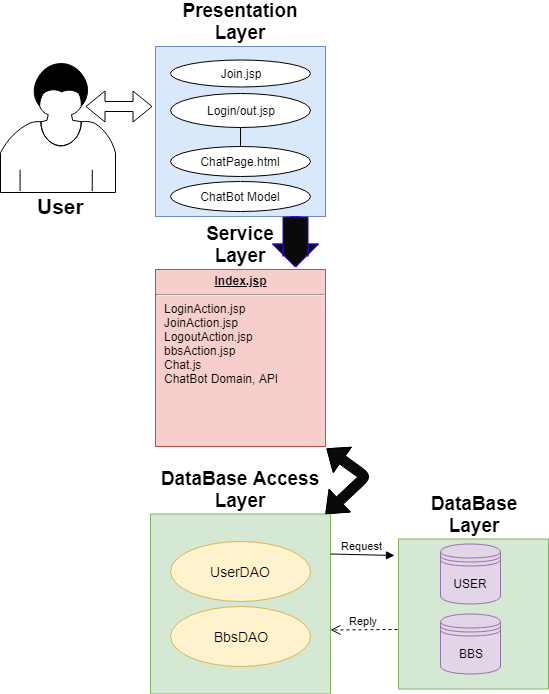
**[Layered Architecture]**

****

**[회원관리, 게시판 기본 환경 설정 & Tool]**

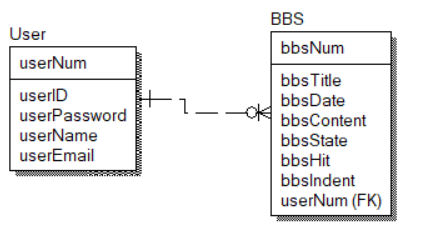
Eclipse Runtime Target: Web Apache Tomcat 8.5

DB: Mysql

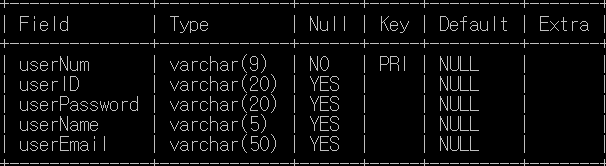
JSP-DB 연동 :mysql jdbc driver

Web Framework: Bootstrap

**[ERD]**

****

<User Table>



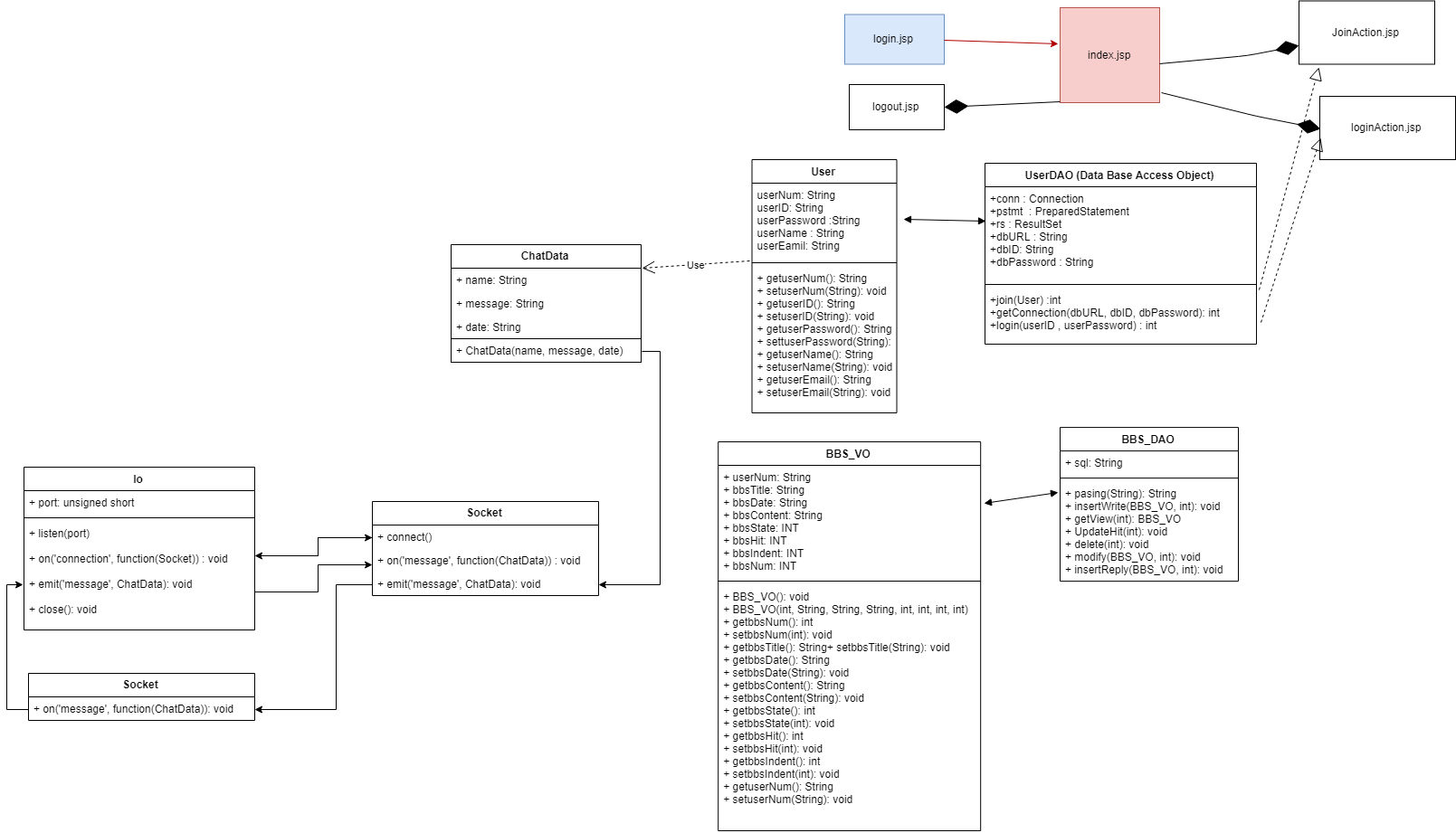
기본 멤버 필드(속성)값은

학번, 아이디, 비밀번호, 학생명, 학생 이메일주소를 의미

서버단 DB에는 모든 학번(userNum)이 사전 입력되어있고

학번이 등록되지 않은 사람은 회원가입할 수 없다.

**[회원관리, 게시판, 채팅 관련 클래스 설계도]**



**[1. Login, Join]**

① User

JSP JAVA object로 여러 JSP 페이지에 사용 될 수 있다.

DB의 UserTable과 동일하게 아이디, 비빌번호, 학생명, 학번을 멤버 변수로 가지고

이에 대한 getter, setter 메소드로 구성된다.

② UserDAO

User가 회원가입 기능을 수행할 때 DB에 유저 정보를 등록하고

로그인 기능을 수행할 때 DB에 등록된 유저 정보와 대조하는 역할을 한다.

성공적으로 회원가입, 로그인이 완료될 경우

.index (메인 화면)으로 이동한다.

-Connection conn : 데이터베이스에 접근하기 위한 객체

-PreparedStatement pstmt : 정해진 SQL문을 이용해서 상태정보(ID)를 가져온다.

-ResultSet rs: 데이터베이스에 접근한 결과를 저장하는 객체

- dbURL: Mysql 서버 주소

- dbID, dbPassword: 데이터베이스에 저장된 ID, 패스워드와 비교할 값

- int getConnection(dbURL, dbID, dbPassword)

mysql driver와 연결하여 DB에 실제적으로 접근할 수 있도록 하는 메소드

- int join(User user)

회원가입을 수행할 메소드로 SQL 쿼리문을 통해 DB에 유저 정보를 저장한다.

로그인 수행결과에 따라 리턴 값을 반환한다

★ 이때, DB에 사전등록되지 않은 학번의 유저는 가입할 수 없다. (학번 Check Box 활용)

- int login(userID, userPassword)

로그인 기능을 수행할 메소드로 SQL 쿼리문을 통해 DB에 등록된 유저 정보와 대조한다.

로그인 수행결과에 따라 리턴 값을 반환한다

ex) return 값 -> 1: 로그인 성공/ 0: 비밀번호 불일치 / -1: ID가 존재하지 않음.

-2: 데이터베이스 오류

loginAction.jsp: userDAO 객체를 생성하여 아이디와 비밀번호를 입력받는다.

로그인 실패시 -> 경고메시지를 띄우고 login.jsp로 되돌아간다.

로그인 성공시 -> index.jsp로 이동한다.

joinAction.jsp: userDAO 객체를 생성하여 학번, 아이디, 비밀번호, 이메일을 입력받는다.

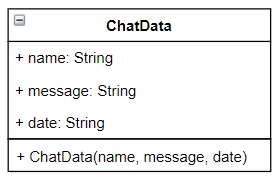
회원가입 실패시 -> 경고메시지를 띄우고 join.jsp로 돌아간다.

회원가인 성공시 -> index.jsp로 이동한다.

logoutAction: 세션을 종료시켜 로그인된 유저를 로그아웃시키고 login.jsp로 이동한다.

**[2.Chatting]**

**1. ChatData : Object – 서버와 클라이언트가 message 이벤트로 주고 받을 데이터**

****

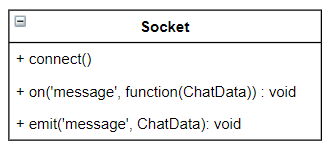
name : 채팅 참여자 이름

message : 보내고자 하는 채팅 내용

date : 채팅 보낸 시간 & 날짜

ChatData(name, message, date) : ChatData의 생성자

**2. Socket : Object – 클라이언트 소켓 객체**



connect() : 서버 소켓에 연결

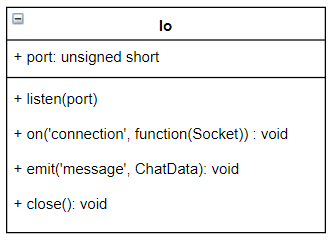
on(‘message’, function(ChatData))

* message : 사용자 정의 이벤트
* function(ChatData) : message 이벤트를 서버로부터 받았을 때, 화면에 Data의 내용(name, message, date)를 보여주는 함수

emit(‘message’, ChatData)

* message : 사용자 정의 이벤트, message 이벤트를 서버에 전달
* ChatData : message 이벤트를 서버에 전달할 때 함께 전달할 데이터

**3. Io : Object – 서버 소켓 객체**



port : 웹 서버의 포트번호

listen() : 서버 소켓 생성 및 실행

on(‘connection’, function(Socket))

* connection : 클라이언트가 연결할 때 발생하는 이벤트
* function(Socket) : 접속한 클라이언트의 정보를 가진 소켓 객체 생성

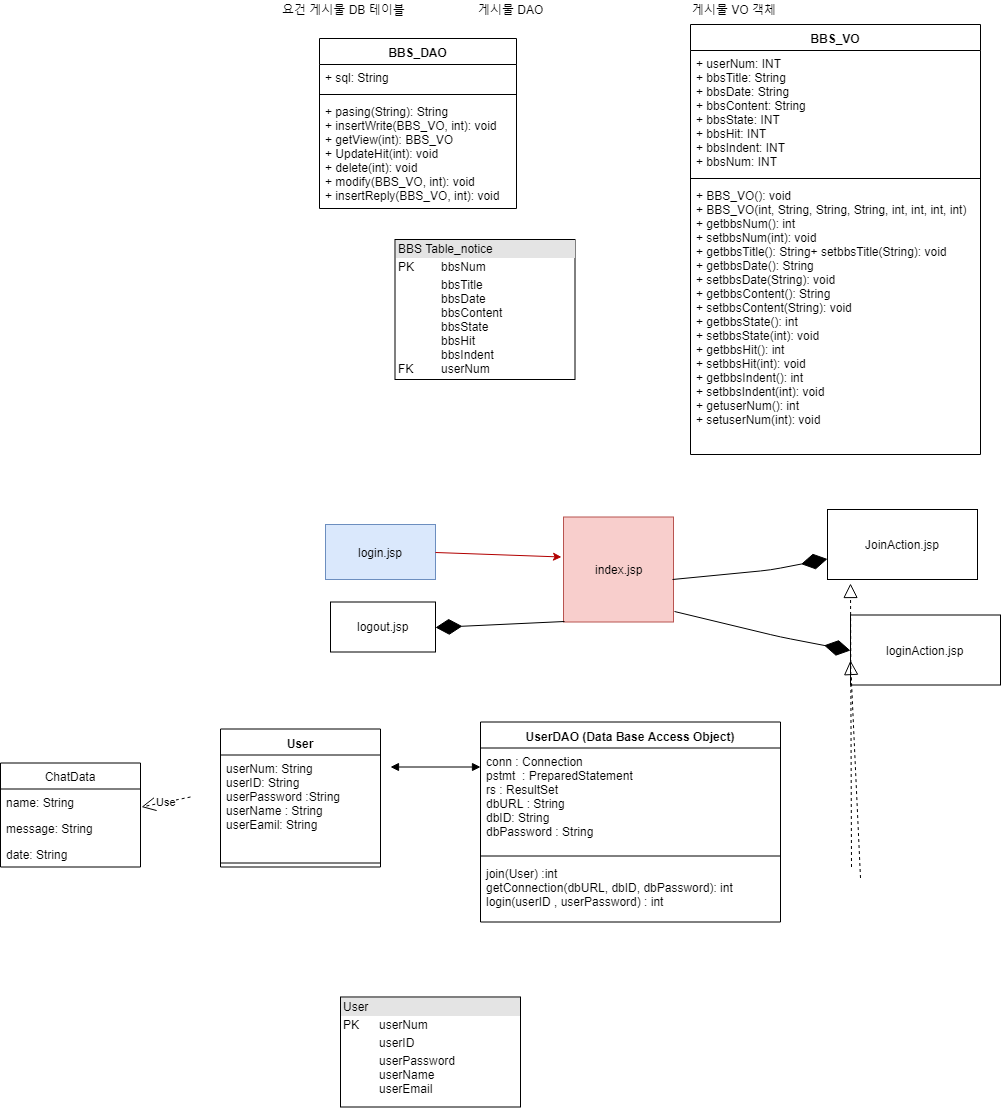
emit(‘message’, ChatData)

* message : 사용자 정의 이벤트, message 이벤트를 서버에 접속한 모든 클라이언트에게 전달
* ChatData : 한 클라이언트에게서 받은 데이터(name, message, date)를 이벤트를 발생시킨 클라이언트를 포함한 모든 클라이언트에게 전달

close() : 서버 소켓 종료

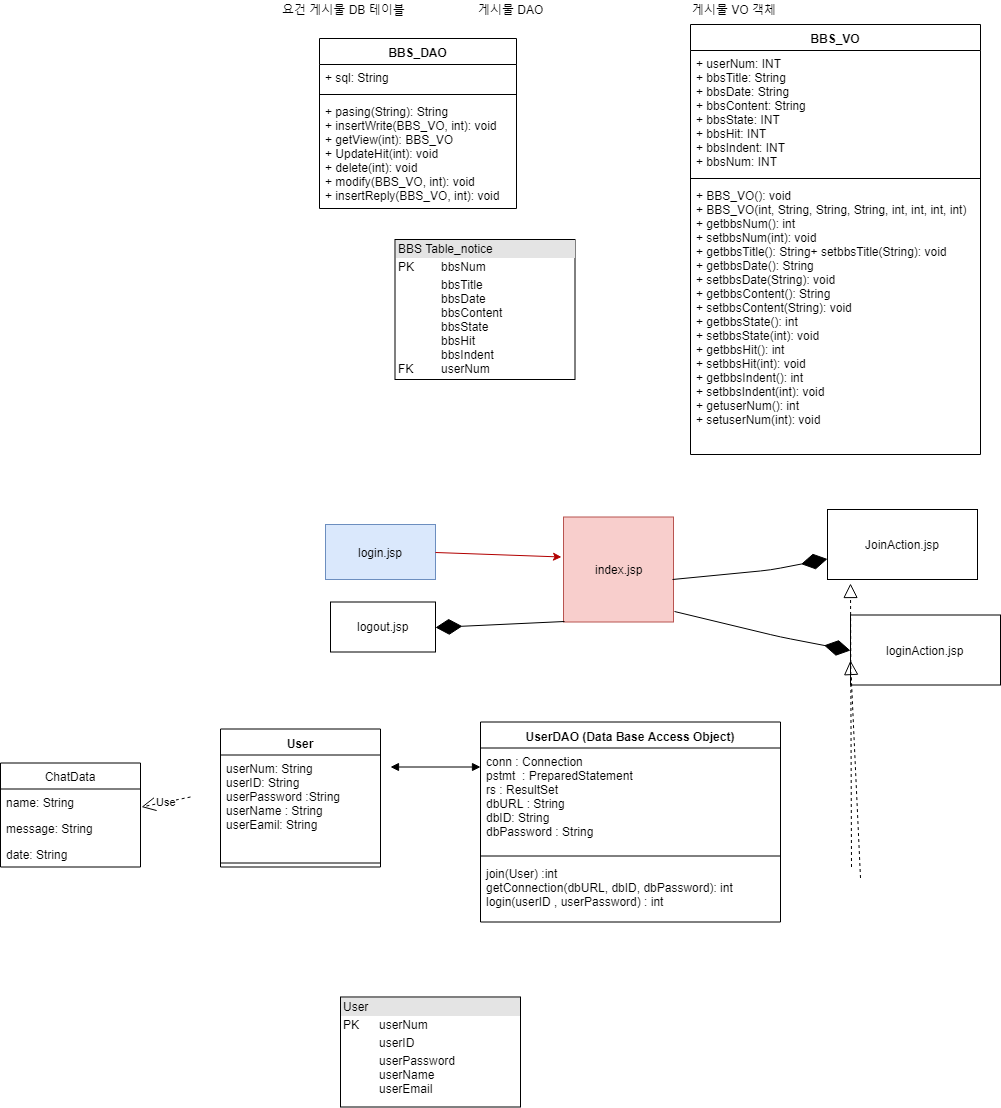
**[3. Bulletin Board]**

**1. BBS\_VO: 게시물 객체**

****

DB의 게시판 Table과 동일하게 게시물 번호, 제목, 작성일자, 내용, 상태 (삭제여부), 조회수, 답글수준, 유저 정보를 멤버 변수로 가지고 이에 대한 getter, setter 메소드로 구성되었다.

**2****. BBS\_DAO: 게시물 작업을 실질적으로 처리. DB와 연동**



게시물을 조회, 작성, 수정, 삭제할 때 DB에 정보를 등록하고 수정하는 역할을 한다.

+ pasing(String): String 인코딩 파싱 함수

DB데이터에 저장된 String을 웹 인코딩에 맞게 파싱한다.

+ insertWrite(BBS\_VO, int): void 게시물 삽입 함수

작성할 게시물 정보와 지금까지의 게시물 개수를 입력받아 DB의 max+1에 게시물을 삽입한다.

+ getView(int): VO 게시물 로드 함수

게시물 번호 (인덱스)에 맞는 게시물을 로드한다.

+ UpdateHit(int): void 게시물 조회수 업데이트

게시물 열람시, 번호 (인덱스)에 맞는 게시물의 조회수를 높인다.

+ delete(int): void 게시물 삭제

게시물 번호 (인덱스)에 맞는 게시물을 삭제한다. (게시물 상태 변수 변경. Soft Delete방식)

+ modify(BBS\_VO, int): void 게시물 수정 함수

수정할 게시물 번호와 수정 정보를 이용해 게시물을 수정한다.

+ insertReply(BBS\_VO, int): void 답글 삽입함수

게시물 하단에 답글을 삽입한다. (답글 수준 +1 하여 들여쓰기)

**3. jsp 연결**

List.jsp 해당 게시판 테이블을 통해 게시글 목록을 보여준다.

delete.jsp BBS\_DAO의 delete을 이용하여 게시물을 삭제하는 기능을 한다.

성공시 delete\_ok로 이동한다.

modify.jsp BBS\_DAO의 modify를 이용하여 게시물을 수정한다.

성공시 modify\_ok로 이동한다.

view.jsp list에서 게시물을 클릭하였을 때 게시물을 보여주는 기능을 한다.

BBS\_DAO의 pasing, getView, UpdateHit을 이용한다.

write.jsp BBS\_DAO의 insertWrite를 이용하여 게시물을 작성한다.

성공시 write\_ok로 이동한다.

reply.jsp BBS\_DAO의 insertReply를 이용해 답글을 작성한다.

성공시 reply\_ok로 이동한다.

**[4. Chatbot]**

스크린샷이(가) 표시된 사진

자동 생성된 설명

* Class Domain: 챗봇의 기본단위인 도메인생성과 삭제에 관한 클래스

1. void CreateDomain(): 도메인을 생성한다

* String DomainName: domain 이름을 저장하는 변수
* String DomainCode: domain코드를 저장하는 변수
* String Language: 사용할 자연어 변수

1. boolean Manage(String DomainName): 해당 도메인의 대화의 작업을 관리한다. 대화가 잘 업데이트 되면 true를 error가 발생하면 false를 return한다.

* boolean check

1. void DeleteDomain(): 도메인을 삭제한다.

* class Dialog: 챗봇의 대화 생성과

1. void CreateDialog():

* String DialogName: 대화 생성의 이름 저장하는 변수
* String DialogType: 사용자 발화 유형의 상태를 정의하는 변수

1. void RegisterA(): 대화의 답변을 등록한다.

* StringA: 답변들을 저장할 array

1. void RegisterQ(): 사용자가 질문할 것이라 예상되는 질문들을 등록한다.

* StringQ:: 질문을 저장할 array

1. void GetQ(): 실제 사용자에게 질문을 받는다

* String Q: 사용자가 한 질문을 저장할 변수

1. String GiveA(String Q): 사용자에게 받은 질문을 가지고 그에 해당하는 답변을 준다.

* class Build: 챗봇을 학습하는 클래스

1. void Build(String Adjust): 챗봇의 대화 모델을 빌드 한다.

* String Adjust: 빌드 내용 수정할 것이 있으면 개발자로부터 받는다. 없다면 null.

1. String BuildState(): 현재 빌드 되고 있는 현황 확인

* Create\_API: API를 생성하는 클래스

1. void API\_gateway(): product에서 project를 만드는 메소드

* String ProductName: 생성할 product이름

1. void Create\_API(): API를 생성하는 메소드
2. void Stage\_API(): stage API를 생성

* String StageName: 만들 stage이름을 저장하는 변수
* String Endpoint: stage의 endpoint저장하는 변수
* Invoke\_URL

1. void Copy\_IU(): invoke url을 복사하는 메소드

* String Urladdress: 복사된 url의 주소를 저장하는 변수
* Register\_URL: url을

1. void register(String Urladdress,Endpoint): invoke URL을 stage endpoint에 등록하는 메소드

* String StageInvokeUrl: 생성한 stage의 invoke url을 복사하여 보관하는 변수