

 > [Session](#) > **Storing Spring Sessions in Database using JDBC**

Storing Spring Sessions in Database using JDBC



Raja Anbazhagan January 3, 2021

SESSION

In this post, We will take a look at setting up Spring Boot Session module using database/JDBC as the backend.

Introduction

When running multiple instances of the same application, sharing the session data can be a good idea. In this post we will take a look at using a database as a session store for spring boot application.

Session store is usually a key-value map that contains a session id and the UserDetails of the currently logged-in user. With the Spring Sessions module, we get to supply different type of session stores.

Spring session dependencies

First, you need to add the **spring session jdbc** to your spring boot application. You also need to add jdbc or JPA dependency. In this case, I'm using JPA.



AD

```

    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-jdbc</artifactId>
  </dependency>

```

By default, spring boot can auto-detect the session implementation based on the dependencies. If you have dependencies for two session stores, then you will have to specify one.

```
spring.session.store-type=jdbc
```

Setting up the tables for storing session

When using embedded servers, spring boot automatically creates two tables called **SPRING_SESSION** and **SPRING_SESSION_ATTRIBUTES**. This behaviour is controlled by the `spring.session.jdbc.initialize-schema` Configuration. This configuration can be set to one of `embedded`, `always` or `never` and the default is `embedded`.

If you are using databases like MySQL, Postgres or Oracle, you may need to set the `initialize-schema` setting to `never` and create these tables manually. For example, here is the create script for MySQL.

```

CREATE TABLE SPRING_SESSION
(
    PRIMARY_ID          CHAR(36) NOT NULL,
    SESSION_ID          CHAR(36) NOT NULL,
    CREATION_TIME       BIGINT   NOT NULL,

```

```

        PRINCIPAL_NAME          VARCHAR(100),
        CONSTRAINT SPRING_SESSION_PK PRIMARY KEY (PRIMARY_ID)
    ) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;

CREATE UNIQUE INDEX SPRING_SESSION_IX1 ON SPRING_SESSION (SESSION_ID);
CREATE INDEX SPRING_SESSION_IX2 ON SPRING_SESSION (EXPIRY_TIME);
CREATE INDEX SPRING_SESSION_IX3 ON SPRING_SESSION (PRINCIPAL_NAME);

CREATE TABLE SPRING_SESSION_ATTRIBUTES
(
    SESSION_PRIMARY_ID CHAR(36) NOT NULL,
    ATTRIBUTE_NAME      VARCHAR(200) NOT NULL,
    ATTRIBUTE_BYTES     BLOB NOT NULL,
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_PK PRIMARY KEY (SESSION_PRIMARY_ID),
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_FK FOREIGN KEY (SESSION_PRIMARY_ID) REFERENCES SPRING_SESSION (SESSION_ID)
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;

```

Here you can find the [table creation scripts for different databases](#).

Verifying our setup

For instance, I have used in-memory H2 database. So, spring boot will create the database tables for me.

```

spring.datasource.url=jdbc:h2:mem:test
spring.datasource.username=root
spring.datasource.password=root
# To enable h2 console to check embedded database
spring.h2.console.enabled=true

```



SELECT * FROM SPRING_SESSION;

PRIMARY_ID	SESSION_ID	CREATION_TIME	LAST_ACCESS_TIME	MAX_INACTIVE_INTERVAL	EXPIRY_TIME	PRINCIPAL_NAME
d85be90a-ea45-44d4-9dab-d1a8d1a9400f	5331c66f-9d8b-4b7f-b850-28c4e104f1f0	1609669322600	1609669558561	1800	1609671358561	admin
a47fc953-86be-4ea1-977d-4cc6bf539a24	bd1cd198-1006-4cd9-b388-64f5f0bd78ac	1609669573380	1609669583366	1800	1609671383366	user1
ff7f3abc-1b5a-4f48-9c38-bd29b67daa4f	e4e3919a-0897-4119-8660-11403491c3c2	1609669594659	1609669598541	1800	1609671398541	user2

(3 rows, 1 ms)

SELECT * FROM SPRING_SESSION_ATTRIBUTES;

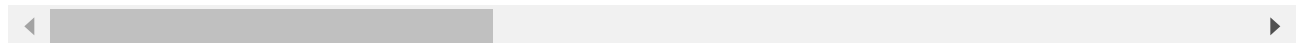
SESSION_PRIMARY_ID	ATTRIBUTE_NAME	ATTRIBUTE_BYTES
d85be90a-ea45-44d4-9dab-d1a8d1a9400f	SPRING_SECURITY_CONTEXT	aced00057372003d6f72672e737072696e676672616d66
a47fc953-86be-4ea1-977d-4cc6bf539a24	SPRING_SECURITY_CONTEXT	aced00057372003d6f72672e737072696e676672616d66
ff7f3abc-1b5a-4f48-9c38-bd29b67daa4f	SPRING_SECURITY_CONTEXT	aced00057372003d6f72672e737072696e676672616d66

Spring Session JDBC in action

Spring session store is pretty much a key value pair. Here, the key is the session id and the value is the session object. That is, We can store this object anywhere by simply serializing them. In our case, the spring module will use JDBC to store the session data.

The columns like **LAST_ACCESS_TIME** and **EXPIRY_TIME** make sure that the session expires properly. A housekeeping process will keep looking for expired records and delete them if necessary. You can see these operations when running the application with `debug=true`.

```
2021-01-03 16:48:00.002 DEBUG 18824 --- [pool-1-thread-1] o.s.:
```



By default, the housekeeping runs every minute. If you want to change it let's say every 3 minutes, you need to configure the application as shown below.

```
spring.session.jdbc.cleanup-cron=0 */3 * * * *
```



So choose an optimal time that fits for your server.

Changing spring session table names

We can override the default table names using **spring.session.jdbc.table-name** property. For example, you can instruct spring sessions to use the tables *USER_SESSION* and *USER_SESSION_ATTRIBUTES* using the following configuration.

```
spring.session.jdbc.table-name=USER_SESSION
```

However, The auto configuration can't create custom table names. So, You need to create the **USER_SESSION** and **USER_SESSION_ATTRIBUTES** tables manually. Just replace the table names in the script from setting up the tables section and your work is complete.

Summary

So far, We learned how to add Spring Session and how to configure it to use JDBC as session store. If you liked this article, you would also like to read the following write-ups.

- [Storing Spring sessions in a Separate Database](#)
- [Spring Session using Redis as Session Store](#)
- [Using JdbcTemplate with Spring Boot](#)
- [Customizing spring Session Cookies](#)
- [Session Tracking modes in Spring security](#)

The full example is available in this [GitHub Repository](#).



Similar Posts

Spring sessions in a Separate Database

January 3, 2021 Session, Spring Boot

Redis as Session Store in Spring Boot

January 1, 2021 Session, Spring Boot

Leave a Reply

*Your email address will not be published. Required fields are marked **

Comment *



Name *

Email *

☐ Save my name, email, and website in this browser for the next time I comment.

☒ Notify me via e-mail if anyone answers my comment.

Post Comment

One Comment

Rex Roy says:

March 10, 2021 at 10:30 am

Good read ..

Reply

Search...



AD

AD



Download now

Compatible PC and Google account

AD



BETA

Download now

Compatible PC and Google account



AD

Recent Posts

Why Field injection is not Recommended?

Spring Boot and Postgres Using Docker Compose

How to Run a Spring Boot Application on Docker

Logging In Spring Boot

Changing Context Path in a Spring Boot Application

Ways to add Servlet Filters in Spring Boot

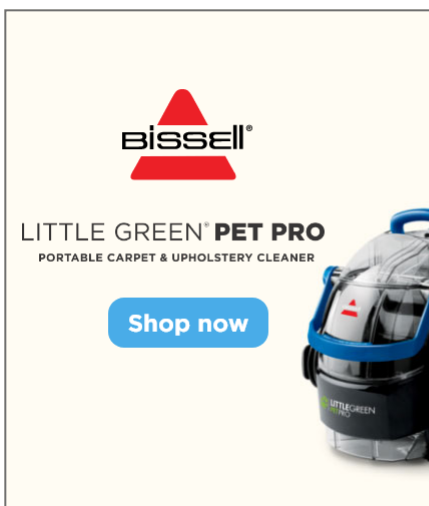
Ways to run Code on Application Startup in Spring Boot

What is the purpose of mvnw and mvnw.cmd files?

Apache Commons Logging – Explained

Accessing application.properties in Spring Boot

AD



AD

AD



\$0 Means More at Schwab

**\$0 Online Listed Stock,
ETF, and Base
Options Commissions***


+

Satisfaction Guara

CHOOSE SCHWAB


*Disclosure *Own your.*

AD



LITTLE GREEN® PET PRO
PORTABLE CARPET & UPHOLSTERY CLEANER

Shop now





AD

