[ 13 ] JDBC

1. JDBC를 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>

**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**

**public** **class** BDao {

**private** JdbcTemplate template;

**public** BDao() {

template = Constant.*template*;

}

// SELECT \* FROM MVC\_BOARD 이것을 수행하는 list()

**public** ArrayList<BDto> list(){

ArrayList<BDto> dtos = **null**;

String query = "SELECT \* FROM MVC\_BOARD "

+ "ORDER BY bGroup DESC, bSTEP";

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

**return** dtos;

}

// SELECT \* FROM MVC\_BOARD의 페이징 이것을 수행하는 list(int, int)

**public** ArrayList<BDto> list(**int** startRow, **int** endRow){

String query = "SELECT \* FROM "

+"(SELECT ROWNUM RN, a.\* FROM "

+" (SELECT \* FROM MVC\_BOARD "

+" ORDER BY bGroup DESC, bSTEP) a)" +

" WHERE RN BETWEEN "+startRow+" AND "+endRow;

**return** (ArrayList<BDto>) template.query(query, **new** BeanPropertyRowMapper<BDto>(BDto.**class**));

}

// 글 insert write(BDto)

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

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

@Override

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

String query = "INSERT INTO MVC\_BOARD "

+ "(bId, bName, bTitle, bContent, bGroup, bStep, bIndent)"

+ " VALUES "

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

+ " 0, 0)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, bname);

pstmt.setString(2, btitle);

pstmt.setString(3, bcontent);

**return** pstmt;

}

});

}

//전체 글 갯수를 리턴하는 method

**public** **int** getTotCnt() {

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

**return** template.~~queryForInt~~(query);

}

// 해당 글번호(bid)로 hit수 하나 올리는 method

**private** **void** upHit(**final** **int** bid) {

String query = "UPDATE MVC\_BOARD SET BHIT = BHIT+1 "

+ "WHERE BID=?";

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

@Override

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

pstmt.setInt(1, bid);

}

});

}

// 해당 글번호(bid)로 BDto를 리턴하는 method

**public** BDto contentView(**int** bid) {

upHit(bid);

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

**return** template.queryForObject(query, **new** BeanPropertyRowMapper<BDto>(BDto.**class**));

}

// 해당 글번호(bid)로 bname, btitle, bcontent를 수정하는 method

**public** **int** modify(**final** **int** bid, **final** String bname,

**final** String btitle, **final** String bcontent) {

String query = "UPDATE MVC\_BOARD "

+ " SET BNAME = ?, BTITLE=?, BCONTENT=? "

+ " WHERE BID=?";

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

@Override

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

pstmt.setString(1, bname);

pstmt.setString(2, btitle);

pstmt.setString(3, bcontent);

pstmt.setInt(4, bid);

}

});

}

// 해당 글번호(bid)로 삭제하는 method

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

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

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

@Override

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

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setInt(1, bid);

**return** pstmt;

}

});

}

// 원글 글번호(bid)로 원글의 dto를 리턴하는 method

**public** BDto reply\_view(**int** bid) {

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

**return** template.queryForObject(query, **new** BeanPropertyRowMapper<BDto>(BDto.**class**));

}

// 답변글 DB에 추가하는 method

**public** **int** reply(**final** String bname, **final** String btitle, **final** String bcontent,

**final** **int** bgroup, **final** **int** bstep, **final** **int** bindent) {

// bname, btitle, bcontent 답글에 추가할 내용

// bgroup, bstep, bindent 원글의 내용

replyAstep(bgroup, bstep);

String query = "INSERT INTO MVC\_BOARD "

+ "(bId, bName, bTitle, bContent, bGroup, bStep, bIndent)"

+ " VALUES "

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

+ " ?, ?)";

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

@Override

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

pstmt.setString(1, bname);

pstmt.setString(2, btitle);

pstmt.setString(3, bcontent);

pstmt.setInt(4, bgroup);

pstmt.setInt(5, bstep);

pstmt.setInt(6, bindent);

}

});

}

// 엑셀의 ⓐ step method

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

String query = "UPDATE MVC\_BOARD SET BSTEP = BSTEP+1 "

+ " WHERE BGROUP =? AND BSTEP>?";

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

@Override

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

pstmt.setInt(1, bgroup);

pstmt.setInt(2, bstep);

}

});

}

}