

# JSP



개발내용

## JSP, DB를 이용한 드라마 소개 홈페이지 제작

- 드라마 검색, 표시, 로그인, 로그아웃, 회원수정, 탈퇴구현
- Cos 라이브러리를 이용한 업로드 이미지 구현
- Bundle을 이용하여 한 · 영버전으로 홈페이지 구현
- Commons codec 라이브러리를 이용한 데이터 암호/복호화 기능 구현
- Jsoup 라이브러리를 이용한 날씨 크롤링
- Ajax와 모달을 이용하여 자가진단 · 날씨 구현
- OPEN API 사용 (KaKao 우편검색, KaKao 음성합성, Speech Synthesis API)

# JSP



## 개발툴

**MySQL, JSP, TOMCAT**

- Model : MySQL
- View, Controller: JSP
- AWS : TomCat v8.5

## 언어

**HTML5/ CSS/ Javascript**

- JQuery
- Java
- JSP
- Ajax

# DATABASE TABLE DES



## DB 테이블 구조

### Members

Members

🔑 id NOT NULL  
◆ password NOT NULL  
◆ name NOT NULL  
◆ gender NULL  
◆ birth NULL  
◆ mail/UK NULL  
◆ phone/UK NULL  
◆ address NOT NULL  
◆ regist\_day NULL

### rvdrama

rvdrama

◆ num NOT NULL  
◆ id NOT NULL (FK)  
◆ title NOT NULL  
◆ content NULL  
◆ date NULL  
◆ view NULL  
◆ repl NULL  
◆ starinput NULL

### wish

wish

◆ num NOT NULL  
◆ l\_id NULL (FK)  
◆ p\_id NOT NULL

# DATABASE CONNECTION



```
private Connection getConnection(){
```

```
    try {
```

```
        //Context 객체를 생성 (프로젝트 정보를 가지고있는객체)
```

```
        Context initCTX = new InitialContext();
```

```
        // DB연동 정보를 불러오기(context.xml)
```

```
        DataSource ds =
```

```
        (DataSource) initCTX.lookup("java:comp/env/jdbc/mysqlDB");
```

```
        conn = ds.getConnection();
```

```
    }
```

```
    catch (NamingException e) {e.printStackTrace();}
```

```
    catch (SQLException e) {e.printStackTrace();}
```

```
    return conn;
```

```
}
```

MySQL(데이터베이스)을  
연결하기 위한 코드  
DAO 작성

# DATABASE CONNECTION



```
public void closeDB(){  
    try{  
        if(rs !=null)  
        {  
            rs.close();  
        }  
        if(pstmt!=null)  
        {  
            pstmt.close();  
        }  
        if(conn!=null)  
        {  
            conn.close();  
        }  
    }catch(SQLException e){  
    }  
}
```

MySQL(데이터베이스)을  
자원 해제하기 위한 코드  
DAO 작성

# AES256 Libraries



```
/**
 * 양방향 암호화 알고리즘인 AES256 암호화를 지원하는 클래스
 */
public class AES256Util {
    private String iv;
    private Key keySpec;

    /**
     * 16자리의 키값을 입력하여 객체를 생성한다.
     *
     * @param key
     *          암호/복호화를 위한 키값
     * @throws UnsupportedOperationException
     *          키값의 길이가 16이하일 경우 발생
     */
    final static String key = "                ";

    public AES256Util() throws UnsupportedOperationException {
        this.iv = key.substring(0, 16);
        byte[] keyBytes = new byte[16];
        byte[] b = key.getBytes("UTF-8");
        int len = b.length;
        if (len > keyBytes.length) {
            len = keyBytes.length;
        }
        System.arraycopy(b, 0, keyBytes, 0, len);
        SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES");

        this.keySpec = keySpec;
    }
}
```

키 값의 길이를 16이상으로 맞춰  
16자리의 키 값을 입력하여 객체를 생성

# AES256 Libraries



비밀번호를 암호화

```
/**
 * AES256 으로 암호화 한다.
 *
 * @param str
 *         암호화할 문자열
 *
 * @return
 * @throws NoSuchAlgorithmException
 * @throws GeneralSecurityException
 * @throws UnsupportedEncodingException
 */
public String encrypt(String str) throws NoSuchAlgorithmException,
        GeneralSecurityException, UnsupportedEncodingException {
    Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
    c.init(Cipher.ENCRYPT_MODE, keySpec, new IvParameterSpec(iv.getBytes()));
    byte[] encrypted = c.doFinal(str.getBytes("UTF-8"));
    String enStr = new String(Base64.encodeBase64(encrypted));
    return enStr;
}
```

# AES256 Libraries



## 비밀번호를 복호화

```
/**
 * AES256으로 암호화된 txt 를 복호화한다.
 *
 * @param str
 *         복호화할 문자열
 * @return
 * @throws NoSuchAlgorithmException
 * @throws GeneralSecurityException
 * @throws UnsupportedEncodingException
 */
public String decrypt(String str) throws NoSuchAlgorithmException,
        GeneralSecurityException, UnsupportedEncodingException {
    Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
    c.init(Cipher.DECRYPT_MODE, keySpec, new IvParameterSpec(iv.getBytes()));
    byte[] byteStr = Base64.decodeBase64(str.getBytes());
    return new String(c.doFinal(byteStr), "UTF-8");
}
```



# POST API



```
function execDaumPostcode() {  
    new daum.Postcode({  
        oncomplete: function(data) {  
            var fullAddr = '';  
            var extraAddr = '';  
            if (data.userSelectedType === 'R') {  
                fullAddr = data.roadAddress;  
  
            } else {  
                fullAddr = data.jibunAddress;  
            }  
            if(data.userSelectedType === 'R'){  
                if(data.bname !== ''){  
                    extraAddr += data.bname;  
                }  
                if(data.buildingName !== ''){  
                    extraAddr += (extraAddr !== '' ? ', ' + data.buildingName : data.buildingName);  
                }  
                fullAddr += (extraAddr !== '' ? ' (' + extraAddr + ')' : '');  
            }  
            document.getElementById('post_address').value = data.zonecode;  
            document.getElementById('user_address').value = fullAddr;  
            document.getElementById('detail_address').focus();  
        }  
    }).open();  
}
```

우편번호 API

# Weather crawling



```
<%  
Document doc2 = Jsoup.connect("http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108").get();  
Elements local = doc2.select("location");  
Elements time = local.select("tmEf");  
Elements city = local.select("city");  
Elements wf = local.select("wf");  
  
ArrayList<String> arrTime = new ArrayList<>();  
ArrayList<String> arrCity = new ArrayList<>();  
ArrayList<String> arrWf = new ArrayList<>();  
  
arrList(arrTime,time.text().replace(":00 ", ":00/").split("/"));  
arrList(arrCity,city.text().split(" "));  
arrAllList(arrWf,wf.text().replace("리고 비", "리고_비").replace(" ", "/").split("/"));  
  
int cnt=0;  
%>  
<%=arrTime%>  
<%=arrCity%>  
<%=arrWf%>
```

Jsoup을 이용한 크롤링

# Covid19



```
<script type="text/javascript">
$(document).ready(function(){
    $('#su').click(function(){
        $.ajax({
            url: "../covid19/covidpro.jsp",
            type: "post",
            data: {strAns1:$('#strAns1').is(":checked"),strAns2:$('#strAns2').is(":checked"),strAns3:$('#strAns3').is(":checked"),
                strAns4:$('#strAns4').is(":checked"),strAns5:$('#strAns5').is(":checked"),strAns6:$('#strAns6').is(":checked"),
                strAns7:$('#strAns7').is(":checked"),strAns8:$('#strAns8').is(":checked"),strAns9:$('#strAns9').is(":checked"),
                strAns10:$('#strAns10').is(":checked")},
            success: function(data){
                if(data.trim()=="true")
                {
                    var audio = new Audio('../covid19/voice/yes.mp3');
                }
                else
                {
                    var audio = new Audio('../covid19/voice/no.mp3');
                }
                audio.play();
            }
        });
    });
});
</script>
```

AJAX을 이용한 자가진단

# Menu

# -Main



```
<ul class="navbar-nav mr-auto">
  <li class="nav-item"><jsp:include page="../covid19/covid19.jsp"/></li>
  <li class="nav-item"><jsp:include page="../Weather/Weather.jsp"/></li>
  <c:choose>
    <c:when test="${empty sessionId}">
      <li class="nav-item"><a class="nav-link"
        href="<c:url value='../members/Login.do'/> ">로그인</a></li>

      <li class="nav-item"><a class="nav-link"
        href="<c:url value="/members/MemberAdd.do"/> ">회원가입</a></li>

    </c:when>
  </c:choose>
```

로그인 회원가입

로그인 하지 않았을 경우

메뉴창에서 로그인과 회원가입을 보여줌

# menu



```
<c:otherwise>
  <li style="padding-top: 7px; color:white">[<%=sessionId %>님]</li>

  <li class="nav-item"><a class="nav-link" href="<c:url value=" ../products/CartAction.do"/>">찜목록</a></li>

  <li class="nav-item"><a class="nav-link" href="<c:url value=" ../members/MemberUpdate.do"/>">회원수정</a></li>

  <li class="nav-item"><a class="nav-link" href="<c:url value=" ../members/MemberLogout.do"/>">로그아웃</a></li>

</c:otherwise>
</c:choose>
```

[test님] 찜목록 회원수정 로그아웃

로그인 하였을 경우  
메뉴 창에서 회원의 아이디를 찍어주고  
찜 목록, 회원 수정, 로그아웃을 보여줌

# menu



```
<c:choose>
  <c:when test="${sessionId eq 'admin' }">
    <li class="nav-item"><a class="nav-link" href="../products/ViewProducts.do">드라마 목록</a></li>
    <li class="nav-item"><a class="nav-link" href="../admin/ProductAdd.do">드라마 등록</a></li>
    <li class="nav-item"><a class="nav-link" href="../admin/ProductEdit.do?edit=update">드라마 수정</a></li>
    <li class="nav-item"><a class="nav-link" href="../admin/ProductEdit.do?edit=delete">드라마 삭제</a></li>
    <li class="nav-item"><a class="nav-link" href="../rvdrama/ReviewAction.do">리뷰 목록</a></li>
  </c:when>
```

[admin님] 찜목록 회원수정 로그아웃 드라마 목록 드라마 등록 드라마 수정 드라마 삭제 리뷰 목록

관리자의 아이디로 로그인 한 경우

메뉴 창에 회원의 아이디를 찍어주고  
찜 목록, 회원 수정, 로그아웃만이 아니라  
드라마의 등록, 수정, 삭제를 할 수 있는 기능까지 보여줌

# menu



```
<c:otherwise>
  <li class="nav-item"><a class="nav-link" href="../products/ViewProducts.do">드라마 목록</a></li>
  <li class="nav-item"><a class="nav-link" href="../rvdrama/ReviewAction.do">리뷰 목록</a></li>
</c:otherwise>
```

드라마 목록   리뷰 목록

관리자를 제외한 로그인 상태에서 마지막으로 보여지는 메뉴 창

# ProcessLogin - Member



```
request.setCharacterEncoding("utf-8");  
String id= request.getParameter("id");  
String password=request.getParameter("password");
```

```
AES256Util pw_secure= new AES256Util();  
String hpassword="", check_pw="";
```

```
DramaDAO dd = new DramaDAO();
```

```
MembersDramaBean mb = dd.LoginMember(id);  
if(dd.LoginMemberCk(id)==1)  
{  
    hpassword=mb.getPw();  
}  
check_pw=pw_secure.decrypt(hpassword);
```

LoginForm 에서 입력한 값을  
Request를 통해 가져와

DramaDAO를 통해 DB에서 값을  
얻어오는 코드



# ProcessLogin



```
if(check_pw.equals(password))
{
    session.setAttribute("sessionId", id);
    response.sendRedirect("../members/resultMember.jsp?msg=2");
}
else
{
    response.sendRedirect("../members/loginMember.jsp?error=1");
}
```

로그인

복호화된 비밀번호가 일치할 경우  
세션에 id값을 설정  
resultMember.jsp?msg=2로 이동  
로그인 성공 페이지가 보임

비밀번호가 일치하지 않을 경우  
loginMember.jsp?error=1값을 반환  
로그인이 실패했다는 메시지를 띄움

# DramaDAO (LoginMember)



```
public MembersDramaBean LoginMember(String id){
    conn=getConnection();
    sql="select password from members where id=?";
    MembersDramaBean mb=null;
    try{
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1,id);
        rs=pstmt.executeQuery();

        if(rs.next()){
            mb= new MembersDramaBean();

            mb.setPw(rs.getString(1));
        }
    }catch(SQLException e)
    {
        e.printStackTrace();
    }
    finally{
        closeDB();
    }
    return mb;
}
```

LoginForm에서 입력한 값을 얻어와  
데이터베이스에서 결과값을 얻어  
MemberBean 객체를 반환해줌

# LoginMember



```
String error= request.getParameter("error");
if(error!=null)
{
    if(cnt==3){
        out.println("<div class='alert alert-danger'> ");
        out.println("존재하는 아이디가 없습니다. 회원가입을 해주세요!");
        out.println("</div>");
    }
    else
    {
        out.println("<div class='alert alert-danger'> ");
        out.println("아이디와 비밀번호를 확인해주세요!");
        out.println("</div>");
    }
    cnt++;
}
```

로그인 에러메세지를 받을 때의 코드

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

if(msg!=null)
{
    if(msg.equals("0"))
    {
        out.println("<h2 class='alert alert-danger'>회원정보가 수정되었습니다.</h2>");
    }
    else if(msg.equals("1"))
    {
        out.println("<h2 class='alert alert-danger'>회원가입을 축하드립니다.</h2>");
    }
    else if(msg.equals("2"))
    {
        String loginId=(String)session.getAttribute("sessionId");
        out.println("<h2 class='alert alert-danger'>"+ loginId+ " 님 환영합니다.</h2>");
    }
}
else
{
    out.println("<h2 class='alert alert-danger'>회원 정보가 삭제되었습니다.");
}
```

msg의 값에 따른 처리(resultMember.jsp)

# ProcessAddMember



```
if(check)
{
    dd.addMember(id, hpw, name, gender, birth, email, phone, address);
%>
<c:redirect url="../members/resultMember.jsp?msg=1"/>
<%
}
else
{
%>
<script>
    alert(error);
    window.history.back();
</script>
<%
}
%>
```

유효성 검사 이후 DramaDAO객체에  
값을 입력하는 코드  
(회원가입)

# DramaDAO (addMember)



```
public void addMember(String id, String hpw, String name, String gender,
    String birth, String email, String phone, String address){
    conn= getConnection();
    sql = "insert into members values(?,?,?,?,?,?,?,?,?)";
    Date date=new Date();
    SimpleDateFormat format= new SimpleDateFormat("yyyy-MM-dd");
    try {
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1, id);
        pstmt.setString(2, hpw);
        pstmt.setString(3, name);
        pstmt.setString(4, gender);
        pstmt.setString(5, birth);
        pstmt.setString(6, email);
        pstmt.setString(7, phone);
        pstmt.setString(8, address);
        pstmt.setString(9, format.format(date));
        pstmt.executeUpdate();
        System.out.print("저장완료");
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        closeDB();
    }
}
```

회원가입을 위한 코드

비밀번호의 경우

암호화된 비밀번호를

데이터베이스에 저장

# DeleteMember



```
DramaDAO dd = new DramaDAO();  
request.setCharacterEncoding("utf-8");  
String id = (String)session.getAttribute("sessionId");  
  
dd.deleteMember(id);  
  
session.invalidate();  
response.sendRedirect("../members/resultMember.jsp");
```

회원수정에서 회원탈퇴를 눌렀을 때  
이동하는 페이지로, DAO객체에서 처리 후  
session.invalidate()를 이용하여 세션 값 제거

# DramaDAO (deleteMember)



```
public void deleteMember(String id){  
    conn=getConnection();  
    sql="delete from members where id=?";  
    try{  
        pstmt=conn.prepareStatement(sql);  
        pstmt.setString(1, id);  
        pstmt.executeUpdate();  
        RDBinit();  
        DBinit();  
    }catch (SQLException e) {  
        e.printStackTrace();  
    }finally {  
        closeDB();  
    }  
}
```

데이터베이스에서  
입력한 값을 제거  
(회원탈퇴)

# ProcessUpdate



```
if(UpdateCheck(pw, mail).equals("all"))
{
    if(dd.UpdateMemberCheck(id))
    {
        dd.UpdateMember(hpw, mail, phone, address, id);
        check=true;
    }
}
else if(UpdateCheck(pw, mail).equals("pw"))
{
    if(dd.UpdateMemberCheck(id))
    {
        dd.UpdatePwMember(hpw, phone, address, id);
        check=true;
    }
}
else if(UpdateCheck(pw, mail).equals("mail"))
{
    if(dd.UpdateMemberCheck(id))
    {
        dd.UpdateMailMember(mail, phone, address, id);
        check=true;
    }
}
else
{
    if(dd.UpdateMemberCheck(id))
    {
        dd.UpdateMember(phone, address, id);
        check=true;
    }
}
```

회원수정에서 나눠지는 값들을  
DAO 에서 세분화 시켜 데이터를 처리

All : 비밀번호와 이메일 변경  
Pw: 비밀번호 변경  
Mail: 이메일 변경

→ PW 와 Mail의 정보는 필요하기 때문에  
각 각 데이터 처리를 해줌



# DAO (UpdateMemberCheck)



```
public boolean UpdateMemberCheck(String id){
    boolean check=false;
    conn=getConnection();
    sql = "select password from members where id= ?";
    try {
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1, id);
        rs=pstmt.executeQuery();
        if(rs.next())
        {
            check=true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        closeDB();
    }
    return check;
}
```

회원 정보 수정을 위해

데이터베이스에서

회원이 있는지 확인

# DramaDAO (UpdateMember)



```
public void UpdateMember(String hpw,String mail, String phone,
                        String address,String id){
    conn=getConnection();
    sql="update members set password=?, mail=?,phone=?,address=? where id=?";
    try {
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1, hpw);
        pstmt.setString(2, mail);
        pstmt.setString(3, phone);
        pstmt.setString(4, address);
        pstmt.setString(5, id);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        closeDB();
    }
}
```

비밀번호와 이메일 값의  
데이터 정보를  
처리하는 코드

# DAO (UpdateMailMember)



```
public void UpdateMailMember(String mail, String phone, String address, String id) {  
    conn = getConnection();  
    sql = "update members set mail=?,phone=?,address=? where id=?";  
    try {  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, mail);  
        pstmt.setString(2, phone);  
        pstmt.setString(3, address);  
        pstmt.setString(4, id);  
        pstmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        closeDB();  
    }  
}
```

이메일 값의  
데이터 정보를  
처리하는 코드

# DAO (UpdatePWMember)



```
public void UpdatePWMember(String hpw, String phone, String address, String id) {  
    conn = getConnection();  
    sql = "update members set password=?,phone=?,address=? where id=?";  
    try {  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, hpw);  
        pstmt.setString(2, phone);  
        pstmt.setString(3, address);  
        pstmt.setString(4, id);  
        pstmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        closeDB();  
    }  
}
```

비밀번호 값의  
데이터 정보를  
처리하는 코드

# DAO (UpdateMember)



```
public void UpdateMember(String phone, String address, String id) {  
    conn = getConnection();  
    sql = "update members set phone=?,address=? where id=?";  
    try {  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, phone);  
        pstmt.setString(2, address);  
        pstmt.setString(3, id);  
        pstmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        closeDB();  
    }  
}
```

비밀번호와 이메일  
값이 없는 데이터 정보를  
처리하는 코드

오버라이딩 이용

# SearchMember - ID



```
<%  
    if(search.equals("id"))  
    {  
%>  
        <form class="form-horizontal" action="processSearchMember.jsp?search=id" method="post">  
  
            <div class="form-group row">  
                <label class="col-sm-4"><fmt:message key="user_email"/></label>  
                <div class="col-sm-3">  
                    <input type="text" placeholder="<fmt:message key="user_email"/>" name="user_email" required >  
                </div>  
            </div>  
  
            <div class="form-group row">  
                <label class="col-sm-4"><fmt:message key="user_phone"/></label>  
                <div class="col-sm-8">  
                    <select name="phone1">  
                        <option value="010">010</option>  
                        <option value="011">011</option>  
                        <option value="016">016</option>  
                        <option value="017">017</option>  
                        <option value="018">018</option>  
                        <option value="019">019</option>  
                    </select> -  
                    <input type="text" maxlength="4" size="4" name="phone2"> -  
                    <input type="text" maxlength="4" size="4" name="phone3">  
                </div>  
            </div>  
            <button class="btn btn-lg btn-success btn-block" type="submit">회원 찾기</button>  
        </form>  
%>  
    }  
}
```

SearchMember.jsp

Parameter값에 따라  
아이디 와 비밀번호 처리

Parameter값이 id인 경우  
보여지는 화면

# SearchMember – PW



```
if(search.equals("pw"))
{
%>
    <form class="form-horizontal" action="processSearchMember.jsp?search=pw" method="post">

        <div class="form-group row">
            <label class="col-sm-4"><fmt:message key="user_id"/></label>
            <div class="col-sm-3">
                <input type="text" placeholder="<fmt:message key="user_id"/>" name="user_id" required autofocus><br/>
            </div>
        </div>

        <div class="form-group row">
            <label class="col-sm-4"><fmt:message key="user_email"/></label>
            <div class="col-sm-3">
                <input type="text" placeholder="<fmt:message key="user_email"/>" name="user_email" required >
            </div>
        </div>

        <div class="form-group row">
            <label class="col-sm-4"><fmt:message key="user_phone"/></label>
            <div class="col-sm-8">
                <select name="phone1">
                    <option value="010">010</option>
                    <option value="011">011</option>
                    <option value="016">016</option>
                    <option value="017">017</option>
                    <option value="018">018</option>
                    <option value="019">019</option>
                </select> -
                <input type="text" maxlength="4" size="4" name="phone2"> -
                <input type="text" maxlength="4" size="4" name="phone3">
            </div>
        </div>
        <button class="btn btn-lg btn-success btn-block" type="submit">회원 찾기</button>
    </form>
```

## SearchMember.jsp

Parameter값에 따라  
아이디 와 비밀번호 처리

Parameter값이 pw인 경우  
보여지는 화면

# DAO (addProduct) – Admin



```
public void addProduct(String filename, String code,
    String name, String bu, String description,
    String genre, String broad, String person, String year){
    getConnection();
    sql="insert into jpdrama values(?,?,?,?,?,?,?,now(6),?)";
    try {
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1,filename);
        pstmt.setString(2,code);
        pstmt.setString(3,name);
        pstmt.setString(4,bu);
        pstmt.setString(5,description);
        pstmt.setString(6,genre);
        pstmt.setString(7,broad);
        pstmt.setString(8,person);
        pstmt.setString(9,year);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally{
        closeDB();
    }
}
```

드라마를 업로드할 경우  
데이터베이스에서  
처리하는 코드



# editProduct



```
if(edit.equals("update"))
{
    %>

    <a href="../../admin/updateProduct.jsp?id=<%=jb.getP_code()%>"
    class="btn btn-success" role="button">수정 &raquo;</a>
    <%}

    else if(edit.equals("delete"))
    {
        %>

        <a href="#" onclick="deleteConfirm('<%=jb.getP_code() %>')"
        class="btn btn-danger" role="button">삭제 &raquo;</a>
        <%
    }
    %>
```

드라마를 수정할 경우  
Parameter값(edit)에 따라  
처리하는 코드

# DAO (updateProduct)



```
public void updateProduct(String filename, String code,
    String name, String bu, String description,
    String genre, String broad, String person, String year){
    getConnection();
    try {
        if(filename !=null)
        {
            sql="update jpdrama set jp_filename=?, jp_name=?,jp_bu=?,jp_description=?, jp_genre=?, jp_broad=?, jp_person=?, jp_year =? where jp_code=?";
            pstmt=conn.prepareStatement(sql);
            pstmt.setString(1,filename);
            pstmt.setString(2,name);
            pstmt.setString(3, bu);
            pstmt.setString(4,description);
            pstmt.setString(5,genre);
            pstmt.setString(6,broad);
            pstmt.setString(7,person);
            pstmt.setString(8,year);
            pstmt.setString(9,code);
            pstmt.executeUpdate();
        }
        else
        {
            sql="update jpdrama set jp_name=?,jp_bu=?,jp_description=?, jp_genre=?, jp_broad=?, jp_person=?, jp_year =? where jp_code=?";
            pstmt=conn.prepareStatement(sql);
            pstmt.setString(1,name);
            pstmt.setString(2, bu);
            pstmt.setString(3,description);
            pstmt.setString(4,genre);
            pstmt.setString(5,broad);
            pstmt.setString(6,person);
            pstmt.setString(7,year);
            pstmt.setString(8,code);
            pstmt.executeUpdate();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally{
        closeDB();
    }
}
```

관리자가 상품을 수정 시

이미지 변경 할 경우

→ 데이터베이스에서  
이미지 값까지 수정

이미지를 변경하지 않을 경우

→데이터베이스에서  
이미지 값 없이 수정

# DAO (deleteProduct)



```
public void deleteProduct(String code){
    getConnection();
    sql="select * from jpdrama";
    try {
        pstmt=conn.prepareStatement(sql);
        rs=pstmt.executeQuery();

        if(rs.next())
        {
            sql="delete from jpdrama where jp_code= ?";
            pstmt= conn.prepareStatement(sql);
            pstmt.setString(1, code);
            pstmt.executeUpdate();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally{
        closeDB();
    }
}
```

관리자가 상품을 삭제 시  
코드 번호가 같은 상품을 삭제

# DAO (getDramaList) -Product



```
public ArrayList getDramaList(int PageSize, int record_start_no) {  
    conn = getConnection();  
  
    sql = "select * from jpdrama order by jp_date desc limit ? offset ?";  
  
    ArrayList DramaList = new ArrayList();  
  
    JPDramaBean jb = null;  
  
    try {  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setInt(1, PageSize);  
        pstmt.setInt(2, record_start_no - 1);  
        // 4 sql 실행  
        rs = pstmt.executeQuery();  
  
        // 5. 데이터 처리  
        while (rs.next()) {  
            jb = new JPDramaBean();  
            jb.setJp_filename(rs.getString("jp_filename"));  
            jb.setJp_name(rs.getString("jp_name"));  
            jb.setJp_description(rs.getString("jp_description"));  
            jb.setJp_genre(rs.getString("jp_genre"));  
            jb.setJp_broad(rs.getString("jp_broad"));  
            jb.setJp_person(rs.getString("jp_person"));  
            jb.setJp_bu(rs.getString("jp_bu"));  
            jb.setJp_code(rs.getString("jp_code"));  
  
            // Bean -> ArrayList 한칸에 저장  
            DramaList.add(jb);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        closeDB();  
    }  
  
    return DramaList;  
}
```

데이터베이스에 있는 모든 드라마 데이터를  
가져와서 뿌려주는 작업

# DAO (getDetailList)



```
public JPDramaBean getDetailList(String code){
    conn=getConnection();
    sql="select * from jpdrama where jp_code=?";
    JPDramaBean jb=null;
    try{
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1,code);
        rs=pstmt.executeQuery();

        if(rs.next()){
            jb= new JPDramaBean();

            jb.setJp_filename(rs.getString("jp_filename"));
            jb.setJp_name(rs.getString("jp_name"));
            jb.setJp_description(rs.getString("jp_description"));
            jb.setJp_genre(rs.getString("jp_genre"));
            jb.setJp_bu(rs.getString("jp_bu"));
            jb.setJp_broad(rs.getString("jp_broad"));
            jb.setJp_year(rs.getString("jp_year"));
            jb.setJp_person(rs.getString("jp_person"));
            jb.setJp_code(rs.getString("jp_code"));

        }
    }catch(SQLException e)
    {
        e.printStackTrace();
    }
    finally{
        closeDB();
    }
    return jb;
}
```

데이터베이스에서 선택한 드라마 데이터를  
가져와서 뿌려주는 작업

# DAO (addCart)



```
public void addCart(String loginId, String name){  
    conn= getConnection();  
    sql = "insert into wish values(null,?,?)";  
    try {  
        pstmt=conn.prepareStatement(sql);  
        pstmt.setString(1,loginId);  
        pstmt.setString(2,name);  
        pstmt.executeUpdate();  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }finally{  
        closeDB();  
    }  
}
```

찜목록에서 상품을 추가 시

데이터베이스에 데이터 추가

# DAO (Cart)



```
public ArrayList Cart(String loginId) {  
    conn = getConnection();  
    sql = "select * from wish where l_id=?";  
    ArrayList WishList = new ArrayList();  
    WishDramaBean wb = null;  
    try {  
        pstmt=conn.prepareStatement(sql);  
        pstmt.setString(1, loginId);  
        rs=pstmt.executeQuery();  
        // 4 sql 실행  
        rs = pstmt.executeQuery();  
        // 5. 데이터 처리  
        while (rs.next()) {  
            wb = new WishDramaBean();  
            wb.setL_id(rs.getString("l_id"));  
            wb.setP_id(rs.getString("p_id"));  
            // Bean -> ArrayList 한칸에 저장  
            WishList.add(wb);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        closeDB();  
    }  
    return WishList;  
}
```

데이터베이스에서 아이디가 동일한  
찜목록을 뿌려주는 코드

# DAO (RemoveWish)



```
public void RemoveWish(String loginId,String id){  
    conn=getConnection();  
    sql="delete from wish where l_id=? and p_id= ?";  
    try {  
        pstmt=conn.prepareStatement(sql);  
        pstmt.setString(1, loginId);  
        pstmt.setString(2, id);  
        pstmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }finally {  
        closeDB();  
    }  
}
```

찜목록에서 아이디와 상품의 이름이  
동일한 상품을 삭제 하는 코드  
(선택 삭제)



# DAO (DeleteWish)



```
public void DeleteWish(String loginId){
    conn=getConnection();
    sql="delete from wish where l_id=?";
    try {
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1, loginId);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        closeDB();
    }
}
```

썸한 아이디의 모든 상품을 삭제 하는 코드

# DAO (insertDrama) - rvdrama



```
public void insertDrama(ReviewDramaBean rb){
    int num = 0;
    try {
        conn = getConnection();

        sql = "select max(num) from rvdrama";

        pstmt = conn.prepareStatement(sql);

        rs = pstmt.executeQuery();

        if(rs.next()){
            num = rs.getInt(1)+1;
        }

        // 3 sql 작성 (insert) & pstmt 객체 생성
        sql = "insert into rvdrama values(?,?,?,?,now(),?,?,?,0)";

        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, num);
        pstmt.setString(2, rb.getId());
        pstmt.setString(3, rb.getTitle());
        pstmt.setString(4, rb.getContent());
        pstmt.setInt(5, rb.getView());
        pstmt.setInt(6, rb.getRepl());
        pstmt.setFloat(7, rb.getStarinput());
        pstmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    } finally{
        // 자원해제
        try {
            pstmt.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

드라마의 리뷰를 폼에서 받아서  
데이터베이스에 저장하는 코드

# DAO (UpdateViewDrama)



```
public int updateViewDrama(ReviewDramaBean rb){
    int check=0;
    try{
        conn= getConnection();
        sql="select * from rvdrama where num=?";
        pstmt=conn.prepareStatement(sql);
        pstmt.setInt(1, rb.getNum());
        rs=pstmt.executeQuery();
        if(rs.next())
        {
            sql ="update rvdrama set title=?, content=? where num=?";
            pstmt=conn.prepareStatement(sql);
            pstmt.setString(1, rb.getTitle());
            pstmt.setString(2,rb.getContent());
            pstmt.setInt(3, rb.getNum());
            check=pstmt.executeUpdate();
        }
        else
        {
            check=-1;
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        closeDB();
    }
    return check;
}
```

드라마의 리뷰를 수정하는 코드

# DAO (deleteReview)



```
public void deleteReview(int num){  
    conn=getConnection();  
    sql="delete from rvdrama where num=?";  
    try{  
        pstmt=conn.prepareStatement(sql);  
        pstmt.setInt(1, num);  
        pstmt.executeUpdate();  
    }catch (SQLException e) {  
        e.printStackTrace();  
    }finally {  
        closeDB();  
    }  
}
```

드라마의 리뷰를 삭제하는 코드

# DAO (reInsertReview)



```
public void reInsertReview(ReviewDramaBean rb){  
    int num = 0;  
  
    try {  
        // 1) 답글 작성 번호(num)계산  
        // 1,2 디비연결  
        conn = getConnection();  
        // 3 sql 구문 & pstmt 객체  
        sql = "select max(num) from rvdrama";  
        pstmt = conn.prepareStatement(sql);  
  
        // 4 sql 실행  
        rs = pstmt.executeQuery();  
  
        // 5 데이터 처리  
        if(rs.next()){  
            //rs.getInt("max(num)");  
            num = rs.getInt(1)+1;  
        }  
    }  
}
```

답글을 작성하기 위해 값을 계산

# DAO (reInsertReview)



```
sql = "update rvdrama set repl = repl+1 where title=? and repl>?";

pstmt = conn.prepareStatement(sql);
pstmt.setString(1, rb.getTitle());
pstmt.setInt(2, rb.getRepl());

// sql 실행
pstmt.executeUpdate();
```

답글을 작성 할 경우 댓글을 하나 올림

# DAO (reInsertReview)



```
// 3) 답글 쓰기
sql = "insert into rvdrama(num,id,title,content,date,repl,pagenum) values(?,?,?,?,?,now(),?,?)";

pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, num);
pstmt.setString(2, rb.getId());
pstmt.setString(3, rb.getTitle());
pstmt.setString(4, rb.getContent());
pstmt.setInt(5, rb.getRepl()+1);
pstmt.setInt(6, rb.getNum());
// sql 실행
pstmt.executeUpdate();

System.out.println(" 답글 작성완료! ");

} catch (SQLException e) {
    e.printStackTrace();
} finally {
    closeDB();
}

}
```

답글을 작성한 부분을 데이터베이스에 저장

# DAO (reupdateViewDrama)



```
public int reupdateViewDrama(ReviewDramaBean rb){
    int check=0;
    try{
        conn= getConnection();
        sql="select * from rvdrama where num=?";
        pstmt=conn.prepareStatement(sql);
        pstmt.setInt(1, rb.getNum());
        rs=pstmt.executeQuery();
        if(rs.next())
        {
            sql ="update rvdrama set content=? where num=?";
            pstmt=conn.prepareStatement(sql);
            pstmt.setString(1,rb.getContent());
            pstmt.setInt(2, rb.getNum());
            check=pstmt.executeUpdate();
        }
        else
        {
            check=-1;
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        closeDB();
    }
    return check;
}
```

리뷰에 대한 답글 수정 코드



# Search(search\_admin) - query



```
<form action="../../query/search.jsp">
  <div class="form-group" align="center">
    <label><strong>드라마 찾기</strong></label>
  </div>
  <div class="form-group" align="center">
    <input type="text" name="search">
    <input class="btn btn-primary" type="submit" value="검색">
  </div>
</form>
```

드라마를 찾기 위해 값을 보내는 form

# Search(search\_admin)



```
request.setCharacterEncoding("utf-8");  
String search = request.getParameter("search");  
String page_search=request.getParameter("search");  
search="%" + search + "%";
```

드라마 찾기

Products.jsp 에서 값을 넣어 보내면

검색한 값에 대한 Parameter를 받아 처리하는 코드