(http://baeldung.com)

Auto-generated Keys in Spring JDBC

Last modified: April 18, 2018

by baeldung (http://www.baeldung.com/author/baeldung/)

Persistence (http://www.baeldung.com/category/persistence/)
Spring (http://www.baeldung.com/category/spring/) +

1. Introduction

In this quick tutorial, we'll explore the possibility of getting the auto-generated key after inserting entities when working with *Spring JDBC*.

2. Maven Dependencies

At first, we need to have *spring-boot-starter-jdbc* and *H2* dependencies defined in our *pom.xml*:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```


We can check out the latest version of those two dependencies on Maven Central: *spring-boot-starter-jdbc* (https://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22org.springframework.boot%22%20AND%20a%3A%22spring-boot-starter-jdbc%22) and *h2* *(https://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22com.h2database%22%20AND%20a%3A%22h2%22)*.

3. Getting the Auto-Generated Key

3

Le h has 2 columns: *id* (auto-generated key) and *message*:

```
essage (
INCREMENT,
L,
```

3.2. Using the *JdbcTemplate*

Now, let's implement a method which will use *JDBCTemplate* to insert the new record and return the auto-generated *id*.

Therefore, **we'll use the *JDBCTemplate update()*** method which supports the retrieval of primary keys generated by the database. This method takes an instance of the *PrepareStatementCreator* interface as the first argument and the other argument is the *KeyHolder*.

Since the *PrepareStatementCreator* interface is a *FunctionalInterface* where its method accepts an instance of *java.sql.Connection* and return a *java.sql.PreparedStatement* object, for simplicity, we can use a lambda expression:

```
String INSERT_MESSAGE_SQL
  = "insert into sys_message (message) values(?) ";

public long insertMessage(String message) {
    KeyHolder keyHolder = new GeneratedKeyHolder();

    jdbcTemplate.update(connection -> {
        PreparedStatement ps = connection
          .prepareStatement(INSERT_MESSAGE_SQL);
          ps.setString(1, message);
          return ps;
        }, keyHolder);

        return (long) keyHolder.getKey();
    }
}
```

It's worth noting that the *keyHolder* object will contain the auto-generated key return from the *JDBCTemplate update()* method.

We can retrieve that key by calling *keyHolder.getKey()*.

Besides, we can verify the method:

```
@Test
public void
  insertJDBC_whenLoadMessageByKey_thenGetTheSameMessage() {
    long key = messageRepositoryJDBCTemplate.insert(MESSAGE_CONTENT);
    String loadedMessage = messageRepositoryJDBCTemplate
      .getMessageById(key);

    assertEquals(MESSAGE_CONTENT, loadedMessage);
}
```


3.3. Using *SimpleJdbcInsert*

In addition to the *JDBCTemplate*, we also can use *SimpleJdbcInsert* to achieve the same result.

Hence, we need to initialize an instance of the *SimpleJdbcInsert*:

```
@Repository
public class MessageRepositorySimpleJDBCInsert {

                                                    Insert;


                                                    pleJDBCInsert(DataSource dataSource) {
                                                    SimpleJdbcInsert(dataSource)
                                                    essage").usingGeneratedKeyColumns("id");
```


C **teAndReturnKey method of the *SimpleJdbcInsert* to insert a new record to *sys_message* table and get back the auto-generated key**:

```
public long insert(String message) {
    Map<String, Object> parameters = new HashMap<>(1);
    parameters.put("message", message);
    Number newId = simpleJdbcInsert.executeAndReturnKey(parameters);
    return (long) newId;
}
```

Furthermore, we can verify that method quite simply:

```
@Test
public void
  insertSimpleInsert_whenLoadMessageKey_thenGetTheSameMessage() {
    long key = messageRepositorySimpleJDBCInsert.insert(MESSAGE_CONTENT);
    String loadedMessage = messageRepositoryJDBCTemplate.getMessageById(key);

    assertEquals(MESSAGE_CONTENT, loadedMessage);
}
```

4. Conclusion

We've explored the possibility of using *JDBCTemplate* and *SimpleJdbcInsert* for inserting a new record and getting the auto-generated key back.

As always, we can find the implementation of this article over on Github (https://github.com/eugenp/tutorials/tree/master/persistence-modules/spring-jpa).

VADI VEL

It looks good. Please let me know Any other way to get PK with respect to sequence

2 days ago

anu (http://www.baeldung.com/author/loredana-crusoveanu/)

om/author/loredana-

ring with JDBC. This is getting the id of the record inserted by the executed, whether it's coming from a sequence or not. Did you have another scenario in mind?

2 days ago

CATEGORIES

SPRING (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIES

JAVA "BACK TO BASICS" TUTORIAL (HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (HTTP://WWW.BAELDUNG.COM/JACKSON)
HTTPCLIENT 4 TUTORIAL (HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/)
SPRING PERSISTENCE TUTORIAL (HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/)
SECURITY WITH SPRING (HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING)

ABOUT

ABOUT BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
THE COURSES (HTTP://COURSES.BAELDUNG.COM)
CONSULTING WORK (HTTP://WWW.BAELDUNG.COM/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
THE FULL ARCHIVE (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
WRITE FOR BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)