

웹프로젝트 기말고사 (20241519 조예성)

쿠키는 (Key)랑 (Value)로 구성된다

쿠키의 최대 크기: 4kb

쿠키 최대 개수: 3000개

DB 접속을 위하여 필요한 5가지: ip, port, id, pw, instance

예외처리 2종류: 혼자 처리, 호출한 대상한테 throw

데이터를 처리하는 객체: Data Access Object

데이터 저장하는 객체: Data Transfer Object

한번에 처리해야 하는 업무 단위: 트랜잭션

트랜잭션의 단위: 서비스

```
MemberDao memberDao = new MemberDao();
List<MemberDto> memberList = memberDao.getMemberList();

int totalCount = memberList.size();
int i = 0;
for(MemberDto member : memberList) {
    StringBuilder sb = new StringBuilder();
```

```
public static Connection getConnection() {
    try {
        //디비저장- 드라이버클래스 로드
        //Class.forName("org.mariadb.jdbc.Driver");
        Class.forName("com.mysql.cj.jdbc.Driver");

    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }

    //DB 연결객체생성
    String host = "223.194.169.121";
    int port = 3306;
    String id = "jsp_user";
    String pwd = "1234";
    String dbInstance = "jsp_db";
    //String url = String.format("jdbc:mariadb://%s:%d/%s", host, port, dbInstance);
    String url = String.format("jdbc:mysql://%s:%d/%s", host, port, dbInstance);
    Connection connection = null;

    try {
        connection = DriverManager.getConnection(url, id, pwd);
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return connection;
}
```

```

public MemberDto getMember(String userId) {

    MemberDto member = getMemberList().stream()
        .filter( e-> e.getUserId().equals(userId))
        .findFirst()
        .orElse(null);

    return member;
}

```

```

public boolean addMember(MemberDto member) {

    boolean result = false;

    Connection connection = DbConnector.getConnection();

    //쿼리실행
    String sql = "insert into member (user_id, user_name, password) values (?, ?, ?)";
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(1, member.getUserId());
        preparedStatement.setString(2, member.getUserName());
        preparedStatement.setString(3, member.getPassword());
        int affectedRow = preparedStatement.executeUpdate();
        if(affectedRow > 0) {
            result = true;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return result;
}

```

SELECT 문 실행: executeQuery

이외 나머지 문 실행: executeUpdate

preparedStatement 중 ? 에 추가하는 메소드: setString(index, value);

```

@Getter
@Setter
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Override
public class MemberDto {
    String userId;
    String userName;
    String password;
    Date createDt;
    Date updateDt;
}

```

```
request.setCharacterEncoding("utf-8");

String writer = request.getParameter("writer");
String subject = request.getParameter("subject");
String contents = request.getParameter("contents");
String ipAddr = request.getRemoteAddr();

BoardDto board = BoardDto.builder()
    .writer(writer)
    .subject(subject)
    .contents(contents)
    .ipAddr(ipAddr)
    .build();
BoardDao boardDao = new BoardDao();
boolean result = boardDao.addBoard(board);
```

```
String userId = request.getParameter("user_id");

MemberDao memberDao = new MemberDao();
MemberDto member = memberDao.getMember(userId);
```

```
//로그인 정보를 세션에 저장!!!
session.setAttribute("userId", userId);
session.setAttribute("userName", member.getUserName());
//로그인한 사람의 이름정보도 세션에 씀.

//하고, 페이지를 메인으로 보냄
response.sendRedirect("/");
```

```
MemberDao memberDao = new MemberDao();
int memberCount = memberDao.getMemberList().size();

BoardDao boardDao = new BoardDao();
int boardCount = boardDao.getBoardList().size();
// memberCount 랑 bbsContentsCountn 빌딩
return DashboardDto.builder()
    .memberCount(memberCount)
    .bbsContentsCount(boardCount)
    .build();
```

```
//LoginService loginService = new FakeLoginServiceImpl();
LoginService loginService = new DbLoginServiceImpl();
// post 방식에서 가져오기
String userId = request.getParameter("user_id");
String password = request.getParameter("password");

MemberDto member = loginService.login(userId, password);
```