 인스턴스 유형 | 상태 검사 | 경보 상태 | 가용 영역 | 퍼블릭 IPv4 DNS | 퍼블릭 IPv4 주소 |
|-------------------------------------|------------|---------------------|---------|----------|---------------|-------|------------|--|----------------|
| <input type="checkbox"/> | ubuntu18 | i-03d7d64df5c40bb0f | 중지됨 | t2.micro | - | 경보 없음 | us-east-2c | - | - |
| <input checked="" type="checkbox"/> | window2016 | i-0f451b0d712bca25d | 실행 중 | t2.micro | 2/2개 검사 통과... | 경보 없음 | us-east-2c | ec2-18-116-204-169.us-east-2.compute.amazonaws.com | 18.116.204.169 |

← → ↻ 주의 요함 | ec2-18-116-204-169.us-east-2.compute.amazonaws.com

My Homepage!!

← → ↻ 주의 요함 | 18.116.204.169

My Homepage!!

- 인바운드 규칙 설정 후, RDP (Web Server) 가 아닌 외부 서버에서도 DNS 주소를 통해 접근 가능해짐.

- *AWS (Amazon Web Services)*
- *Amazon EC2 (Amazon Elastic Compute Cloud)*

MySQL을 활용한 MVC1

게임정보조회 웹사이트 구축

강 지 수 (참여 인원 **4명**)

목 차

1. 프로젝트 개요

가. 사용 기술 및 개발 도구

나. 개발 기간

다. 흐름도 및 요약

2. 상세 구현 내용

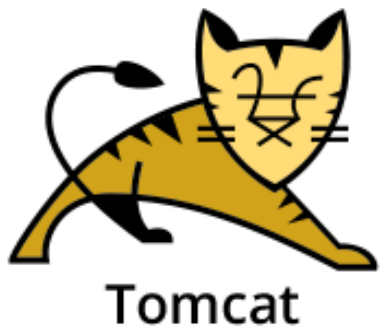
가. 챔피언 소개

나. 회원 정보 수정/탈퇴

3. 소감

가. 사용 기술 및 개발 도구

<1. 프로젝트 개요>



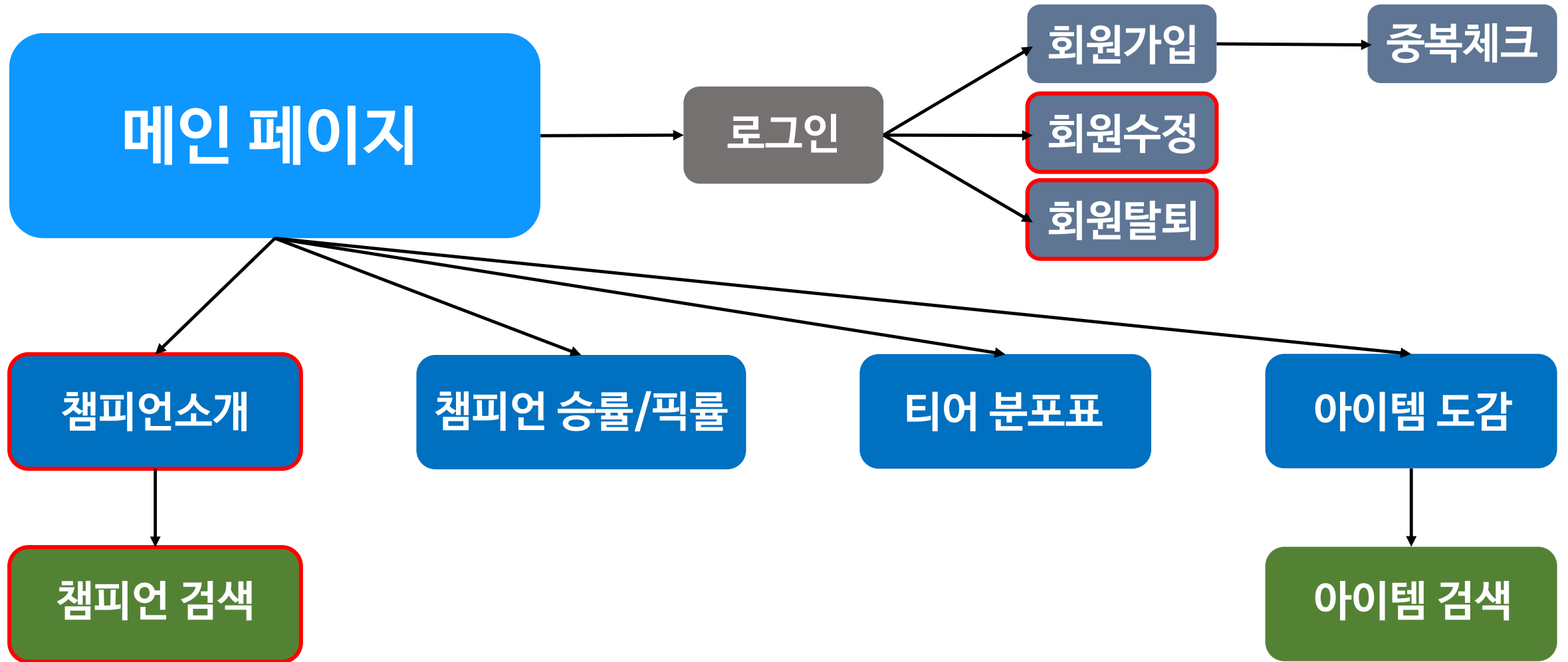
나. 개발 기간

<1. 프로젝트 개요>



다. 흐름도 및 요약

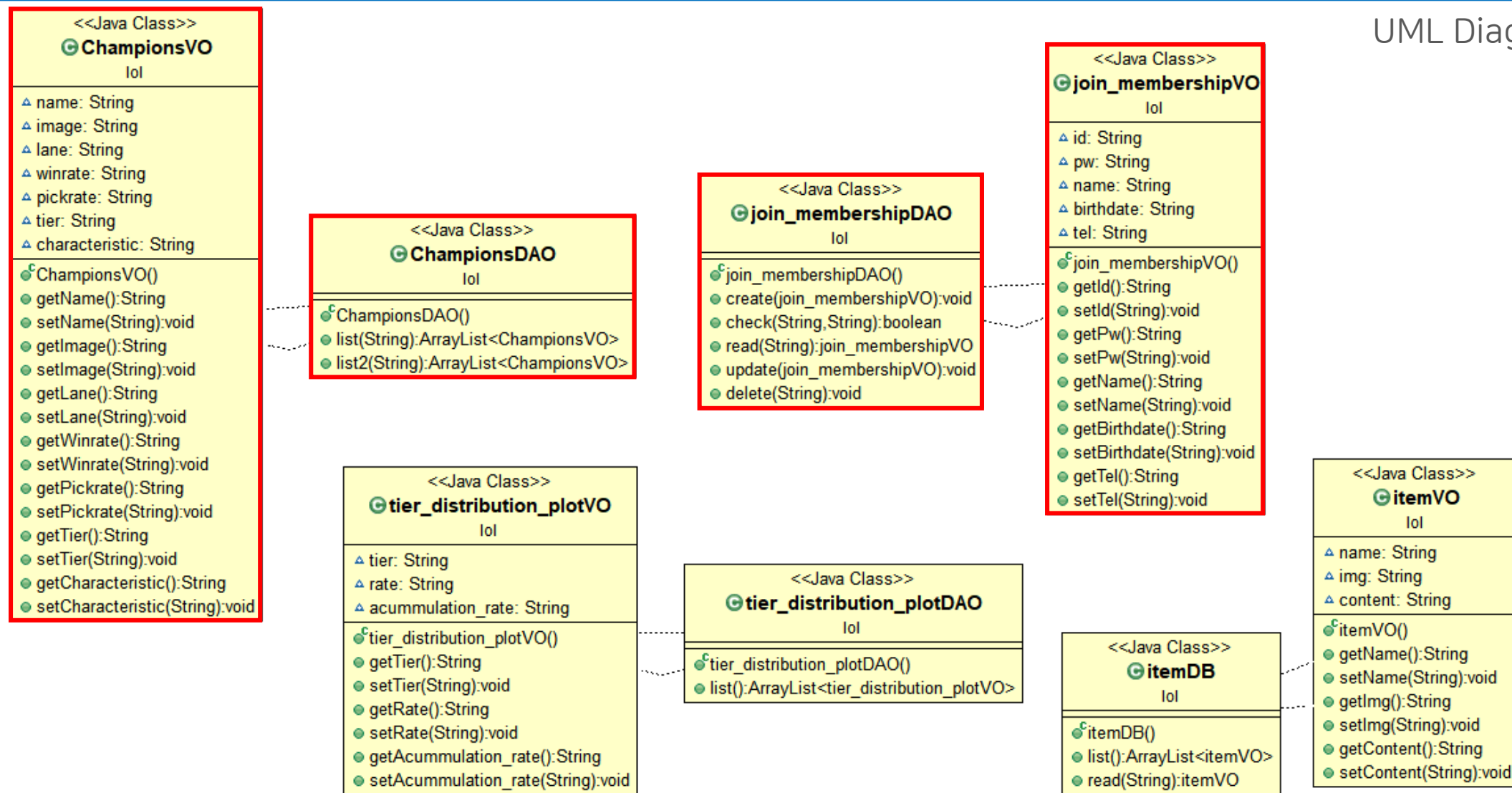
<1. 프로젝트 개요>



다. 흐름도 및 요약

<1. 프로젝트 개요>

UML Diagrams



다. 흐름도 및 요약

<1. 프로젝트 개요>

요약 → **MySQL**을 활용한 **MVC1** 게임 정보 조회 웹사이트

리그 오브 레전드(=League of Legends) 라는 게임의 DATA를 기반으로 챔피언 정보(이름, 이미지, 특성, 순위, 승률, 픽률, 티어 등)와 유저별 티어 분포, 아이템 정보(이름, 이미지, 설명 등)를 제공하는 조회 중심의 웹 사이트입니다.

가. 챔피언 소개

<2. 상세 구현 내용>

챔피언 소개 뷰

3 검색

1 챔피언 소개

2 TOP

4 로그인

상위 메뉴

로그인 및 회원 정보 관리

라인 별 분류

| Name | Image | Characteristic |
|--------|-------|------------------|
| Akali | | 암살자, 둔화, 은신 |
| Anivia | | 마법사, 서포터, 둔화, 기절 |
| Annie | | 마법사, 기절, 보호막 |

가. 챔피언 소개

<2. 상세 구현 내용>

1

상위 메뉴

- top.jsp

[챔피언 소개](#)

[챔피언 승률/픽률](#)

[유저별 티어 분포표](#)

[아이템 도감](#)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<ul>
  <li><a href="champions.jsp">챔피언 소개</a></li>
  <li><a href="rates.jsp">챔피언 승률/픽률</a></li>
  <li><a href="plot페이지.jsp">유저별 티어 분포표</a></li>
  <li><a href="items.jsp">아이템 도감</a></li>
</ul>
```

- 메인 페이지(main.jsp)에 상위 메뉴 (top.jsp) 공간을 생성.
- 각 메뉴에 해당하는 .jsp 를 하이퍼링크 (href)를 통해 연결.

2

라인 별 분류

- champions.jsp

[TOP](#)

[JG](#)

[MID](#)

[BOT](#)

[SUP](#)

```
<div id="center">
  <ul>
    <li><a href="champions.jsp?lane=top">TOP</a></li>
    <li><a href="champions.jsp?lane=jg">JG</a></li>
    <li><a href="champions.jsp?lane=mid">MID</a></li>
    <li><a href="champions.jsp?lane=bot">BOT</a></li>
    <li><a href="champions.jsp?lane=sup">SUP</a></li>
  </ul>
```

- 메인(top.jsp)에서 챔피언 소개 (champions.jsp) 메뉴 클릭
- 라인(lane)값을 매개변수로 받아 150개 챔피언을 라인별로 분류 (뒷장에 이어짐)

가. 챔피언 소개

<2. 상세 구현 내용>

2

라인 별 분류 - champions.jsp

```
// 매개변수에 입력 및 전송된 데이터를 받아온다.  
String lane = request.getParameter("lane");  
ChampionsDAO db = new ChampionsDAO(); // db 사용  
ArrayList<ChampionsVO> list = db.list(lane); // list 사용
```

- ChampionsVO.java 클래스를 통해
list 객체 생성 (getters and setters 함수 사용)

| TOP | JG | MID | BOT | SUP |
|---|----|------------------|-----|-----|
| Image | | Characteristic | | |
|  | | 암살자, 둔화, 은신 | | |
| Anivia | | 마법사, 서포터, 둔화, 기절 | | |
| Annie | | 마법사, 기절, 보호막 | | |

```
public class ChampionsVO {  
    String name;  
    String image;  
    String lane;  
    String winrate;  
    String pickrate;  
    String tier;  
    String characteristic;  
    //값을 넣을 때 사용하는 메서드 setters  
    //값을 꺼낼 때 사용하는 메서드 getters
```

ChampionsVO.java

```
    public String getName() {  
        return name;  
    }
```

```
    public void setName(String name) {  
        this.name = name;  
    }
```

```
    public String getImage() {  
        return image;  
    }
```

```
    public void setImage(String image) {  
        this.image = image;  
    }
```

```
    public String getLane() {  
        return lane;  
    }
```

```
    public void setLane(String lane) {  
        this.lane = lane;  
    }
```

```
    public String getWinrate() {  
        return winrate;  
    }
```

```
    public void setWinrate(String winrate) {  
        this.winrate = winrate;  
    }
```

```
    public void setPickrate(String pickrate) {  
        this.pickrate = pickrate;  
    }
```

```
    public String getTier() {  
        return tier;  
    }
```

```
    public void setTier(String tier) {  
        this.tier = tier;  
    }
```

```
    public String getCharacteristic() {  
        return characteristic;  
    }
```

```
    public void setCharacteristic(String characteristic) {  
        this.characteristic = characteristic;  
    }
```

가. 챔피언 소개

<2. 상세 구현 내용>

2

라인 별 분류 - ChampionsDAO.java

```
public class ChampionsDAO {  
    // 라인(lane)을 매개변수로 구분하여 답을 ArrayList 생성 => list()  
    public ArrayList<ChampionsVO> list(String lane) throws Exception {  
        ArrayList<ChampionsVO> list = new ArrayList<>();  
        // 1. 커넥터 사용하도록 설정  
        Class.forName("com.mysql.jdbc.Driver");  
        // 2. DB 연결 및 한글 사용 설정 => lol, root, 1234  
        String url = "jdbc:mysql://localhost:3306/lol?useUnicode=true&characterEncoding=utf8";  
        Connection con = DriverManager.getConnection(url, "root", "1234");  
        // 3. SQL문 생성 -> 라인으로 검색한 후, 이를 오름차순으로 정렬하여 가져오기.  
        String sql = "select name, image, characteristic from champions where lane = ? order by name";  
        PreparedStatement ps = con.prepareStatement(sql);  
        ps.setString(1, lane);  
        // 4. SQL문을 MySQL서버로 전송함.  
        ResultSet rs = ps.executeQuery();  
        // 1. 검색결과가 있는지 체크.  
        System.out.println(rs.next());  
        // 있으면 true, 없으면 false  
        // rs.next()를 호출할 때 마다 한 행씩 내려가서 그 행이 존재하는지 체크 : boolean
```

// 매개변수에 입력 및 전송된 데이터를 받아온다.

String lane = request.getParameter("lane");

ChampionsDAO db = new ChampionsDAO(); // db 사용

ArrayList<ChampionsVO> list = db.list(lane); // list 사용

- ChampionsDAO.java 클래스를 통해
db 객체 생성체크하도록 구현함.

- ChampionsDAO.java 클래스를 통해
db 객체 생성
- ChampionsDAO.java 클래스는
DB 연결 > SQL 생성 > SQL문을 MySQL
서버로 전송 > setString()과 ps.exe
cuteQurey()함수를 통해 입력값(lane)
이 DB에 존재하는지 체크하도록 구현함.

가. 챔피언 소개

<2. 상세 구현 내용>

2

라인 별 분류

- ChampionsDAO.java

// 2. 만약에 결과 true이면, 테이블에서 원하는 데이터 항목의 값들을 꺼낸다.

이름으로 오름차순 (기본값)

```
while (rs.next()) { // DB에서 "select name, image, characteristic from champions where lane = ? order by name";
```

// 꺼내주는 방법은 2가지, 1) 항목명, 2) 항목의 인덱스 필요한 값의 컬럼명

```
String name2 = rs.getString("name");
```

```
String image = rs.getString("image");
```

```
String characteristic = rs.getString("characteristic");
```

// 꺼내준 값들을 bag에 넣는다

```
ChampionsVO bag = new ChampionsVO();
```

```
bag.setName(name2);
```

```
bag.setImage(image);
```

```
bag.setCharacteristic(characteristic);
```

// bag들을 리스트에 넣어준다.

```
list.add(bag);
```

```
}
```

```
return list;
```

- rs.next()의 결과가 true이면, DB에서 해당 데이터 row 중 필요한 값을 list로 반환한다.

```
<table border="1" style="width: 100%">
```

```
<tr>
```

```
<th>Name</th>
```

```
<th width="10">Image</th>
```

```
<th>Characteristic</th>
```

```
</tr>
```

```
<% // 리스트 사이즈 = row 수 = 테이블의 행 수 만큼 반복.
```

```
for (int i = 0; i < list.size(); i++) {
```

```
ChampionsVO bag = list.get(i); //가방 꺼내기
```

```
%>
```

```
<tr><!-- 가방에서 보여줄 컬럼(값)들 꺼내기 -->
```

```
<td><%=bag.getName()%></td>
```

```
<td></td>
```

```
<td><%=bag.getCharacteristic()%></td>
```

```
</tr>
```

```
<%
```

```
} //for문 끝.
```

```
%>
```

```
</table>
```

- champions.jsp

- Champions.jsp 의 <body>태그 안에 테이블을 생성
- for문을 사용해 list로 반환된 컬럼들을 꺼내어 테이블로 보여줌
- 즉, 각 라인에 해당하는 챔피언의 이름/사진/특성을 테이블에 표시

가. 챔피언 소개

<2. 상세 구현 내용>

3

검색

- champions.jsp

[검색] :

챔피언 특성 검색

```
<form action="search.jsp">
  [검색] : <input name="characteristic"><br>
  <button type="submit">챔피언 특성 검색</button>
</form>
```

- search.jsp

// 매개변수에 입력 및 전송된 데이터를 받아온다.

```
String characteristic = request.getParameter("characteristic");
```

```
ChampionsDAO db = new ChampionsDAO(); //db 사용
```

```
ArrayList<ChampionsVO> list2 = db.list2(characteristic);
```

- ChampionsDAO.java 클래스를 통해 db 객체 생성체크하도록 구현함.
- ChampionsVO.java 클래스를 통해 list2 객체 생성 (getters and setters 함수 사용)

- 챔피언 소개 페이지(champions.jsp)에 input 태그와 button 태그를 통해 특성을 검색하는 기능 생성
- 하이퍼링크 (href)를 통해 search.jsp 연결.

<h2>챔피언 [특징] 검색 결과</h2>

```
<form action="champions.jsp">
  <button>챔피언 전체 목록으로 돌아가기</button>
  <br> <br> <br>
</form>
```

- 입력값인 characteristic
을 포함하는 rows만 검색됨.

챔피언 [특징] 검색 결과

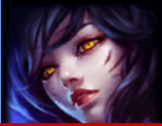

챔피언 전체 목록으로 돌아가기

TOP

JG

MID

BOT

| Name | Image | Characteristic |
|--------|---|----------------------|
| Ahri |  | 마법사, 암살자, 탈출, 치유, 도발 |
| Anivia |  | 마법사, 서포터, 둔화, 기절 |

```
"select name, image, characteristic from champions where characteristic like '%" + characteristic + "%'";
```

SQL문 (입력값을 '포함하는' 행 검색)

MySQL을 활용한 MVC1 게임정보

가. 챔피언 소개

<2. 상세 구현 내용>

3

검색

- champions.jsp

[검색] :

챔피언 특성 검색

```
// 특징(characteristic)을 매개변수로 구분하여 답을 ArrayList 생성 => list2()
public ArrayList<ChampionsVO> list2(String characteristic) throws Exception {
    ArrayList<ChampionsVO> list2 = new ArrayList<>();
    // 1. 커넥터 사용 설정
    Class.forName("com.mysql.jdbc.Driver");
    // 2. 디비연결 및 한글사용 설정
    String url = "jdbc:mysql://localhost:3306/lol?useUnicode=true&characterEncoding=utf8";
    Connection con = DriverManager.getConnection(url, "root", "1234");
    // 3. SQL문 생성
    String sql = "select name, image, characteristic from champions where characteristic like '%" + characteristic + "%'";
    PreparedStatement ps = con.prepareStatement(sql);
    // 4. SQL문 서버로 전송
    ResultSet rs = ps.executeQuery();
```

챔피언 [특징] 검색 결과

챔피언 전체 목록으로 돌아가기

- ChampionsDAO.java 클래스는 커넥터를 사용하여
DB 연결 > SQL 생성 > SQL문을 MySQL 서버로 전송
> setString()과 ps.executeQuery()함수를 통해
입력값(lane) 이 DB에 존재하는지 체크하도록 구현함.

가. 챔피언 소개

<2. 상세 구현 내용>

3 검색 - ChampionsDAO.java - search.jsp

```
while (rs.next()) {  
    // 꺼내주는 방법은 2가지, 1)항목명, 2)항목의 인덱스  
    String name2 = rs.getString("name");// "apple"  
    String image = rs.getString("image");  
    String characteristic2 = rs.getString("characteristic");  
  
    // bag2에 한 행씩 넣는다.  
    ChampionsVO bag2 = new ChampionsVO();  
    bag2.setName(name2);  
    bag2.setImage(image);  
    bag2.setCharacteristic(characteristic2);  
  
    // bag2들을 list에 넣어준다.  
    list2.add(bag2);  
}
```

- rs.next()의 결과가 true이면, DB에서 해당 데이터 row 중 필요한 값을 list2로 반환한다.

- search.jsp 의 <body>태그 안에 테이블을 생성
- for문을 사용해 list2로 반환된 컬럼들을 꺼내어 테이블로 보여줌
- 즉, 입력한 특성을 포함하는 챔피언의 이름/사진/특성을 테이블에 표시

```
<table border="1" style="width: 100%">  
    <tr>  
        <th>Name</th>  
        <th width="10">Image</th>  
        <th>Characteristic</th>  
    </tr>  
    <% // 리스트2 사이즈 = 가방 수 = 테이블의 행 수 만큼 반복.  
        for (int i = 0; i < list2.size(); i++) {  
            ChampionsVO bag3 = list2.get(i); //가방 꺼내기  
            %>  
            <tr><!-- 가방에서 보여줄 컬럼(값)들 꺼내기 -->  
                <td><%=bag3.getName()%></td>  
                <td></td>  
                <td><%=bag3.getCharacteristic()%></td>  
            </tr>  
            <%  
                } // for문 끝.  
            %>  
        </table>  
  
"select name, image, characteristic from champions  
where characteristic like '%" + characteristic + "%'";
```

나. 회원 정보 수정 / 탈퇴

<2. 상세 구현 내용>

4

로그인 및 회원 정보
관리

- memberU.jsp

회원정보수정화면입니다.

아이디:
패스워드:
전화번호:

- db연결_U.jsp

//입력해서 전송된 데이터를 받아야 한다.

```
String id = request.getParameter("id");  
String pw = request.getParameter("pw");  
String tel = request.getParameter("tel");
```

// VO클래스를 통해 bag 객체 생성

```
join_membershipVO bag = new join_membershipVO();  
bag.setId(id);  
bag.setPw(pw);  
bag.setTel(tel);
```

//DAO클래스를 통해 db 객체 생성 -> 입력값 db에 넣음 update

```
join_membershipDAO db = new join_membershipDAO(),  
db.update(bag);
```

abc님의 패스워드가 1004angel,
전화번호가 01199999999(으)로
수정이 완료되었습니다.

```
<h2>회원정보수정화면입니다.</h2>  
<hr color="blue">  
<!-- form태그는 값을 입력하고 싶을 때 사용 -->  
<!-- 입력값들은 모두 form들어가야 한다. -->  
<!-- action: 입력값을 받아서 처리하는 다음 페이지를 지정 -->  
<form action="db연결_U.jsp">  
아이디: <input name="id"><br>  
패스워드: <input name="pw"><br>  
전화번호: <input name="tel"><br>  
<button>저장 하기</button>  
</form>
```

변경 전 join_membership 테이블 DATA

| | id | pw | name | birthdate | tel |
|---|-----|--------|------|-----------|-------------|
| 1 | abc | 232323 | 가나다 | 991106 | 01012345678 |
| 2 | def | 151515 | 마바사 | 760508 | 01098765432 |

변경 후 join_membership 테이블 DATA

| | id | pw | name | birthdate | tel |
|---|-----|-----------|------|-----------|-------------|
| 1 | abc | 1004angel | 가나다 | 991106 | 01199999999 |
| 2 | def | 151515 | 마바사 | 760508 | 01098765432 |

나. 회원 정보 수정 / 탈퇴

<2. 상세 구현 내용>

4

로그인 및 회원 정보
관리

- join_membershipVO.java

```
public class join_membershipVO {  
    String id;  
    String pw;  
    String name;  
    String birthdate;  
    String tel;  
}
```

```
public String getId() {  
    return id;  
}  
public void setId(String id) {  
    this.id = id;  
}
```

```
public String getPw() {  
    return pw;  
}  
public void setPassword(String pw) {  
    this.pw = pw;  
}
```

```
public String getName() {  
    return name;  
}  
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getBirthdate() {  
    return birthdate;  
}
```

```
public void setBirthdate(String birthdate) {  
    this.birthdate = birthdate;  
}
```

```
public String getTel() {  
    return tel;  
}  
public void setTel(String tel) {  
    this.tel = tel;  
}
```

VO클래스 → 변수를 생성하고, getters and setters 함수 정의

```
// 회원정보 수정을 위한 U  
public void update(join_membershipVO bag) throws Exception { // 매개변수(입력값) -> bag  
    // 1. (다운받은)커넥터 사용하겠다고 설정  
    Class.forName("com.mysql.jdbc.Driver"); // Driver(클래스는 대문자)  
    // 2. db 연결 - lol(db) , root(id), 1234(pw)  
    String url = "jdbc:mysql://localhost:3306/lol"; // jdbc:mysql://ip:0000/db명  
    // url 너무 길어서 변수로 따로 뺌.  
    Connection con = DriverManager.getConnection(url, "root", "1234");  
    // 반환값이 있어서 con (참조형)변수에 넣음  
    // 3. SQL문을 만든다  
    String sql = "update join_membership set pw = ?, tel = ? where id = ?";  
    PreparedStatement ps = con.prepareStatement(sql);  
    ps.setString(1, bag.getPw());  
    ps.setString(2, bag.getTel());  
    ps.setString(3, bag.getId());  
    // 4. SQL문을 MySQL서버로 전송  
    ps.executeUpdate();  
}
```

- join_membershipDAO.java

- 1) jdbc 커넥터 사용
- 2) lol db에 연결
- 3) SQL문 생성
→ 입력한 id에 해당하는 row의 pw, tel 수정.
- 4) SQL문을 MySQL 서버로 전송

나. 회원 정보 수정 / 탈퇴

<2. 상세 구현 내용>

4

로그인 및 회원 정보
관리

- memberD.jsp

로그인

아이디:

패스워드:

로그인

회원가입

회원정보수정

회원탈퇴

메인으로

회원탈퇴화면입니다.

아이디: abc

회원 탈퇴 하기

- db연결_D.jsp

//입력해서 전송된 데이터를 받아야 한다.

```
String id = request.getParameter("id");
```

//DAO클래스를 통해 db 객체 생성 -> 입력받은 값(id)에 해당하는

//row 찾아서 삭제(delete)

```
join_membershipDAO db = new join_membershipDAO();  
db.delete(id);
```

회원탈퇴가 완료되었습니다.

```
<h2>회원탈퇴화면입니다.</h2>
```

```
<hr color="blue">
```

```
<!-- form태그는 값을 입력하고 싶을 때 사용 -->
```

```
<!-- 입력값들은 모두 form들어가야 한다. -->
```

```
<!-- action: 입력값을 받아서 처리하는 다음 페이지를 지정 -->
```

```
<form action="db연결_D.jsp">
```

```
아이디: <input name="id"><br>
```

```
<button>회원 탈퇴 하기</button>
```

```
</form>
```

변경 전 join_membership 테이블 DATA

| | id | pw | name | birthdate | tel |
|---|-----|-----------|------|-----------|-------------|
| 1 | abc | 1004angel | 가나다 | 991106 | 01199999999 |
| 2 | def | 151515 | 마바사 | 760508 | 01098765432 |

변경 후 join_membership 테이블 DATA

| | id | pw | name | birthdate | tel |
|---|-----|--------|------|-----------|-------------|
| 1 | def | 151515 | 마바사 | 760508 | 01098765432 |

나. 회원 정보 수정 / 탈퇴

<2. 상세 구현 내용>

4

로그인 및 회원 정보
관리

- join_membershipVO.java

```
public class join_membershipVO {  
    String id;  
    String pw;  
    String name;  
    String birthdate;  
    String tel;  
}
```

```
public String getId() {  
    return id;  
}  
public void setId(String id) {  
    this.id = id;  
}
```

```
public String getPw() {  
    return pw;  
}  
public void setPassword(String pw) {  
    this.pw = pw;  
}
```

```
public String getBirthdate() {  
    return birthdate;  
}  
public void setBirthdate(String birthdate) {  
    this.birthdate = birthdate;  
}  
public String getName() {  
    return name;  
}  
public void setName(String name) {  
    this.name = name;  
}  
public String getTel() {  
    return tel;  
}  
public void setTel(String tel) {  
    this.tel = tel;  
}
```

VO클래스 → 변수를 생성하고, getters and setters 함수 정의

```
// 회원 탈퇴를 위한 D  
public void delete(String id) throws Exception {  
    // 1. 커넥터 사용하겠다고 설정해야함.  
    Class.forName("com.mysql.jdbc.Driver");  
  
    // 2. db 연결-lol, root, 1234  
    String url = "jdbc:mysql://localhost:3306/lol";  
    Connection con = DriverManager.getConnection(url, "root", "1234");  
  
    // 3. SQL문을 만든다.  
    String sql = "delete from join_membership where id = ?";  
    PreparedStatement ps = con.prepareStatement(sql);  
    ps.setString(1, id);  
  
    // 4. SQL문을 mySQL서버로 전송함.  
    ps.executeUpdate();  
}
```

- join_membershipDAO.java

- 1) jdbc 커넥터 사용
- 2) lol db에 연결
- 3) SQL문 생성
→ 입력한 id에 해당하는 row 테이블에서 삭제.
- 4) SQL문을 MySQL 서버로 전송

<3. 소 감>

프로젝트의 주제는 “게임 정보 조회 사이트 ” 로 선정했습니다. 조회 사이트로 정하게 된 이유는 첫 프로젝트인 만큼 기초로 배웠던 기술들을 이용해서 간단한 기능의 사이트를 구현하고자 했습니다.

먼저, 팀원들과 회의를 통해 참고할 사이트, 들어갈 페이지, 기능, 디자인 등을 기획했습니다.

이 후 진행하는 과정에서 처음에 기획했던 기능이나 페이지가 여러 차례 수정되었는데, 마우스 오버 하면 팝업으로 정보를 보여주려고 했던 페이지를, 검색기능으로 변경하였고, 크롤링을 통해 받아오려고 했던 데이터가 크롤링이 불가능 하다는 사실을 알고, 데이터를 직접 입력하는 방식으로 바꾸기도 했습니다.

기획했던 사항들을 실제로 구현해보면서 가능/불가능 여부와 효율성 등을 따져 수정하는 과정이 재미있었고, 프로젝트에 꼭 필요한 과정이라고 느꼈습니다.

또한 프로젝트를 여러 차례 하면서 경험이 쌓이면, 기획 단계에서도 추후에 발생할 문제점을 미리 파악하여 반영하고, 불필요한 수정 단계를 줄일 수 있을 것 같다고 생각했습니다.

감사합니다.

2021-03-03 강 지 수