Atitit db insert sql 数据插入最简单功能

sqlSessionFactory=getSqlSessFctry(dataSource, UserMapper.class); tk\_ini(sqlSessionFactory);

SimpleJdbcInsert SimpleJdbcInsert1= new SimpleJdbcInsert(dataSource).withTableName("user");

Map<String, Object> parameters\_map = new HashMap<String, Object>(3);

parameters\_map.put("id", 123);

parameters\_map.put("first\_name", "aaa");

// SimpleJdbcInsert1.setColumnNames(columnNames);

SimpleJdbcInsert1.execute(parameters\_map);

// SimpleJdbcInsert1.exe

System.out.println("f");

### **11.5.3. 指定SimpleJdbcInsert所使用的字段**

通过指定所使用的字段名称，可以使SimpleJdbcInsert仅使用这些字段作为insert语句所使用的字段。这可以通过usingColumns方法进行配置。

public class JdbcActorDao implements ActorDao {

private SimpleJdbcTemplate simpleJdbcTemplate;

private SimpleJdbcInsert insertActor;

public void setDataSource(DataSource dataSource) {

this.simpleJdbcTemplate = new SimpleJdbcTemplate(dataSource);

this.insertActor =

new SimpleJdbcInsert(dataSource)

.withTableName("t\_actor")

.usingColumns("first\_name", "last\_name")

### **11.5.2. 使用SimpleJdbcInsert来获取自动生成的主键**

接下来，我们对于同样的插入语句，我们并不传入id，而是通过数据库自动获取主键的方式来创建新的Actor对象并插入数据库。 当我们创建SimpleJdbcInsert实例时, 我们不仅需要指定表名，同时我们通过usingGeneratedKeyColumns方法指定需要数据库自动生成主键的列名

### **11.5.4. 使用SqlParameterSource提供参数值**

使用Map来指定参数值有时候工作得非常好，但是这并不是最简单的使用方式。Spring提供了一些其他的SqlParameterSource实现类来指定参数值。 我们首先可以看看BeanPropertySqlParameterSource类，这是一个非常简便的指定参数的实现类，只要你有一个符合JavaBean规范的类就行了。它将使用其中的getter方法来获取参数值。 下面是一个例子：

public class JdbcActorDao implements ActorDao {

private SimpleJdbcTemplate simpleJdbcTemplate;

private SimpleJdbcInsert insertActor;

public void setDataSource(DataSource dataSource) {

this.simpleJdbcTemplate = new SimpleJdbcTemplate(dataSource);

this.insertActor =

new SimpleJdbcInsert(dataSource)

.withTableName("t\_actor")

.usingGeneratedKeyColumns("id");

}

public void add(Actor actor) {

SqlParameterSource parameters = new BeanPropertySqlParameterSource(actor);

Number newId = insertActor.executeAndReturnKey(parameters);

actor.setId(newId.longValue());

}

// ... additional methods

}

另外一个实现类：MapSqlParameterSource也使用Map来指定参数，但是他另外提供了一个非常简便的addValue方法，可以被连续调用，来增加参数。

public class JdbcActorDao implements ActorDao {

private SimpleJdbcTemplate simpleJdbcTemplate;

private SimpleJdbcInsert insertActor;

public void setDataSource(DataSource dataSource) {

this.simpleJdbcTemplate = new SimpleJdbcTemplate(dataSource);

this.insertActor =

new SimpleJdbcInsert(dataSource)

.withTableName("t\_actor")

.usingGeneratedKeyColumns("id");

}

public void add(Actor actor) {

SqlParameterSource parameters = new MapSqlParameterSource()

.addValue("first\_name", actor.getFirstName())

.addValue("last\_name", actor.getLastName());

Number newId = insertActor.executeAndReturnKey(parameters);

actor.setId(newId.longValue());

}

// ... additional methods

## 插入实体类BeanPropertySqlParameterSource

*//方法二* *// 不需要调用executeAndReturnKey方法,而且传入的值可以是Map，也可以是SqlParameterSource* Number aliceId = simpleJdbcInsert.executeAndReturnKey(new BeanPropertySqlParameterSource(new User("alice", 18)));

### **11.5.5. 使用SimpleJdbcCall调用存储过程**

接下来我们把我们的关注点放在使用 SimpleJdbcCall 来进行存储过程的调用上。 设计这个类的目的在于使得调用存储过程尽可能简单。它同样利用的数据库元数据的特性来查找传入的参数和返回值。 这意味着你无需明确声明那些参数。当然，如果你喜欢，可以依然声明这些参数，尤其对于某些参数，你无法直接将他们映射到Java类上，例如ARRAY类型和STRUCT类型的参数。 在我们的第一个示例中，我们可以看到一个简单的存储过程调用，它仅仅返回VARCHAR和DATE类型。 这里，我特地为Actor类增加了一个birthDate的属性，从而可以使得返回值拥有不同的数据类型。 这个存储过程读取actor的主键，并返回first\_name，last\_name，和birth\_date字段作为返回值。 下面是这个存储过程的源码，它可以工作在MySQL数据库上：

CREAT

Spring JDBC 常用批量操作及插入操作\_沉默的鲨鱼的专栏-CSDN博客