# Spring 的JdbcTemplate常用执行sql语句的方法

## execute（）：执行一个独立的sql语句，包括ddl语句；

## update(…)：使用于增、删、改操作；

## queryForInt ：查询出一个整数值

## batchUpdate适合于批量增、删、改操作；

# 实例

## Execute

### Execute 🡪create

/\*\*

\* 创建表

\*/

**public** **void** create(String tableName) { // tb\_test1

jdbcTemplate.execute("create table " + tableName

+ " (id integer,user\_name varchar2(40),password varchar2(40))");

}

## Update

### Update 🡪 insert

/\*\*

\* jdbcTemplate.update适合于insert 、update和delete操作；

\*/

// 第一个参数为执行sql 第二个参数为参数数据

**public** **void** save3(User user) {

Assert.isNull(user, "user is not null");

jdbcTemplate.update("insert into tb\_test1(name,password) values(?,?)",

**new** Object[] { user.getUsername(), user.getPassword() });

}

//第一个参数为执行sql 第二个参数为参数数据 第三个参数为参数类型

@Override

**public** **void** save(User user) {

Assert.isNull(user, "user is not null");

jdbcTemplate.update("insert into tb\_test1(name,password) values(?,?)",

**new** Object[] { user.getUsername(), user.getPassword() },

**new** **int**[] { java.sql.Types.*VARCHAR*, java.sql.Types.*VARCHAR* });

}

// 避免sql注入

**public** **void** save2(**final** User user) {

Assert.isNull(user, "user is not null");

jdbcTemplate.update("insert into tb\_test1(name,password) values(?,?)",

**new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement ps)

**throws** SQLException {

ps.setString(1, user.getUsername());

ps.setString(2, user.getPassword());

}

});

}

**public** **void** save4(User user) {

Assert.isNull(user, "user is not null");

jdbcTemplate.update("insert into tb\_test1(name,password) values(?,?)",

**new** Object[] { user.getUsername(), user.getPassword() });

}

// 返回插入的主键

**public** List save5(**final** User user) {

KeyHolder keyHolder = **new** GeneratedKeyHolder();

jdbcTemplate.update(**new** PreparedStatementCreator() {

@Override

**public** PreparedStatement createPreparedStatement(

Connection connection) **throws** SQLException {

PreparedStatement ps = connection.prepareStatement(

"insert into tb\_test1(name,password) values(?,?)",

**new** String[] { "id" });

ps.setString(1, user.getUsername());

ps.setString(2, user.getPassword());

**return** ps;

}

}, keyHolder);

**return** keyHolder.getKeyList();

}

### Update 🡪 update

@Override

**public** **void** update(**final** User user) {

jdbcTemplate.update(

"update tb\_test1 set name=？,password=？ where id = ?",

**new** PreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement ps)

**throws** SQLException {

ps.setString(1, user.getUsername());

ps.setString(2, user.getPassword());

ps.setInt(3, user.getId());

}

});

}

### Update 🡪 delete

// 删除

@Override

**public** **void** delete(User user) {

Assert.isNull(user, "user is not null");

jdbcTemplate.update("delete from tb\_test1 where id = ?",

**new** Object[] { user.getId() },

**new** **int**[] { java.sql.Types.*INTEGER* });

}

## queryForInt

/\*\*

\* queryForInt

\*

\*/

@Deprecated

// 因为没有查询条件，所以用处不大

**public** **int** ~~queryForInt1~~() {

**return** jdbcTemplate.queryForInt("select count(0) from tb\_test1");

}

**public** **int** queryForInt2(User user) {

**return** jdbcTemplate.queryForInt(

"select count(0) from tb\_test1 where username = ?",

**new** Object[] { user.getUsername() });

}

// 最全的参数3个

**public** **int** queryForInt3(User user) {

**return** jdbcTemplate.queryForInt(

"select count(0) from tb\_test1 where username = ?",

**new** Object[] { user.getUsername() },

**new** **int**[] { java.sql.Types.*VARCHAR* });

}

## queryForObject

/\*\*

\* queryForObject

\*/

// 可以返回是一个基本类型的值

@Deprecated

// 因为没有查询条件，所以用处不大

**public** String ~~queryForObject1~~(User user) {

**return** (String) jdbcTemplate.queryForObject(

"select username from tb\_test1 where id = 100", String.**class**);

}

// 可以返回值是一个对象

@Deprecated

// 因为没有查询条件，所以用处不大

**public** User ~~queryForObject2~~(User user) {

**return** (User) jdbcTemplate.queryForObject(

"select \* from tb\_test1 where id = 100", User.**class**); // class是结果数据的java类型

}

@Deprecated

// 因为没有查询条件，所以用处不大

**public** User ~~queryForObject3~~(User user) {

**return** (User) jdbcTemplate.queryForObject(

"select \* from tb\_test1 where id = 100", **new** RowMapper() {

@Override

**public** Object mapRow(ResultSet rs, **int** rowNum)

**throws** SQLException {

User user = **new** User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

**return** user;

}

});

}

**public** User queryForObject4(User user) {

**return** (User) jdbcTemplate.queryForObject(

"select \* from tb\_test1 where id = ?",

**new** Object[] { user.getId() }, User.**class**); // class是结果数据的java类型

// 实际上这里是做反射，将查询的结果和User进行对应复制

}

**public** User queryForObject5(User user) {

**return** (User) jdbcTemplate.queryForObject(

"select \* from tb\_test1 where id = ?",

**new** Object[] { user.getId() }, **new** RowMapper() {

@Override

**public** Object mapRow(ResultSet rs, **int** rowNum)

**throws** SQLException {

User user = **new** User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

**return** user;

}

}); // class是结果数据的java类型

}

@Override

**public** User queryForObject(User user) {

// 方法有返回值

**return** (User) jdbcTemplate.queryForObject(

"select \* from tb\_test1 where id = ?",

**new** Object[] { user.getId() },

**new** **int**[] { java.sql.Types.*INTEGER* }, **new** RowMapper() {

@Override

**public** Object mapRow(ResultSet rs, **int** rowNum)

**throws** SQLException {

User user = **new** User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

**return** user;

}

});

}

## queryForList

/\*\*

\* queryForList

\*/

@SuppressWarnings("unchecked")

**public** List<User> queryForList1(User user) {

**return** (List<User>) jdbcTemplate.queryForList(

"select \* from tb\_test1 where username = ?",

**new** Object[] { user.getUsername() }, User.**class**);

}

@SuppressWarnings("unchecked")

**public** List<String> queryForList2(User user) {

**return** (List<String>) jdbcTemplate.queryForList(

"select username from tb\_test1 where sex = ?",

**new** Object[] { user.getSex() }, String.**class**);

}

@SuppressWarnings("unchecked")

// 最全的参数查询

**public** List<User> queryForList3(User user) {

**return** (List<User>) jdbcTemplate.queryForList(

"select \* from tb\_test1 where username = ?",

**new** Object[] { user.getUsername() },

**new** **int**[] { java.sql.Types.*VARCHAR* }, User.**class**);

}

## Query

### Query 🡪 return object

/\*\*

\* query

\*/

// 通过RowCallbackHandler对Select语句得到的每行记录进行解析，并为其创建一个User数据对象。实现了手动的OR映射。

**public** User queryUserById4(String id) {

**final** User user = **new** User();

// 该方法返回值为void

**this**.jdbcTemplate.query("select \* from tb\_test1 where id = ?",

**new** Object[] { id }, **new** RowCallbackHandler() {

@Override

**public** **void** processRow(ResultSet rs) **throws** SQLException {

User user = **new** User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

}

});

**return** user;

}

### Query 🡪 return List<object>

@SuppressWarnings("unchecked")

@Override

**public** List<User> list(User user) {

**return** jdbcTemplate.query(

"select \* from tb\_test1 where username like '%?%'",

**new** Object[] { user.getUsername() },

**new** **int**[] { java.sql.Types.*VARCHAR* }, **new** RowMapper() {

@Override

**public** Object mapRow(ResultSet rs, **int** rowNum)

**throws** SQLException {

User user = **new** User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

**return** user;

}

});

}

## batchUpdate

/\*\*

\* batchUpdate

\*/

// 批量操作 适合于增、删、改操作

**public** **int**[] batchUpdate(**final** List users) {

**int**[] updateCounts = jdbcTemplate.batchUpdate(

"update tb\_test1 set username = ?, password = ? where id = ?",

**new** BatchPreparedStatementSetter() {

@Override

**public** **void** setValues(PreparedStatement ps, **int** i)

**throws** SQLException {

ps.setString(1, ((User) users.get(i)).getUsername());

ps.setString(2, ((User) users.get(i)).getPassword());

ps.setLong(3, ((User) users.get(i)).getId());

}

@Override

**public** **int** getBatchSize() {

**return** users.size();

}

});

**return** updateCounts;

}

/\*\*

\* update调用存储过程

\* **@param** id

\*/

// 调用存储过程

**public** **void** callProcedure(**int** id) {

**this**.jdbcTemplate.update("call SUPPORT.REFRESH\_USERS\_SUMMARY(?)",

**new** Object[] { Long.*valueOf*(id) });

}