[ 12 ] JDBC Template

1. JDBC template를 이용한 반복코드 줄이기 : JDBC를 이용하는 가장 큰 이유
   * DAO객체를 이용해서 DataBase의 데이터를 이용 합니다.
   * 이때 매번 같은 동작을 반복하는 부분이 있습니다.(드라이버 로드, 커넥션 생성 및 DB연결, SQL실행, 자원해제)
   * 이런 반복적인 작업을 스프링에서는 간단하게 처리 할 수 있습니다
     1. JDBC 드라이버 로드 – DriverManager

JdbcTemplate

* + 1. 데이터 베이스 연결 – Connection **한방에해결**
    2. SQL문 실행 – PreparedStatement, ResultSet
    3. DB 연결 해제 - \*.close()

1. Spring 빈을 이용한 코드 간소화
   * Spring빈을 이용하여 DataBase관련 클래스를 간소화한다.
   * 스프링 설정파일에서 JdbcTeplate 빈을 생성하여 사용한다

Java에서JdbcTemplate빈 이용

JdbcTemplate빈

Datasource빈

1. JDBC를 이용한 list, Insert, update, delete 처리하기

(예제)

Pom.xml 에 JDBC Template 사용을 위한 dependencies 추가

<!-- JDBC Template -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.1.4.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.2.9.RELEASE</version>

</dependency>

**Servlet-context.xml이나 root-context.xml 어디다 추가해도 무방하나 root-context.xml에**

<context:property-placeholder

location=*"classpath:META-INF/property/db.properties"*/>

<bean name=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"${db.driverClassName}"*/>

<property name=*"url"* value=*"${db.url}"*/>

<property name=*"username"* value=*"${db.username}"*/>

<property name=*"password"* value=*"${db.password}"*/>

</bean>

<bean name=*"template"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"dataSource"*/>

</bean>

(db.properties)

db.driverClassName=oracle.jdbc.driver.OracleDriver

db.url=jdbc:oracle:thin:@localhost:1521:xe

db.username=scott

db.password=tiger

이렇게 설정만 해놓는다고 바로 template을 쓸 수 있는 게 아니다. Controller의 template에 bean이 담겨야 하는데 그러기 위해서 자동으로 setter를 호출해서 jdbcTemplate이 들어오도록 자동 autoscan할 때 @AutoWired 주면 자동으로 빈이 생성되어 이쪽으로 setting되어 할당된다.

**public** JdbcTemplate template;

@Autowired

**public** **void** setTemplate(JdbcTemplate template) {

**this**.template = template;

}

위의 templeate을 어디서든지 쓸 수 있게 패키지를 하나 만든다.

**package** com.ch.mvcboard.util;

**import** org.springframework.jdbc.core.JdbcTemplate;

**public** **class** Constant {

**public** **static** JdbcTemplate *template*;

}

이렇게 클래스안에 static 변수를 두고 다음을 controller에 추가하면

**private** BService service;

**public** JdbcTemplate template;

@Autowired

**public** **void** setTemplate(JdbcTemplate template) {

**this**.template = template;

Constant.*template* = **this**.template;

}

Constant.*template라는 이름으로 어디서든 쓸 수 있다*

**<예제 소스>**

**BDao.java**

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** org.springframework.jdbc.core.BeanPropertyRowMapper;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.core.PreparedStatementCreator;

**import** org.springframework.jdbc.core.PreparedStatementSetter;

**import** com.lec.ch12.dto.BoardDto;

**import** com.lec.ch12.util.Constant;

**public** **class** BoardDao {

**public** **static** **final** **int** ***FAIL*** = 0;

**public** **static** **final** **int** ***SUCCESS*** = 1;

**private** JdbcTemplate template;

**private** **static** BoardDao *instance* = **new** BoardDao();

**public** **static** BoardDao getInstance() {

**return** *instance*;

}

**private** BoardDao() {

template = Constant.*template*;

}

// 글목록

**public** ArrayList<BoardDto> boardList(){

String sql = "SELECT \* FROM MVC\_BOARD ORDER BY BGROUP DESC, BSTEP";

ArrayList<BoardDto> dtos = (ArrayList<BoardDto>)template.query(sql, **new** BeanPropertyRowMapper<BoardDto>(BoardDto.**class**));

**return** dtos;

}

**public** ArrayList<BoardDto> boardList(**final** **int** startRow, **final** **int** endRow){

String sql = "SELECT \* FROM (SELECT ROWNUM RN, A.\* " +

" FROM (SELECT \* FROM MVC\_BOARD ORDER BY BGROUP DESC, BSTEP) A)" +

" WHERE RN BETWEEN ? AND ?";

ArrayList<BoardDto> dtos = (ArrayList<BoardDto>) template.query(sql, **new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement pstmt) **throws** SQLException {

pstmt.setInt(1, startRow);

pstmt.setInt(2, endRow);

}

}, **new** BeanPropertyRowMapper<BoardDto>(BoardDto.**class**));

**return** dtos;

}

// 글 갯수

**public** **int** getBoardTotCnt() {

String sql = "SELECT COUNT(\*) FROM MVC\_BOARD";

**return** template.queryForObject(sql, Integer.**class**);

}

//원글쓰기

**public** **int** write(**final** String bname, **final** String btitle, **final** String bcontent, **final** String bip) {

// String sql = "INSERT INTO MVC\_BOARD "

// + "(bID, bNAME, bTITLE, bCONTENT, bGROUP, bSTEP, bINDENT, bIP)" +

// " VALUES (MVC\_BOARD\_SEQ.NEXTVAL, ?, ?, ?, " +

// " MVC\_BOARD\_SEQ.CURRVAL, 0,0, ?)";

// return template.update(sql, new PreparedStatementSetter() {

// @Override

// public void setValues(PreparedStatement pstmt) throws SQLException {

// pstmt.setString(1, bname);

// pstmt.setString(2, btitle);

// pstmt.setString(3, bcontent);

// pstmt.setString(4, bip);

// }

// });

**return** template.update(**new** PreparedStatementCreator() {

String sql = "INSERT INTO MVC\_BOARD "

+ "(bID, bNAME, bTITLE, bCONTENT, bGROUP, bSTEP, bINDENT, bIP)" +

" VALUES (MVC\_BOARD\_SEQ.NEXTVAL, ?, ?, ?, " +

" MVC\_BOARD\_SEQ.CURRVAL, 0,0, ?)";

@Override

**public** PreparedStatement createPreparedStatement(Connection conn) **throws** SQLException {

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, bname);

pstmt.setString(2, btitle);

pstmt.setString(3, bcontent);

pstmt.setString(4, bip);

**return** pstmt;

}

});

}

// hit수 올리기

**private** **void** hitUp(**int** bid) {

String sql = "UPDATE MVC\_BOARD SET BHIT = BHIT+1 WHERE BID="+bid;

template.update(sql);

}

// 글 상세보기(글 상세볼 때 조회수도 올림)

**public** BoardDto contentView(**final** **int** bid) {

hitUp(bid);

String sql = "SELECT \* FROM MVC\_BOARD WHERE BID="+bid;

**return** template.queryForObject(sql, **new** BeanPropertyRowMapper<BoardDto>(BoardDto.**class**));

}

// bid로 dto 가져오기 (답글쓰기 + 수정)

**public** BoardDto modifyView\_replyView(**int** bid) {

String sql = "SELECT \* FROM MVC\_BOARD WHERE BID="+bid;

**return** template.queryForObject(sql, **new** BeanPropertyRowMapper<BoardDto>(BoardDto.**class**));

}

// 글 수정하기

**public** **int** modify(**final** BoardDto bDto) {

**return** template.update(**new** PreparedStatementCreator() {

String sql = "UPDATE MVC\_BOARD SET BNAME=?, " +

" BTITLE = ?, " +

" BCONTENT = ?, " +

" BIP = ?" +

" WHERE BID=?";

@Override

**public** PreparedStatement createPreparedStatement(Connection con) **throws** SQLException {

PreparedStatement pstmt = con.prepareStatement(sql);

pstmt.setString(1, bDto.getBname());

pstmt.setString(2, bDto.getBtitle());

pstmt.setString(3, bDto.getBcontent());

pstmt.setString(4, bDto.getBip());

pstmt.setInt(5, bDto.getBid());

**return** pstmt;

}

});

}

// 글 삭제

**public** **int** delete(**final** **int** bid) {

String sql = "DELETE FROM MVC\_BOARD WHERE BID=?";

**return** template.update(sql, **new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement pstmt) **throws** SQLException {

pstmt.setInt(1, bid);

}

});

}

// 답변글 쓰기 전 작업 : stepA

**private** **void** preReplyStepA(**final** **int** bgroup, **final** **int** bstep) {

String sql = "UPDATE MVC\_BOARD SET BSTEP = BSTEP+1 WHERE BGROUP=? AND BSTEP>?";

template.update(sql, **new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement pstmt) **throws** SQLException {

pstmt.setInt(1, bgroup);

pstmt.setInt(2, bstep);

}

});

}

// 답변글 쓰기

**public** **int** reply(**final** BoardDto bDto) { // bname, btitle, bcontent, bip : 답변글에 대한 정보

// bgroup, bstep, bindent : 원글에 대한 정보

preReplyStepA(bDto.getBgroup(), bDto.getBstep());

String sql = "INSERT INTO MVC\_BOARD" +

" (BID, BNAME, BTITLE, BCONTENT, BGROUP, BSTEP, BINDENT, BIP) " +

" VALUES (MVC\_BOARD\_SEQ.NEXTVAL, ?,?,?, ?, ?, ?, ?)";

**return** template.update(sql, **new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement pstmt) **throws** SQLException {

pstmt.setString(1, bDto.getBname());

pstmt.setString(2, bDto.getBtitle());

pstmt.setString(3, bDto.getBcontent());

pstmt.setInt(4, bDto.getBgroup());

pstmt.setInt(5, bDto.getBstep()+1);

pstmt.setInt(6, bDto.getBindent()+1);

pstmt.setString(7, bDto.getBip());

}

});

}

}

**public** **class** BWriteService **implements** BService {

@Override

**public** **void** execute(Model model) {

Map<String, Object> map = model.asMap(); // model안의 attribute값을 추출하기 위해 map화

BoardDto bDto = (BoardDto)map.get("bDto"); // bname, btitle, bcontent

HttpServletRequest request = (HttpServletRequest)map.get("request");

bDto.setBip(request.getRemoteAddr());

BoardDao boardDao = BoardDao.*getInstance*();

model.addAttribute("writeResult", boardDao.write(bDto.getBname(), bDto.getBtitle(), bDto.getBcontent(), bDto.getBip()));

}

}