JSP 게시판 강좌 따라하기 4강
로그인 기능 구현하기

JSP 게시판에서 로그인 기능을 구현하려면 JSP에서 회원 데이터베이스에 접근할 수 있도록 하는Data Access Object(DAO)를 만들어야합니다.DAO**는 데이터베이스 접근 객체의 약자**입니다.**하나의 회원정보를 불러오거나 입력할 때 사용합니다.**

UserDAO

```
package user;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class UserDAO {

	private Connection conn; //자바와 데이터베이스를 연결
	private PreparedStatement pstmt; //쿼리문 대기 및 설정
	private ResultSet rs; //결과값 받아오기

	//기본 생성자

	//UserDAO가 실행되면 자동으로 생성되는 부분
	//메소드마다 반복되는 코드를 이곳에 넣으면 코드가 간소화된다
	public UserDAO() {
		try {
			String dbURL = "jdbc:mariadb://localhost:3306/bbs";
			String dbID = "root";
			String dbPassword = "001025";
			Class.forName("org.mariadb.jdbc.Driver");
			conn = DriverManager.getConnection(dbURL, dbID, dbPassword);
		}catch (Exception e) {
			e.printStackTrace();
		}
```

```
        }
}
```

접근제어자를 'private'으로Connection, PreparedStatement, ResultSet을 필드로 선언합니다.

Connection= 연결

PreparedStatement= 설정 및 실행

ResultSet= 결과값

UserDAO() - 기본 생성자 메소드(클래스가 실행될 때 자동으로 생성)	
dbURL	MariaDB 와 연결시켜주는 주소
dbID	MariaDB 계정 - 따로 설정 안 했으면 기본은 'root'
dbPassword	MariaDB 비밀번호 - 설치할 때 설정한 것
Class.forName()	JDBC연결 클래스를 'String'타입으로 불러온다
DriverManager.getConnection	드라이버 매니저에 미리 저장했던 연결URL과 DB계정 정보를 담아 연결 설정을 한다

이 연결 코드들을 기본 생성자에 넣어두지 않으면 각 기능을 담당하는 메소드마다 다 똑같은 코드를 입력해주어야 합니다.**위와 같이 중복되는 것들을 기본생성자 메소드에**넣어두면 매번 같은 코드를 입력할 일이 사라지게 되니 코드가 간소화 되는 것입니다.

로그인 구현 메소드

```
//로그인 영역
        public int login(String userID, String userPassword) {
                String sql = "select userPassword from user where userID = ?";
                try {
                        pstmt = conn.prepareStatement(sql); //sql쿼리문을 대기 시킨다
                        pstmt.setString(1, userID); //첫번째 '?'에 매개변수로 받아온 'userID'를 대입
                        rs = pstmt.executeQuery(); //쿼리를 실행한 결과를 rs에 저장
                        if(rs.next()) {
                                if(rs.getString(1).equals(userPassword)) {
                                        return 1; //로그인 성공
                                }else
                                        return 0; //비밀번호 틀림
                        }
                        return -1; //아이디 없음
                }catch (Exception e) {
                        e.printStackTrace();
                }
```

```
            return -2; //오류
        }
```

로그인 기능을 담당하는 'login'메소드를 선언하고 매개변수로 실제 사용자가 입력할 아이디(userID)와 비밀번호(userPassword)를 매개변수로 설정합니다.실제 사용자가 입력한 내용은 매개변수에 담아와서 로그인 메소드를 실행할 때 각 위치에 브라우저에서 넘어온 아이디와 비밀번호를 적용시켜 결과를 도출하고 'int'타입으로 반환합니다.

login(String userID, String userPassword) - 로그인 메소드	
String sql	실제로 DB에서 입력할 쿼리문을 'sql'변수에 미리 담아둡니다
conn.preparedStatement(sql)	미리 설정한 'sql'을 셋팅하는 코드
pstmt.setString(1, userID)	쿼리문의 '1'번째 물음표에 'userID'를 대입시키는 것
rs = pstmt.executeQuery()	셋팅이 끝난 쿼리문을 실행시키고 나온 결과값을 'rs'에 저장

여기서 'rs.next()'을 실행했을 때 결과값이 존재한다면 해당 결과값을 얻을 수 있습니다**그 결과값에 따라 로그인처리를 각각 상황에 맞게 구현**할 수 있습니다.

로그인 메소드에 셋팅해놓은 쿼리문은**매개변수로 받아온 'userID'의 'userPassword'를 구하는 쿼리문**입니다. 다시 말하면**실제 사용자가 입력한 ID의 비밀번호를 구하는 쿼리문**이죠.

'rs.getString(1)'은 쿼리문을 실행하고 나온 첫 번째 결과값을 'String'타입으로 얻어옵니다.

로그인 결과값에 대한 처리	
결과값이 존재하고 그 결과값이 매개변수로 넘어온 'userPassword'와 일치한다	로그인 성공 (정수 1을 리턴)
결과값이 존재하지만 그 결과값이 매개변수로 넘어온 'userPassword'와 일치하지는 않는다	비밀번호 틀림 (정수 0을 리턴)
쿼리문을 실행하였지만 결과값이 나오지 않았다	존재하지 않는 아이디(회원정보에 없는 아이디, -1을 리턴)

---

loginAction.jsp

```
<%@page import="java.io.PrintWriter"%>
<%@page import="user.UserDAO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<% request.setCharacterEncoding("utf-8"); %>
<jsp:useBean id="user" class="user.User" scope="page" />
```

```
<jsp:setProperty name="user" property="userID" />
<jsp:setProperty name="user" property="userPassword" />
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP 게시판 웹 사이트</title>
</head>
<body>
	<%
		UserDAO userDAO = new UserDAO();
		int result = userDAO.login(user.getUserID(), user.getUserPassword());
		if(result == 1){
			PrintWriter script = response.getWriter();
			script.println("<script>");
			script.println("alert('로그인 성공')");
			script.println("location.href='main.jsp'");
			script.println("</script>");
		}else if(result == 0){
			PrintWriter script = response.getWriter();
			script.println("<script>");
			script.println("alert('비밀번호가 틀립니다')");
			script.println("history.back()");
			script.println("</script>");
		}else if(result == -1){
			PrintWriter script = response.getWriter();
			script.println("<script>");
			script.println("alert('존재하지 않는 아이디입니다')");
			script.println("history.back()");
			script.println("</script>");
		}else if(result == -2){
			PrintWriter script = response.getWriter();
			script.println("<script>");
			script.println("alert('데이터베이스 오류입니다')");
			script.println("history.back()");
			script.println("</script>");
```

```
                }
        %>
</body>
</html>
```

페이지 설정 영역	
<% request.setCharacterEncoding('UTF-8'); %>	넘어온 데이터 한글 깨짐 방지
<%@ page import="user.UserDAO" %>	우리가 만든 클래스를 사용
<%@ page import="java.io.PrintWriter" %>	자바스크립트를 사용하기 위함
<jsp:useBean id="user" class="user.User" scope="page" /> (User클래스를 자바빈즈로써 사용)	
id	이름
class	패키지명을 포함한 클래스 경로
scope	사용범위
<jsp:setProperty name="user" property="userID" />	로그인 페이지에서 받아온 사용자 ID를 'userID'에 저장
<jsp:setProperty name="user" property="userPassword" />	로그인 페이지에서 받아온 사용자 ID를 'userPassword'에 저장

```
UserDAO userDAO = new UserDAO();
int result = userDAO.login(user.getUserID(), user.getUserPassword());
```

1. UserDAO 클래스 인스턴스 'userDAO'를 생성합니다

2. int타입 변수 'result'에 'userDAO'인스턴스 클래스의 'login'메소드의 실행 결과를 저장한다.이 때 'login'메소드의 매개변수는 실제 브라우저에서 사용자가 입력한 아이디와 비밀번호를 넣어 적용시킵니다.

3. 로그인 메소드를 만들 때 결과에 따른 반환값을 모두 다르게 설정하였습니다.'loginAction' 페이지에서도 그 반환값에 맞게 각각 로직을 다르게 처리해줍니다. 로직 처리는 자바스크립트의 도움을 받습니다.

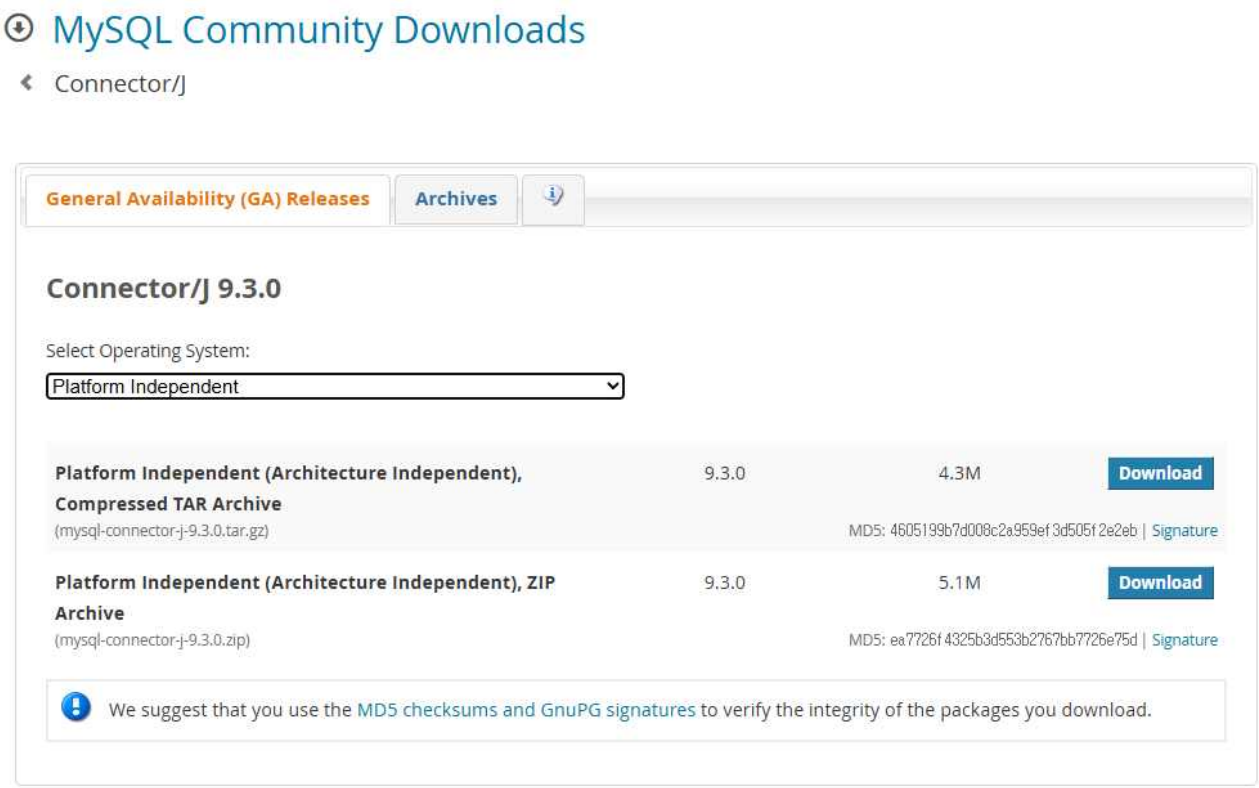
PrintWriter script = response.getWriter();	자바스크립트 문장을 실행하게 해준다
location.href='main.jsp'	해당 페이지(main.jsp)로 이동시켜준다
alert('로그인 성공')	알림창을 띄우며 설정한 문구를 보여준다
history.back()	이전 페이지로 돌아간다

JDBC Driver 프로젝트에 추가하기

자세한 설정 및 테스트는 아래 링크를 참조하셔도 되지만

본 포스팅을 따라해도 상관없습니다

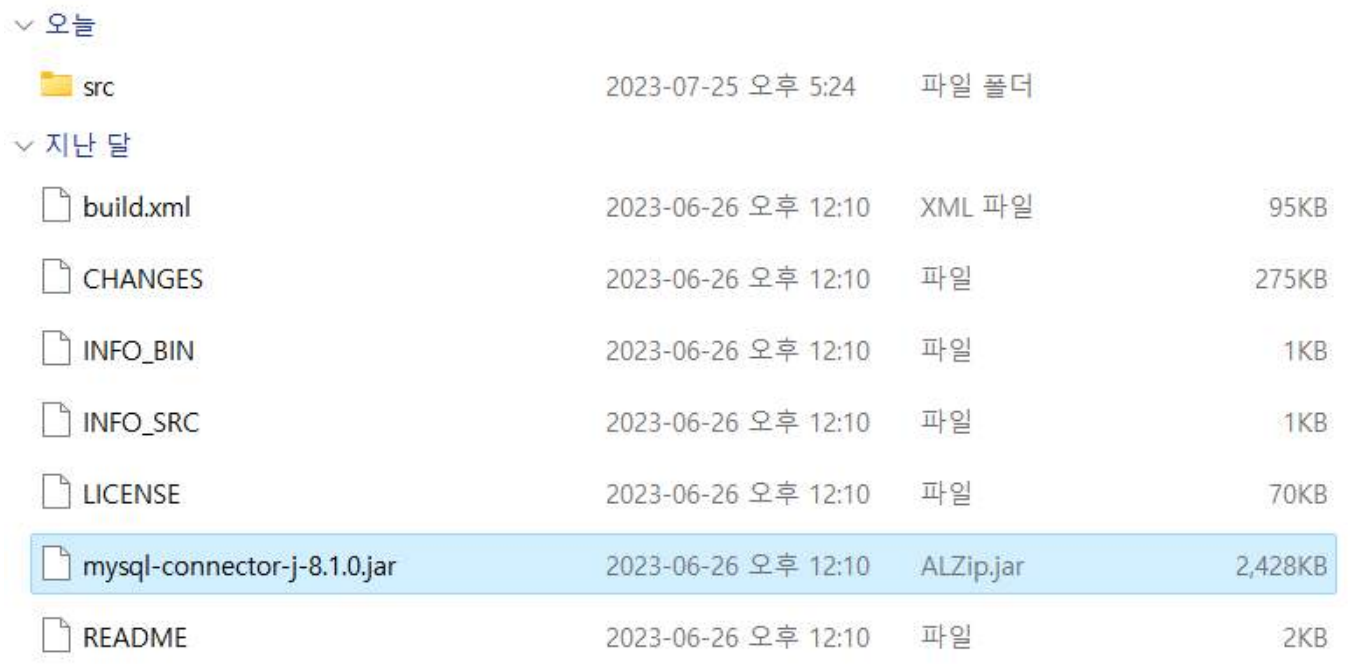
mysql jdbc driver download



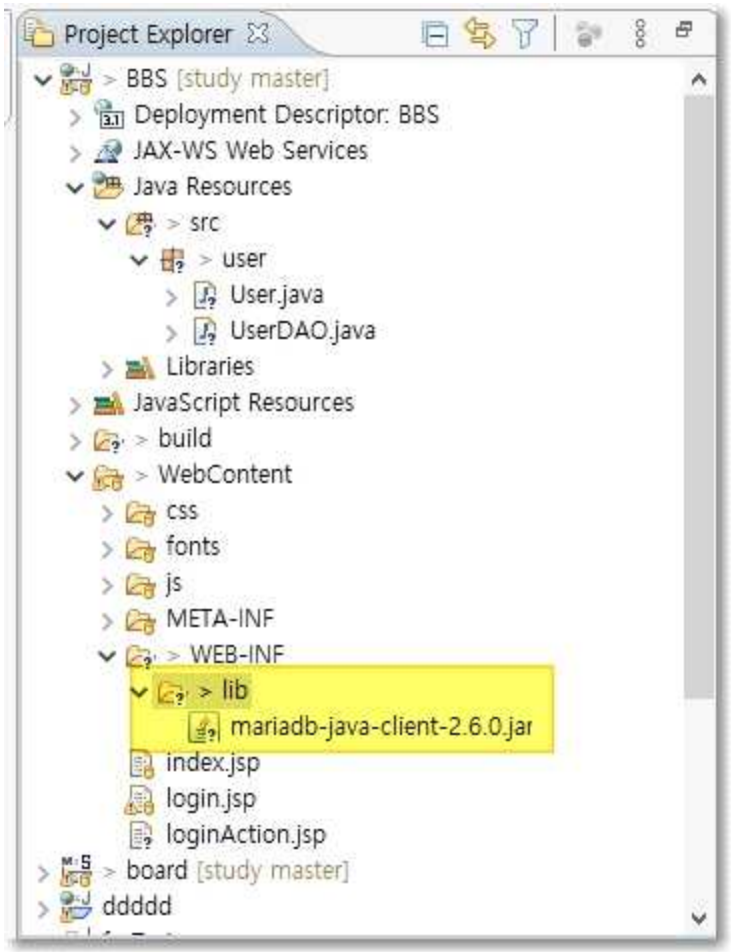
아래있는 zip 팡리 다움로드

2. connector. jar 파일 업로드

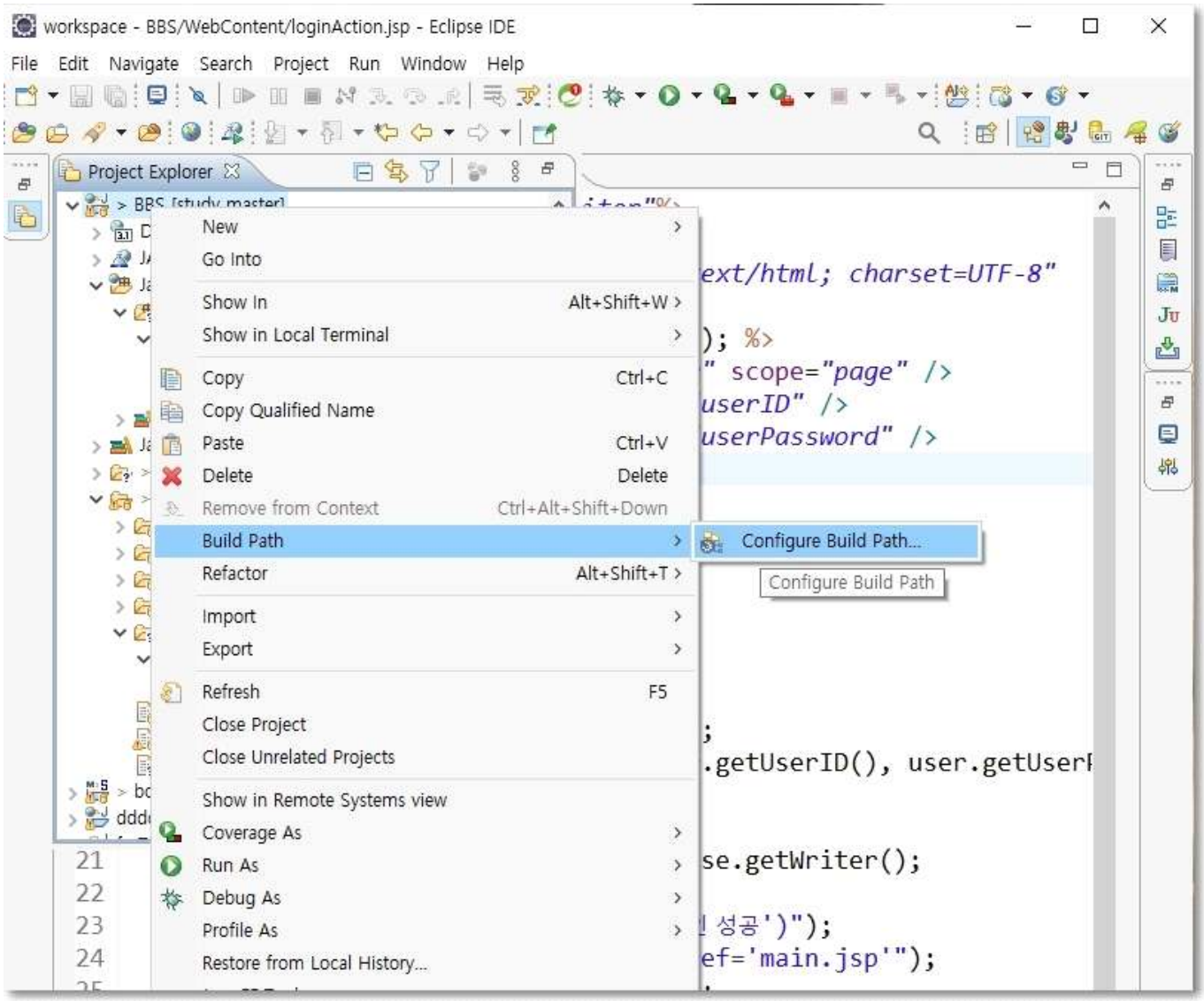
압축을 풀어서 파일을 열면 아래와 같은 파일, 폴더가 있다.
그중에 mysql-connector-j-8.1.0.jar 파일을 프로젝트 빌드 경로에 추가한다.



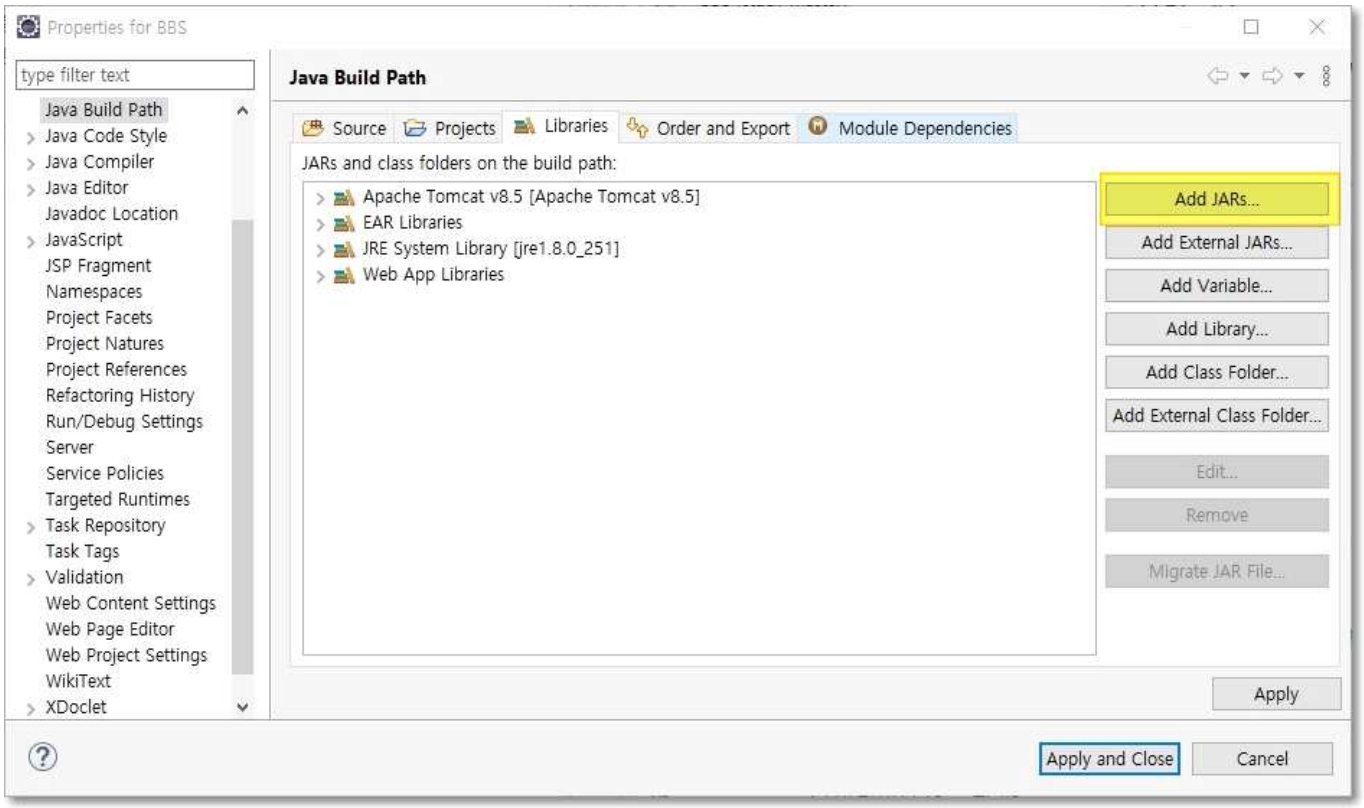
다운받은 파일을 '복사'합니다



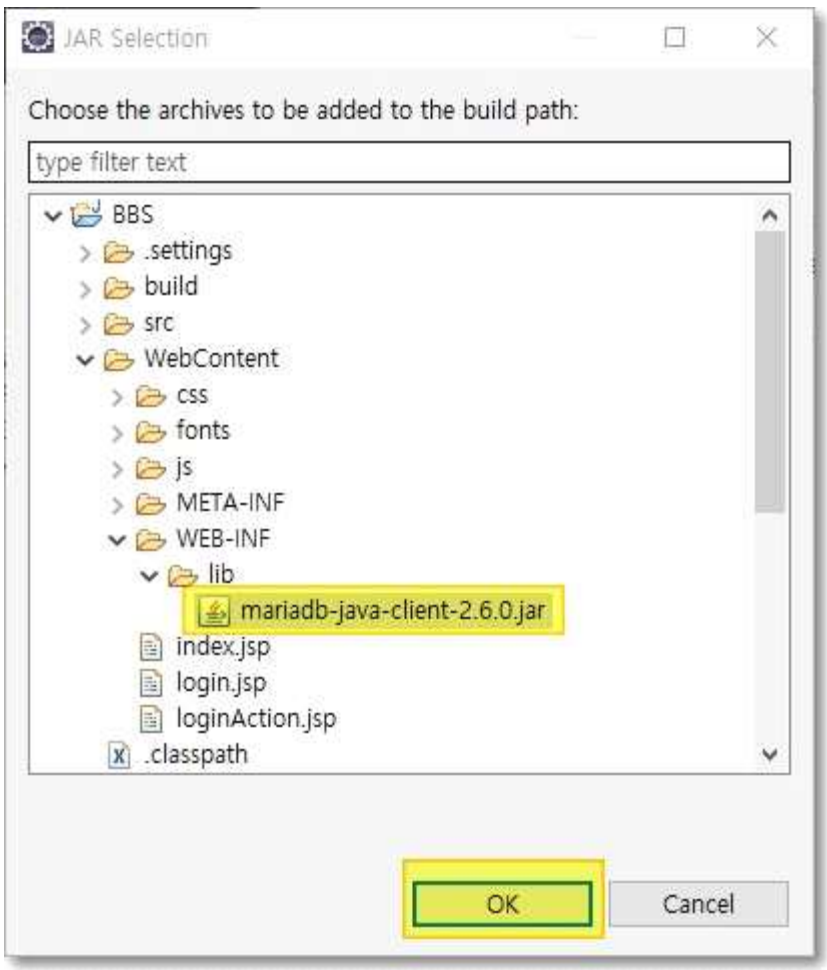
프로젝트 내부에서 [WebContent] - [WEB-INF] - [lib 폴더에 붙여넣기]



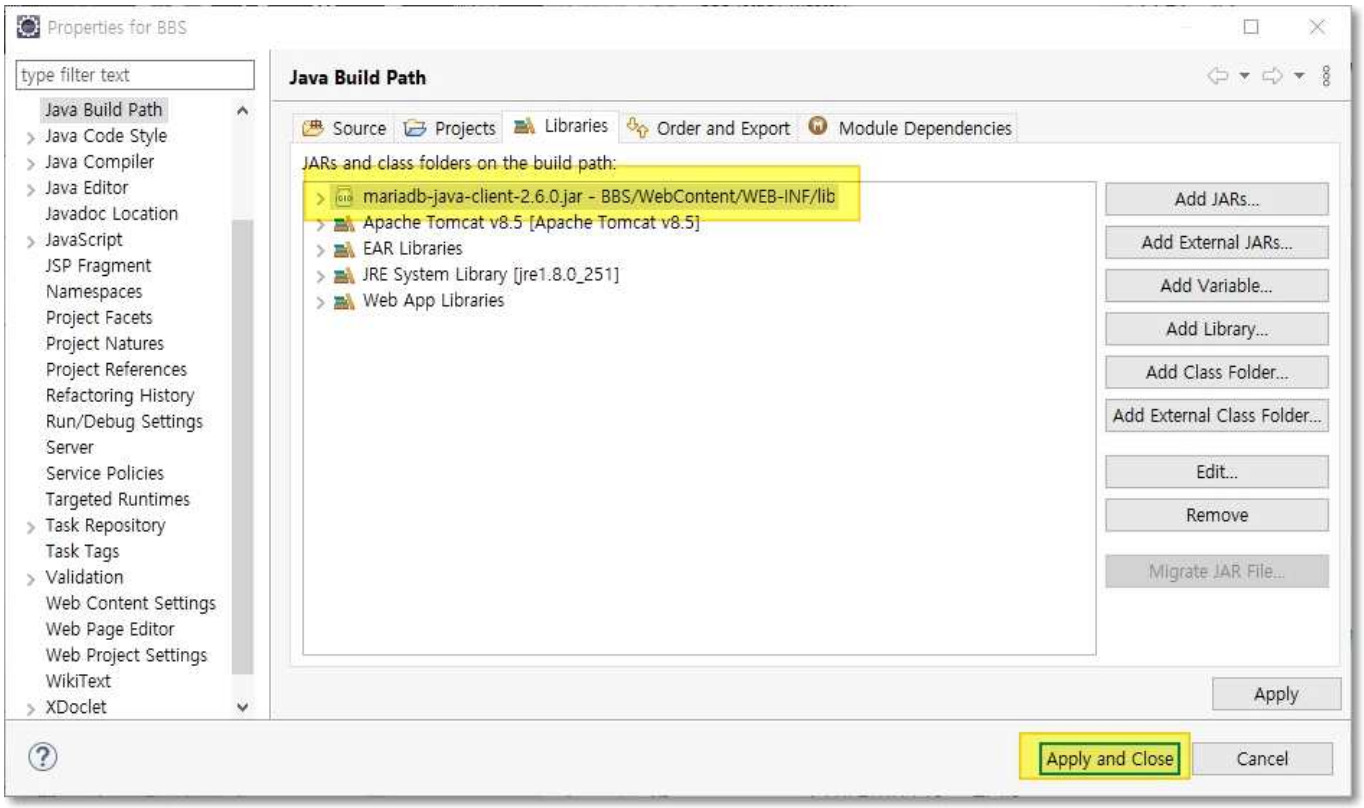
[프로젝트 우클릭] – [Build Path] – [Configure Build Path]



'Add JARs...' 클릭

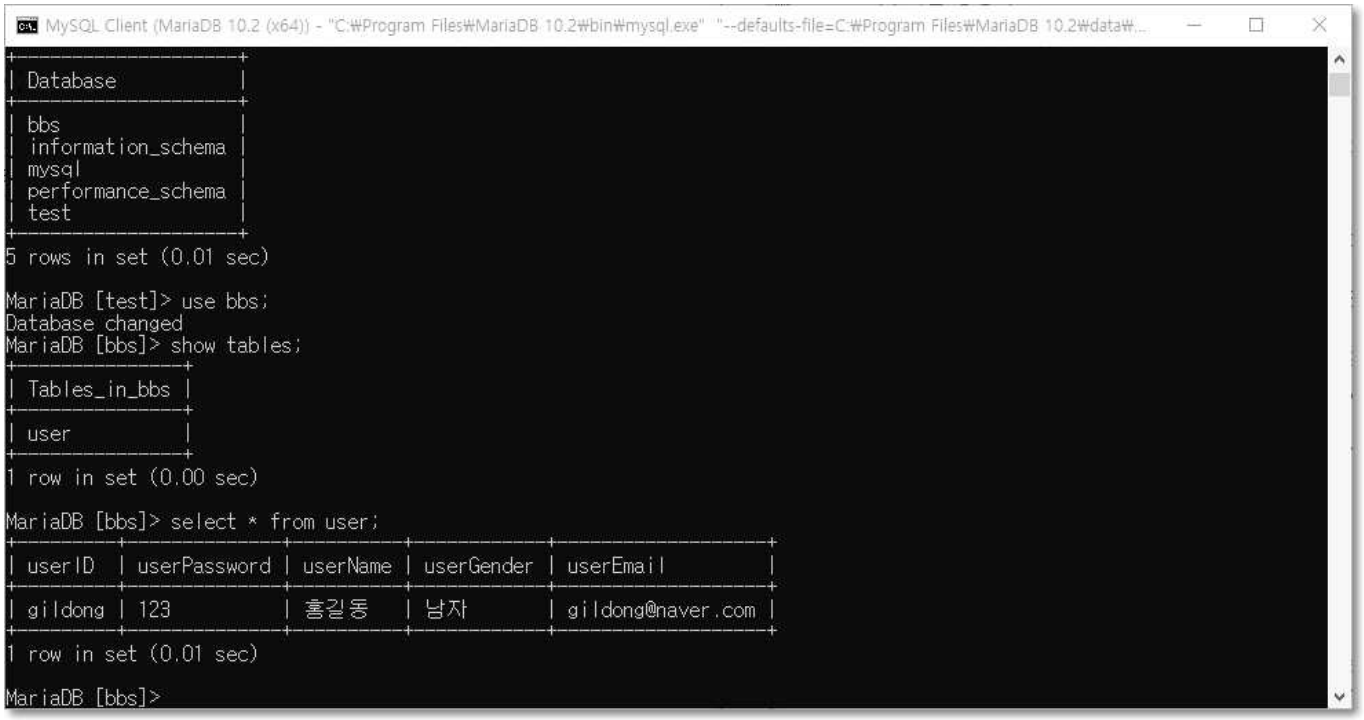


'붙여넣기'했었던 'jar'파일 선택하고 'OK'

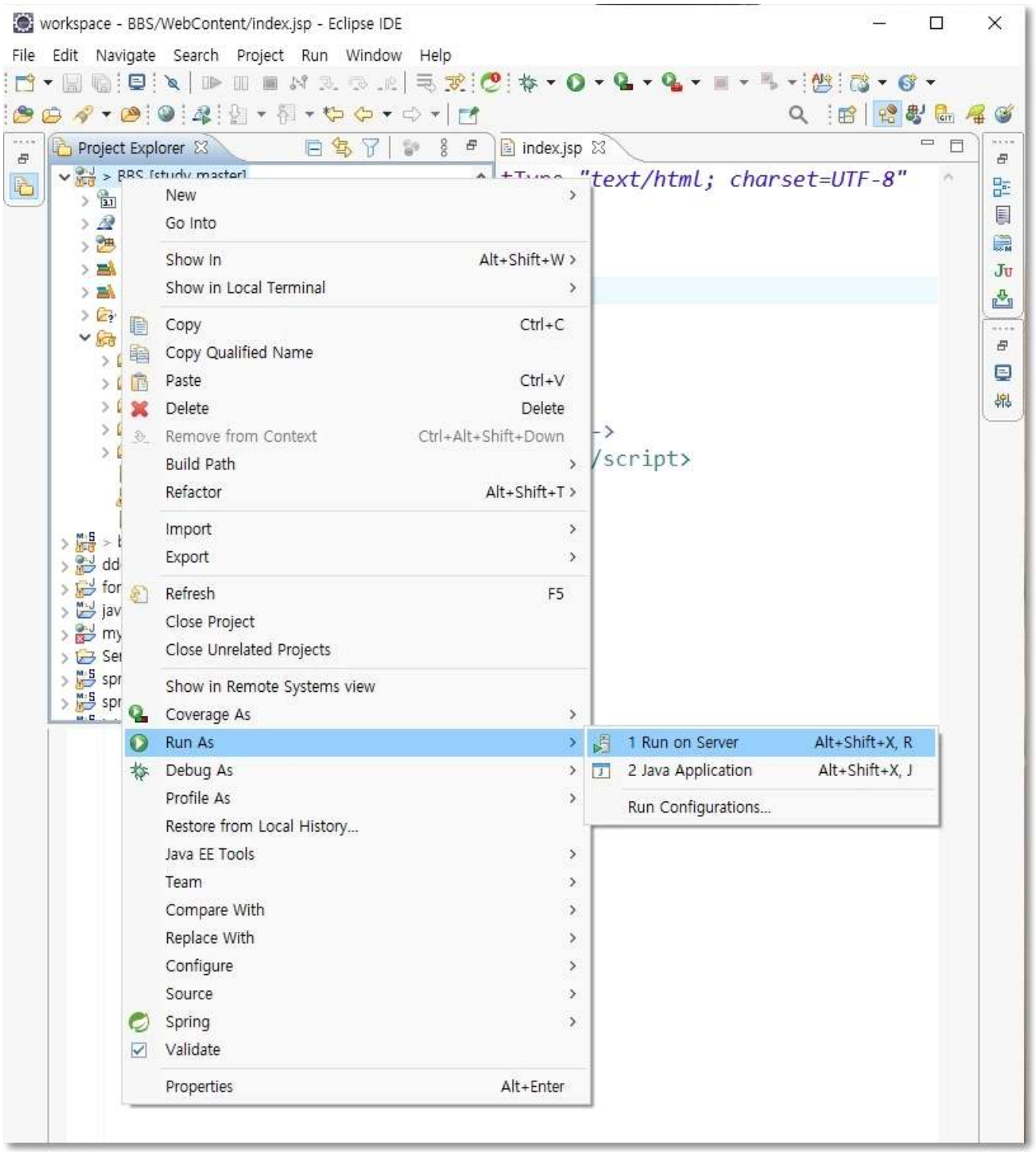


프로젝트에 추가된 것 확인 후'Apply and Close'눌러서 나오기

실행 테스트

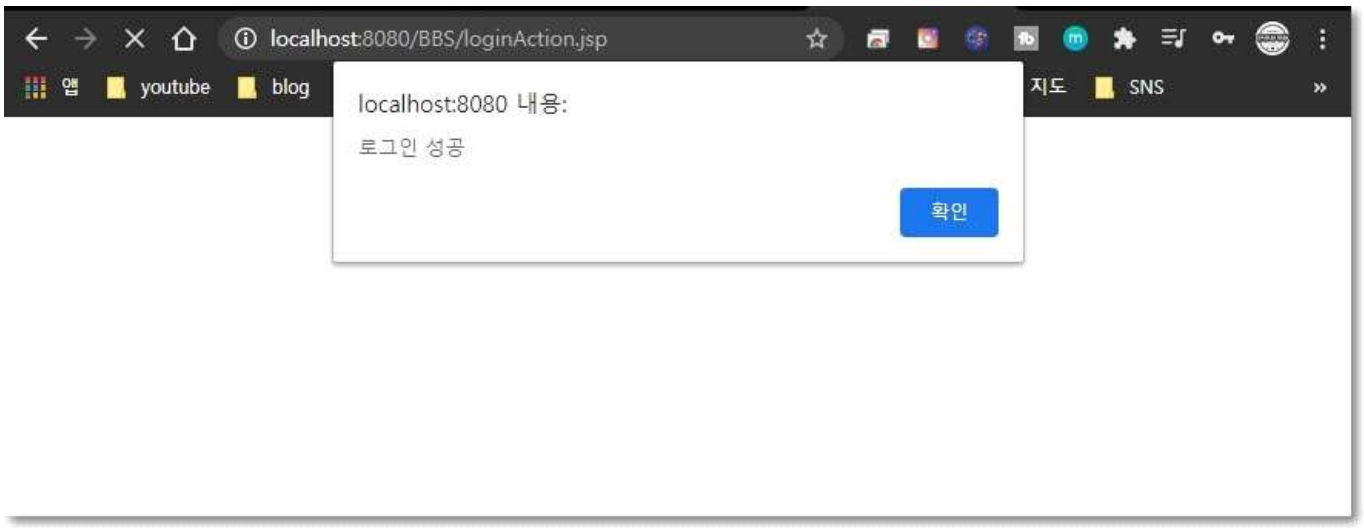


이전에 데이터베이스 구축을 하면서 입력했었던 아이디와 비밀번호를 확인합니다

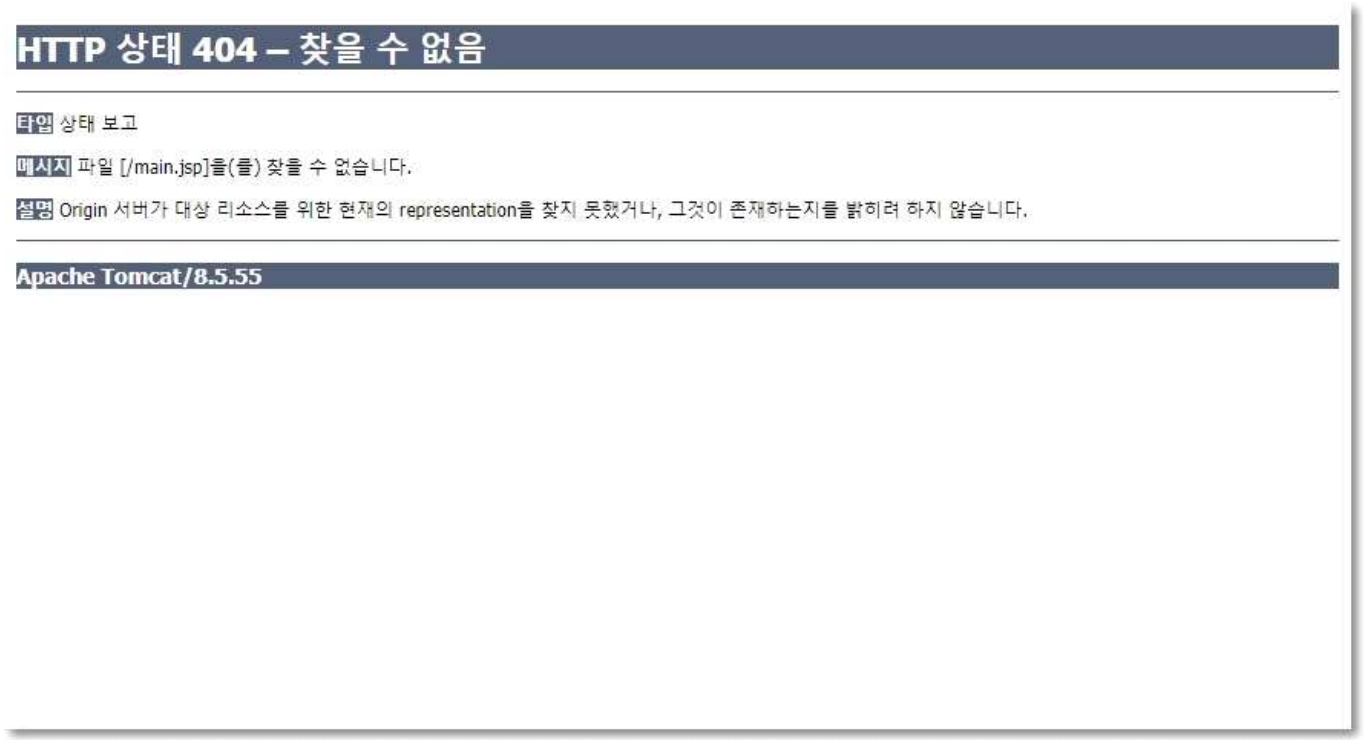


[프로젝트 우클릭] – [Run As] – [Run on Server]

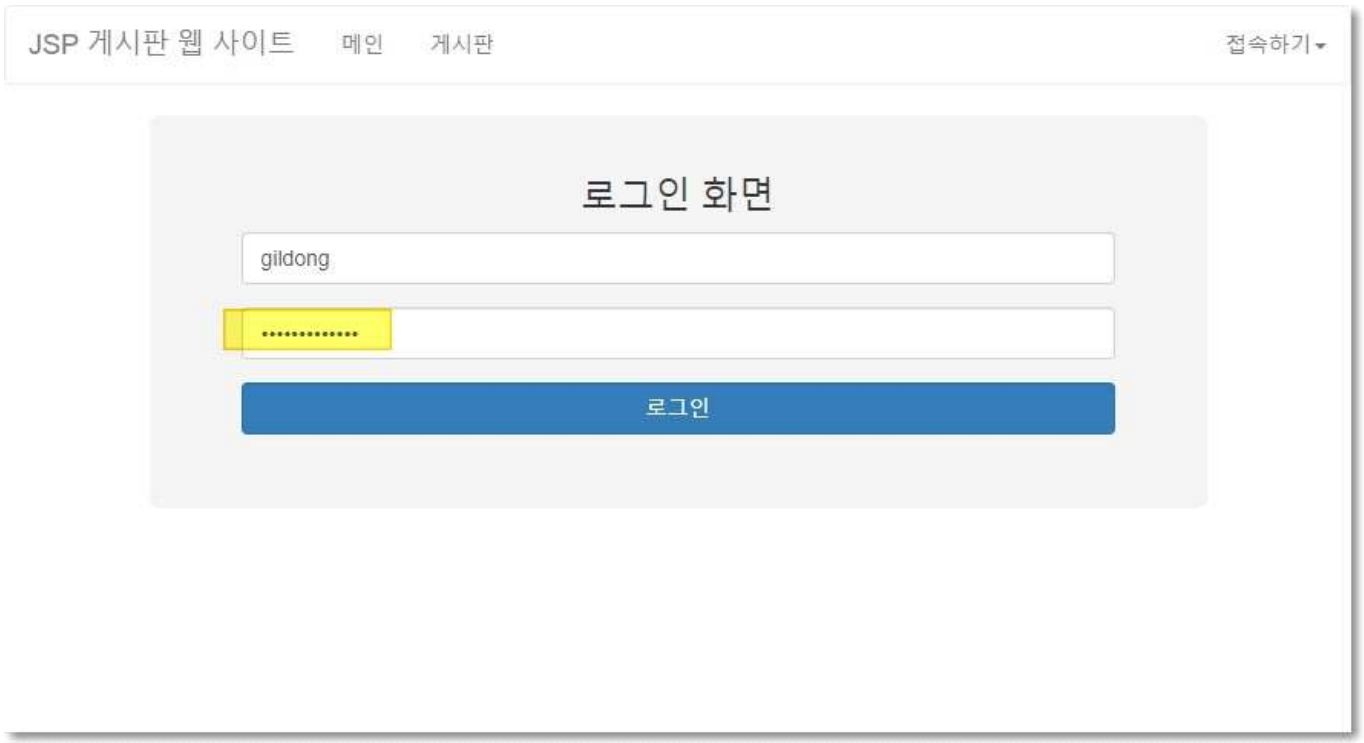
현재 DB에 저장되어 있는 아이디와 비밀번호를 입력해서 테스트해봅니다



아이디와 비밀번호를 모두 맞게 입력했습니다.**'loginAction.jsp'에서 아이디와 비밀번호가 맞으면 '로그인 성공'이라는 알림창을 뜨게끔 자바스크립트를 입력했었습니다.**위의 사진처럼 알림창이 정상적으로 뜨는 것을 확인할 수 있습니다.



그리고 '확인'버튼을 누르면 아마'404에러'가 뜰 것입니다.**아이디와 비밀번호가 모두 맞을 때 '로그인 성공'이라는 알림창을 띄우고 'main.jsp' 페이지로 이동하라는 코드를 입력했는데 아직 'main.jsp' 페이지를 만들지 않았기 때문에 페이지를 찾지 못 해서 당연히 뜨는 오류**이니 넘어가도 됩니다.



이번에는 일부러 비밀번호를 틀리게 입력해보았습니다

아이디는 맞지만 비밀번호가 틀릴 때는**'비밀번호가 틀립니다'라는 알림창을 띄우고 다시 이전 페이지로 돌아가게 하는 자바스크립트 코드를 입력했었습니다.**정상적으로 작동이 되는 것을 볼 수 있습니다. '확인'을 누르면 다시 '로그인 페이지'로 이동할 겁니다.

 # 로그인 기능 구현 끝