Title: Spring retry connection until datasource is available

Post Body:

I have a docker-compose setup to start my SpringBoot application and a MySQL database. If the database starts first, then my application can connect successfully. But if my application starts first, no database exists yet, so the application throws the following exception and exits:

```
app_1       | 2018-05-27 14:15:03.415  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
```

I could edit my docker-compose file to make sure the database is always up before the application starts up, but I want the application to be able to handle this case on its own, and not immediately exit when it cannot reach the database address.

There are ways to configure the datasource in the application.properties file to make the application reconnect to the database, as answered here and here. But that doesn't work for a startup connection to the datasource.

How can I make my SpringBoot application retry the connection at startup to the database at a given interval until it successfully connects to the database?

Accepted Answer:

Set HikariCP's `initializationFailTimeout` property to 0 (zero), or a negative number. As documented here:

> ■`initializationFailTimeout`
>
> This property controls whether the pool will 'fail fast' if the pool cannot be seeded with an initial connection successfully. Any positive number is taken to be the number of milliseconds to attempt to acquire an initial connection; the application thread will be blocked during this period. If a connection cannot be acquired before this timeout occurs, an exception will be thrown. This timeout is applied *after* the `connectionTimeout` period. If the value is zero (0), HikariCP will attempt to obtain and validate a connection. If a connection is obtained, but fails validation, an exception will be thrown and the pool not started. However, if a connection cannot be obtained, the pool will start, but later efforts to obtain a connection may fail. A value less than zero will bypass any initial connection attempt, and the pool will start immediately while trying to obtain connections in the background. Consequently, later efforts to obtain a connection may fail. *Default*: 1

Highest Rated Answer:

There is an alternative way to do this, *which doesn't rely on a specific Connection Pool library or a specific database*. Note that you will need to use `spring-retry` to achieve the desired behaviour with this approach

First you need to add spring-retry to your dependencies :

```
<dependency>     <groupId>org.springframework.retry</groupId>     <artifactId>spring-retry</artifactId>     <version>${spring-
```

Then you can create a decorator over `DataSource` that will extends `AbstractDataSource` like bellow :

```
@Slf4j @RequiredArgsConstructor public class RetryableDataSource extends AbstractDataSource {      private final DataSource da
```

Then you will need to inject this custom DataSource decorator into Spring context by creating a custom `BeanPostProcessor` :

```
@Slf4j @Order(value = Ordered.HIGHEST_PRECEDENCE) @Component public class RetryableDatabasePostProcessor implements BeanPostPr
```

Last but not least you will need to enable Spring retry by adding `@EnableRetry` annotation to spring main class, example :

```
@EnableRetry @SpringBootApplication public class RetryableDbConnectionApplication {      public static void main(String[] args
```