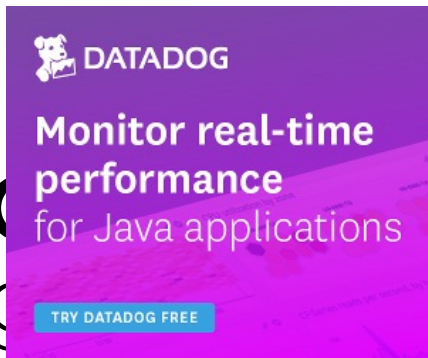


<http://baeldung.com>

Auto-Generated Keys in Spring

[\(datadog\)](#)

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/>) +

I just announced the new *Spring 5* modules in REST With Spring:

>> **CHECK OUT THE COURSE** (</rest-with-spring-course#new-modules>)

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*:

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-jdbc</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>com.h2database</groupId>
7   <artifactId>h2</artifactId>
8   <scope>runtime</scope>
9 </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>).

Which of these is closest to your current job/role?

Developer

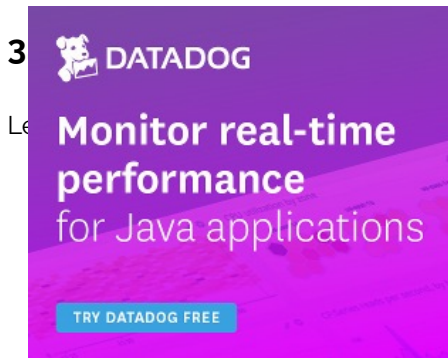
Senior Developer

Lead Developer

Architect

Manager

3. Getting the Auto-Generated Key



Let's create a table `sys_message` which has 2 columns: `id` (auto-generated key) and `message`:

```
CREATE TABLE sys_message (
  id INT(11) UNSIGNED ZEROFILL AUTO_INCREMENT,
  message VARCHAR(255)
);
```

(`/datadog`)

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 *PreparedStatementCreator* interface as the first argument and the other argument is the *KeyHolder*.

Since the *PreparedStatementCreator* 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:

```
1 String INSERT_MESSAGE_SQL
2   = "insert into sys_message (message) values(?) ";
3
4 public long insertMessage(String message) {
5     KeyHolder keyHolder = new GeneratedKeyHolder();
6
7     jdbcTemplate.update(connection -> {
8         PreparedStatement ps = connection
9             .prepareStatement(INSERT_MESSAGE_SQL);
10        ps.setString(1, message);
11        return ps;
12    }, keyHolder);
13
14    return (long) keyHolder.getKey();
15 }
16 }
```

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:

```
1 @Test
2 public void
3     insertJDBC_whenLoadMessageByKey_thenGetTheSameMessage() {
4     long key = messageRepositoryJdbcTemplate.insert(MESSAGE_CONTENT);
5     String loadedMessage = messageRepositoryJdbcTemplate
6         .getMessageById(key);
7
8     assertEquals(MESSAGE_CONTENT, loadedMessage);
9 }
```

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*:

Which of these is closest to your current job/role?

Developer

Senior Developer

Lead Developer

Architect

Manager

```

1 | @Repository
2 | public class MessageRepositorySimpleJDBCInsert {
3 |
4 |     private SimpleJdbcInsert simpleJdbcInsert;
5 |
6 |     public MessageRepositorySimpleJDBCInsert(DataSource dataSource) {
7 |         this.simpleJdbcInsert = new SimpleJdbcInsert(dataSource)
8 |             .withTableName("message")
9 |             .usingGeneratedKeyColumns("id");
10 |    }
11 |
12 |    public long insert(String message) {
13 |        Map<String, Object> parameters = new HashMap<>(1);
14 |        parameters.put("message", message);
15 |        Number newId = simpleJdbcInsert.executeAndReturnKey(parameters);
16 |        return newId.longValue();
17 |    }
18 | }

```



Call the `executeAndReturnKey` method of the `SimpleJdbcInsert` to insert a new record to the `message` table and get back the auto-generated key:

```

1 | public long insert(String message) {
2 |     Map<String, Object> parameters = new HashMap<>(1);
3 |     parameters.put("message", message);
4 |     Number newId = simpleJdbcInsert.executeAndReturnKey(parameters);
5 |     return newId.longValue();
6 | }

```

Furthermore, we can verify that method quite simply:

```

1 | @Test
2 | public void insertSimpleInsert_whenLoadMessageKey_thenGetTheSameMessage() {
3 |     long key = messageRepositorySimpleJDBCInsert.insert(MESSAGE_CONTENT);
4 |     String loadedMessage = messageRepositoryJDBCTemplate.getMessageById(key);
5 |
6 |     assertEquals(MESSAGE_CONTENT, loadedMessage);
7 | }

```

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>).

I just announced the new Spring 5 modules in REST With Spring:

>> CHECK OUT THE LESSONS ([/rest-with-spring-course#new-modules](#))

Leave a Reply



Join the discussion

Subscribe ▼

Which of these is closest to your current job/role?

2 Comments on "Obtaining Auto-generated Keys in Spring JDBC"

Senior Developer

Lead Developer

Architect

▲ newest ▲ **oldest** ▲ most voted

Manager



VADI VEL



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

 **DATADOG**

Monitor real-time performance for Java applications

[TRY DATADOG FREE](#)

🕒 2 days ago ^

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



com/author/loredana-crusoveanu/ I'm using Spring with JDBC. This is getting the id of the record inserted by the
is executed, whether it's coming from a sequence or not. Did you have another scenario in mind?

🕒 2 days ago

(/datadog)

CATEGORIES

- SPRING ([HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/](http://www.baeldung.com/category/spring/))
- REST ([HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/](http://www.baeldung.com/category/rest/))
- JAVA ([HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/](http://www.baeldung.com/category/java/))
- SECURITY ([HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/](http://www.baeldung.com/category/security-2/))
- PERSISTENCE ([HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/](http://www.baeldung.com/category/persistence/))
- JACKSON ([HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/](http://www.baeldung.com/category/jackson/))
- HTTPCLIENT ([HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/](http://www.baeldung.com/category/http/))
- KOTLIN ([HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/](http://www.baeldung.com/category/kotlin/))

SERIES

- JAVA "BACK TO BASICS" TUTORIAL ([HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL](http://www.baeldung.com/java-tutorial))
- JACKSON JSON TUTORIAL ([HTTP://WWW.BAELDUNG.COM/JACKSON](http://www.baeldung.com/jackson))
- HTTPCLIENT 4 TUTORIAL ([HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE](http://www.baeldung.com/httpclient-guide))
- REST WITH SPRING TUTORIAL ([HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/](http://www.baeldung.com/rest-with-spring-series/))
- SPRING PERSISTENCE TUTORIAL ([HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/](http://www.baeldung.com/persistence-with-spring-series/))
- SECURITY WITH SPRING ([HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING](http://www.baeldung.com/security-spring))

ABOUT

- ABOUT BAELDUNG ([HTTP://WWW.BAELDUNG.COM/ABOUT/](http://www.baeldung.com/about/))
- THE COURSES ([HTTP://COURSES.BAELDUNG.COM](http://courses.baeldung.com))
- CONSULTING WORK ([HTTP://WWW.BAELDUNG.COM/CONSULTING](http://www.baeldung.com/consulting))
- META BAELDUNG ([HTTP://META.BAELDUNG.COM/](http://meta.baeldung.com/))
- THE FULL ARCHIVE ([HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE](http://www.baeldung.com/full_archive))
- WRITE FOR BAELDUNG ([HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES](http://www.baeldung.com/contribution-guidelines))

Which of these is closest to your current job/role?

Developer

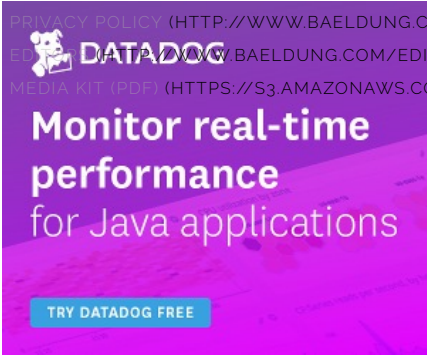
Senior Developer

Lead Developer

Architect

Manager

CONTACT ([HTTP://WWW.BAELDUNG.COM/CONTACT](http://www.baeldung.com/contact))
COMPANY INFO ([HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO](http://www.baeldung.com/baeldung-company-info))
TERMS OF SERVICE ([HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE](http://www.baeldung.com/terms-of-service))
PRIVACY POLICY ([HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY](http://www.baeldung.com/privacy-policy))
EDITORIAL POLICY ([HTTP://WWW.BAELDUNG.COM/EDITORS](http://www.baeldung.com/editors))
MEDIA KIT (PDF) ([HTTPS://S3.AMAZONAWS.COM/BAELDUNG+-+MEDIA+KIT.PDF](https://s3.amazonaws.com/baeldung.com/baeldung+-+media+kit.pdf))



(/datadog)

Which of these is closest to your current job/role?

Developer

Senior Developer

Lead Developer

Architect

Manager