After Spring Boot 2.0 migration: jdbcUrl is required with driverClassName

Asked 4 years, 5 months ago Modified 7 months ago Viewed 183k times

98

29

SOLVED: My solution: I delete dataSource() method from DatabaseConfig.java. Then, the application was started successfully :)

I've just updated my **Spring Boot** project from **1.5.x** to **2.0.0**. Before the update, this application works properly but after the update, I'm getting some errors are below. What is the problem, can you help me?

I use PostgreSQL, Hibernate, JPA in the project.

I've tried also this but it doesn't work for me.

Thanks for your time :)

Here is problems:

```
2018-03-03 23:19:37.934 ERROR 42323 --- [           main]
com.zaxxer.hikari.HikariConfig           : HikariPool-1 - dataSource or
dataSourceClassName or jdbcUrl is required.
2018-03-03 23:19:37.938  WARN 42323 --- [           main]
ConfigServletWebServerApplicationContext : Exception encountered during context
initialization - cancelling refresh attempt:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error
creating bean with name 'entityManagerFactory' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Unsatisfied dependency expressed through method 'entityManagerFactory'
parameter 0; nested exception is
org.springframework.beans.factory.UnsatisfiedDependencyException: Error
creating bean with name 'entityManagerFactoryBuilder' defined in class path
resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Unsatisfied dependency expressed through method 'entityManagerFactoryBuilder'
parameter 0; nested exception is
org.springframework.beans.factory.BeanCreationException: Error creating bean
with name 'jpaVendorAdapter' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Bean instantiation via factory method failed; nested exception is
org.springframework.beans.BeanInstantiationException: Failed to instantiate
[org.springframework.orm.jpa.JpaVendorAdapter]: Factory method
'jpaVendorAdapter' threw exception; nested exception is
java.lang.IllegalArgumentException: dataSource or dataSourceClassName or
jdbcUrl is required.
2018-03-03 23:19:37.939  INFO 42323 --- [           main]
o.apache.catalina.core.StandardService   : Stopping service [Tomcat]
2018-03-03 23:19:37.954  INFO 42323 --- [           main]
ConditionEvaluationReportLoggingListener :

Error starting ApplicationContext. To display the conditions report re-run your
application with 'debug' enabled.
```

```
org.springframework.beans.factory.UnsatisfiedDependencyException: Error
creating bean with name 'entityManagerFactory' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Unsatisfied dependency expressed through method 'entityManagerFactory'
parameter 0; nested exception is
org.springframework.beans.factory.UnsatisfiedDependencyException: Error
creating bean with name 'entityManagerFactoryBuilder' defined in class path
resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Unsatisfied dependency expressed through method 'entityManagerFactoryBuilder'
parameter 0; nested exception is
org.springframework.beans.factory.BeanCreationException: Error creating bean
with name 'jpaVendorAdapter' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Bean instantiation via factory method failed; nested exception is
org.springframework.beans.BeanInstantiationException: Failed to instantiate
[org.springframework.orm.jpa.JpaVendorAdapter]: Factory method
'jpaVendorAdapter' threw exception; nested exception is
java.lang.IllegalArgumentException: dataSource or dataSourceClassName or
jdbcUrl is required.
    at
org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFact
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.insta
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCre
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(A
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSinglet(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBe
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBean
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.context.support.AbstractApplicationContext.getBean(AbstractApp
 ~[spring-context-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryIn
 ~[spring-context-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApp
 ~[spring-context-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext.re
 ~[spring-boot-2.0.0.RELEASE.jar:2.0.0.RELEASE]
```

```
    at
org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:
 [spring-boot-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.SpringApplication.run(SpringApplication.java:327)
[spring-boot-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1246)
[spring-boot-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1234)
[spring-boot-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at com.Test.Test.TestApplication.main(TestApplication.java:17)
[classes/:na]
Caused by: org.springframework.beans.factory.UnsatisfiedDependencyException:
Error creating bean with name 'entityManagerFactoryBuilder' defined in class
path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Unsatisfied dependency expressed through method 'entityManagerFactoryBuilder'
parameter 0; nested exception is
org.springframework.beans.factory.BeanCreationException: Error creating bean
with name 'jpaVendorAdapter' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Bean instantiation via factory method failed; nested exception is
org.springframework.beans.BeanInstantiationException: Failed to instantiate
[org.springframework.orm.jpa.JpaVendorAdapter]: Factory method
'jpaVendorAdapter' threw exception; nested exception is
java.lang.IllegalArgumentException: dataSource or dataSourceClassName or
jdbcUrl is required.
    at
org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFact
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.insta
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCre
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(A
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSinglet
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBe
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBean
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.config.DependencyDescriptor.resolveCandidate(Dep
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
```

```
    at
org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDepend
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.resolveAutowiredArgu
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    ... 19 common frames omitted
Caused by: org.springframework.beans.factory.BeanCreationException: Error
creating bean with name 'jpaVendorAdapter' defined in class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]:
 Bean instantiation via factory method failed; nested exception is
org.springframework.beans.BeanInstantiationException: Failed to instantiate
[org.springframework.orm.jpa.JpaVendorAdapter]: Factory method
'jpaVendorAdapter' threw exception; nested exception is
java.lang.IllegalArgumentException: dataSource or dataSourceClassName or
jdbcUrl is required.
    at
org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFact
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.insta
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCre
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.creat
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(A
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSinglet
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBe
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBean
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.config.DependencyDescriptor.resolveCandidate(Dep
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDepe
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDepend
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.resolveAutowiredArgu
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    ... 33 common frames omitted
```

```
java.lang.IllegalArgumentException: dataSource or dataSourceClassName or
jdbcUrl is required.
    at
org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFact
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    ... 46 common frames omitted
Caused by: java.lang.IllegalArgumentException: dataSource or
dataSourceClassName or jdbcUrl is required.
    at com.zaxxer.hikari.HikariConfig.validate(HikariConfig.java:1063) ~
[HikariCP-2.7.8.jar:na]
    at
com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:109) ~
[HikariCP-2.7.8.jar:na]
    at
org.springframework.jdbc.datasource.DataSourceUtils.fetchConnection(DataSourceUti
 ~[spring-jdbc-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.jdbc.datasource.DataSourceUtils.doGetConnection(DataSourceUti
 ~[spring-jdbc-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.jdbc.datasource.DataSourceUtils.getConnection(DataSourceUtils.
 ~[spring-jdbc-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.jdbc.support.JdbcUtils.extractDatabaseMetaData(JdbcUtils.java:
 ~[spring-jdbc-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.jdbc.support.JdbcUtils.extractDatabaseMetaData(JdbcUtils.java:
 ~[spring-jdbc-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.DatabaseLookup.getDatabase(Database
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties.determineDatabase(Jpa
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.JpaBaseConfiguration.jpaVendorAdapt
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration$$Enhancer
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration$$Enhancer
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at
org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:228) ~
[spring-core-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterc
 ~[spring-context-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    at
org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration$$Enhancer
 ~[spring-boot-autoconfigure-2.0.0.RELEASE.jar:2.0.0.RELEASE]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native
Method) ~[na:na]
    at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccesso
 ~[na:na]
    at
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMetho
```

```
org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate
 ~[spring-beans-5.0.4.RELEASE.jar:5.0.4.RELEASE]
    ... 47 common frames omitted


Process finished with exit code 1
```

Here is my pom.xml:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.0.RELEASE</version>
</parent>

<properties>
     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
       <java.version>1.9</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-properties-migrator</artifactId>
    </dependency>
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
    </dependency>
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
    </dependency>
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.7.0</version>
    </dependency>
    <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.7.0</version>
    </dependency>
    <dependency>
```

```
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
    </dependency>
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
            <version>1.4.3.RELEASE</version>
    </dependency>
</dependencies>
```

Here is my application.yml:

```
spring:
  application:
    name: Test

  jpa:
    hibernate:
      ddl-auto: update
      use-new-id-generator-mappings: true
    generate-ddl: true
    properties:
      dialect: org.hibernate.dialect.PostgreSQLDialect

  session:
    store-type: none

  datasource:
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/test
    username: test
    password: 1234

    tomcat:
      max-idle: 2
      max-active: 3
    type: com.zaxxer.hikari.HikariDataSource
```

UPDATE:

My DatabaseConfig.java:

```
@Configuration
public class DatabaseConfig {
    @Bean
    @Primary
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource dataSource() { return DataSourceBuilder.create().build();
}

    @Bean
    public SimpleMailMessage templateSimpleMessage() {
```

```
            return message;
        }
    }
```

java spring spring-boot

Share Improve this question

Follow

edited Mar 4, 2018 at 14:34

asked Mar 3, 2018 at 20:22

Batuhan Kök

1,162 1 9 10

Are you sure your `application.yml` is properly formatted? I don't expect `tomcat` to be a child element of `datasource` – Abhijit Sarkar Mar 3, 2018 at 20:45

Yes I'm sure about that. Also it couldn't be problem because in 1.5.x there was no problem. Thanks :) – Batuhan Kök Mar 3, 2018 at 21:33

1 That's the tomcat connection pool (which was the default in 1.5). The last 4 lines of that yaml are useless as Spring Boot 2 switched to Hikari by default and configuring another connection pool's implementation will have no effect. The exception is weird. Are you configuring the DataSource yourself in code? Can you share a sample? – Stephane Nicoll Mar 4, 2018 at 7:53

@StephaneNicoll thanks for your answer first of all. I'm gonna delete unnecessary lines in application.yml. I've inserted my database config class to the post below of the UPDATE title. – Batuhan Kök Mar 4, 2018 at 13:59

1 I just fix it, thanks for your all positive response. I also delete `dataSource()` method from DatabaseConfig.java. Then, the application was started successfully :) – Batuhan Kök Mar 4, 2018 at 14:33

10 Answers

Sorted by:

Trending sort available

Highest score (default)

167

As this post gets a bit of popularity I edited it a bit. Spring Boot 2.x.x changed default JDBC connection pool from Tomcat to faster and better HikariCP. Here comes incompatibility, because HikariCP uses different property of jdbc url. There are two ways how to handle it:

OPTION ONE

There is very good explanation and workaround in spring docs:

Also, if you happen to have Hikari on the classpath, this basic setup does not work, because Hikari has no url property (but does have a jdbcUrl property). In that case, you must rewrite your configuration as follows:

```
app.datasource.jdbc-url=jdbc:mysql://localhost/test
app.datasource.username=dbuser
```

OPTION TWO

There is also how-to in the docs how to get it working from "both worlds". It would look like below. ConfigurationProperties bean would do "conversion" for `jdbcUrl` from `app.datasource.url`

```
@Configuration
public class DatabaseConfig {
    @Bean
    @ConfigurationProperties("app.datasource")
    public DataSourceProperties dataSourceProperties() {
        return new DataSourceProperties();
    }

    @Bean
    @ConfigurationProperties("app.datasource")
    public HikariDataSource dataSource(DataSourceProperties properties) {
        return
properties.initializeDataSourceBuilder().type(HikariDataSource.class)
                .build();
    }
}
```

Share Improve this answer Follow

edited Jun 1, 2018 at 15:08

answered Mar 6, 2018 at 23:11

lapkritinis
2,424 3 16 27

5 thank you! renaming from app.datasource.url to app.datasource.jdbc-url fixed my problem. – 1housand Jan 2, 2019 at 23:07

2 I am getting 'Duplicated prefix' for both strings passed as a parameter to ConfigurationProperties. Did something changed to do that? – acarlstein Apr 19, 2019 at 14:34

Indeed, renaming the datasource.url to datasource.jdbc-url fixed my issue as well, thank you – user9802118 Sep 30, 2021 at 21:31

got the same issue as @acarlstein mentioned. Also got an issue with "properties" parameter in the second method: "Could not autowire. There is more than one bean of 'DataSourceProperties' type." – FARS Dec 4, 2021 at 10:26

For the OPTION TWO to pick up hikari specific settings, configuration properties can be specified as: `@ConfigurationProperties("app.datasource.hikari")` . More details: stackoverflow.com/a/71474522/854386 – Andrey Mar 14 at 21:40

76

This happened to me because I was using:

```
app.datasource.url=jdbc:mysql://localhost/test
```

When I replaced url by `jdbc-url` then it worked:

Share Improve this answer

Follow

edited Aug 17, 2019 at 12:08

Ihor Patsian

1,270 2 15 24

answered Aug 21, 2018 at 14:24

Vivek Garg

1,857 1 17 17

3 This answer perfectly works fine. No need to have any extra code to make it work as an accepted answer. – Paramesh Korrakuti Mar 5, 2019 at 17:59

1 Verified on 2.1.5.release. works fine with jdbc-url. url does NOT work anymore. – Ross Bu May 28, 2019 at 18:36

Works like charm. Verified on 2.2.5.RELEASE – Heril Muratovic Mar 21, 2020 at 18:00

67

In case you do need to define `dataSource()`, for example when you have multiple data sources, you can use:

```
@Autowired Environment env;

@Primary
@Bean
public DataSource customDataSource() {

    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName(env.getProperty("custom.datasource.driver-
class-name"));
    dataSource.setUrl(env.getProperty("custom.datasource.url"));
    dataSource.setUsername(env.getProperty("custom.datasource.username"));
    dataSource.setPassword(env.getProperty("custom.datasource.password"));

    return dataSource;

}
```

By setting up the `dataSource` yourself (instead of using `DataSourceBuilder`), it fixed my problem which you also had.

The always knowledgeable Baeldung has a tutorial which explains in depth.

Share Improve this answer

Follow

edited Oct 15, 2020 at 6:55

im_infamous

934 1 15 29

answered Mar 6, 2018 at 2:48

BigJ

1,952 2 27 43

It's a solution for his problem, and it happens to also be useful when using multiple data sources. – BigJ Apr 23, 2018 at 20:51

Yeah, it solves the problem occasionally - but Spring Data is build greatly on the principle of convention over configuration, it should therefore be solvable without the need for code :) Your solution is great for multiple databases, I will keep that in mind although I don't hope to enter a situation, where I really need 3 different databases inside one App/Microservice (local H2 + Staging-Database + 3rd Database somewhere). – jonashackt Apr 24, 2018 at 19:35

1 Accepted answer should be improved per documentation on how to deal with multiple sources -

`spring.datasource.jdbc-url=...` . That fixed my problem. More to this can be found at docs.spring.io/spring-boot/docs/current/reference/htmlsingle/... – pixel May 18 at 18:57

13

I also read the Spring docs, as lapkritinis suggested - and luckily this brought me on the right path! But I don´t think, that the Spring docs explain this good right now. At least for me, they aren´t consistent IMHO.

The original problem/question is on what to do, if you upgrade an existing Spring Boot 1.5.x application to 2.0.x, which is using PostgreSQL/Hibernate. The main reason, you get your described error, is that Spring Boot 2.0.x uses HikariCP instead of Tomcat JDBC pooling DataSource as a default - and Hikari´s DataSource doesn´t know the `spring.datasource.url` property, instead it want´s to have `spring.datasource.jdbc-url` (lapkritinis also pointed that out).

So far so good. BUT the docs also suggest - and that´s the problem here - that Spring Boot uses `spring.datasource.url` to determine, if the - often locally used - embedded Database like H2 has to back off and instead use a production Database:

> You should at least specify the URL by setting the spring.datasource.url property. Otherwise, Spring Boot tries to auto-configure an embedded database.

You may see the dilemma. If you want to have your embedded DataBase like you´re used to, you have to switch back to Tomcat JDBC. **This is also much more minimally invasive to existing applications, as you don´t have to change source code!** To get your existing application working after the Spring Boot 1.5.x --> 2.0.x upgrade with PostgreSQL, just add `tomcat-jdbc` as a dependency to your pom.xml:

```
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jdbc</artifactId>
</dependency>
```

And then configure Spring Boot to use it accordingly inside application.properties:

```
spring.datasource.type=org.apache.tomcat.jdbc.pool.DataSource
```

Hope to help some folks with this, was quite a time consuming problem. I also hope my beloved Spring folks update the docs - and the way new Hikari pool is configured - to get a more consistent Spring Boot user experience :)

Share Improve this answer Follow

answered Apr 22, 2018 at 19:55

jonashackt

8,652 3 51 87

1 HikariCP should be better than Tomcat – lapkritinis May 30, 2018 at 12:37

Thanks @jonashackt it helped me. – nani21984 Dec 6, 2019 at 17:36

9

This worked for me.

`application.properties` , used **jdbc-url** instead of **url**:

```
datasource.apidb.jdbc-url=jdbc:mysql://localhost:3306/apidb?useSSL=false
datasource.apidb.username=root
datasource.apidb.password=123
datasource.apidb.driver-class-name=com.mysql.jdbc.Driver
```

Configuration class:

```
@Configuration
@EnableJpaRepositories(
        entityManagerFactoryRef = "fooEntityManagerFactory",
        basePackages = {"com.buddhi.multidatasource.foo.repository"}
)
public class FooDataSourceConfig {

    @Bean(name = "fooDataSource")
    @ConfigurationProperties(prefix = "datasource.foo")
    public HikariDataSource dataSource() {
        return DataSourceBuilder.create().type(HikariDataSource.class).build();
    }

    @Bean(name = "fooEntityManagerFactory")
    public LocalContainerEntityManagerFactoryBean fooEntityManagerFactory(
            EntityManagerFactoryBuilder builder,
            @Qualifier("fooDataSource") DataSource dataSource
    ) {
        return builder
                .dataSource(dataSource)
                .packages("com.buddhi.multidatasource.foo.model")
                .persistenceUnit("fooDb")
                .build();
    }
}
```

Share Improve this answer

Follow

edited Aug 17, 2019 at 14:23

Ihor Patsian

1,270 2 15 24

answered Aug 17, 2019 at 8:49

Buddhi

2,104 5 31 41

8

Configure Two DataSources in `Spring Boot 2.0.*` or above

If you need to configure multiple data sources, you have to mark one of the DataSource instances as `@Primary` , because various auto-configurations down the road expect to be able to get one by type.

primary data source:

```
@Bean
@Primary
@ConfigurationProperties("app.datasource.first")
public DataSourceProperties firstDataSourceProperties() {
    return new DataSourceProperties();
}

@Bean
@Primary
@ConfigurationProperties("app.datasource.first")
public DataSource firstDataSource() {
    return firstDataSourceProperties().initializeDataSourceBuilder().build();
}

@Bean
@ConfigurationProperties("app.datasource.second")
public BasicDataSource secondDataSource() {
    return DataSourceBuilder.create().type(BasicDataSource.class).build();
}
```

> **`firstDataSourceProperties`** has to be flagged as **`@Primary`** so that the database initializer feature uses your copy (if you use the initializer).

And your `application.propoerties` will look something like this:

```
app.datasource.first.url=jdbc:oracle:thin:@localhost/first
app.datasource.first.username=dbuser
app.datasource.first.password=dbpass
app.datasource.first.driver-class-name=oracle.jdbc.OracleDriver

app.datasource.second.url=jdbc:mariadb://localhost:3306/springboot_mariadb
app.datasource.second.username=dbuser
app.datasource.second.password=dbpass
app.datasource.second.driver-class-name=org.mariadb.jdbc.Driver
```

The above method is the correct to way to init multiple database in spring boot 2.0 migration and above. More read can be found here.

Share Improve this answer

Follow

edited May 3, 2019 at 7:36

answered Sep 15, 2018 at 20:18

Amar Prakash Pandey

1,128 14 22

With spring-boot 2.1.4 (latest version) This approach doesn't work when you've two different data sources that have different drivers. One using a jdbc MariDB driver & another using an ibm DB2Driver... – S34N Apr 21, 2019 at 8:34

@S34N I have updated my answer, I hope this resolves your query. – Amar Prakash Pandey May 3, 2019 at 7:38

3

```
@Bean
@ConfigurationProperties("app.datasource")
public DataSource dataSource() {
    return DataSourceBuilder.create().build();
}
```

application.properties I have added

```
app.datasource.url=jdbc:mysql://localhost/test
app.datasource.username=dbuser
app.datasource.password=dbpass
app.datasource.pool-size=30
```

More details Configure a Custom DataSource

Share Improve this answer Follow

answered Mar 30, 2020 at 14:43

Ganesh Giri
974 9 16

2

Others have answered so I'll add my 2-cents.

You can either use autoconfiguration (i.e. don't use a @Configuration to create a datasource) or java configuration.

Auto-configuration:

define your datasource type then set the type properties. E.g.

```
spring.datasource.type=com.zaxxer.hikari.HikariDataSource
spring.datasource.hikari.driver-class-name=org.h2.Driver
spring.datasource.hikari.jdbc-url=jdbc:h2:mem:testdb
spring.datasource.hikari.username=sa
spring.datasource.hikari.password=password
spring.datasource.hikari.max-wait=10000
spring.datasource.hikari.connection-timeout=30000
spring.datasource.hikari.idle-timeout=600000
spring.datasource.hikari.max-lifetime=1800000
spring.datasource.hikari.leak-detection-threshold=600000
spring.datasource.hikari.maximum-pool-size=100
spring.datasource.hikari.pool-name=MyDataSourcePoolName
```

Java configuration:

Choose a prefix and define your data source

```
spring.mysystem.datasource.type=com.zaxxer.hikari.HikariDataSource
spring.mysystem.datasource.jdbc-
url=jdbc:sqlserver://databaseserver.com:18889;Database=MyDatabase;
spring.mysystem.datasource.username=dsUsername
```

```
spring.mysystem.datasource.max-wait=10000
spring.mysystem.datasource.connection-timeout=30000
spring.mysystem.datasource.idle-timeout=600000
spring.mysystem.datasource.max-lifetime=1800000
spring.mysystem.datasource.leak-detection-threshold=600000
spring.mysystem.datasource.maximum-pool-size=100
spring.mysystem.datasource.pool-name=MySystemDatasourcePool
```

Create your datasource bean:

```
@Bean(name = { "dataSource", "mysystemDataSource" })
@ConfigurationProperties(prefix = "spring.mysystem.datasource")
public DataSource dataSource() {
    return DataSourceBuilder.create().build();
}
```

You can leave the datasource type out, but then you risk spring guessing what datasource type to use.

Share Improve this answer Follow

answered Jun 19, 2020 at 1:52

Dean McNabb

41 2

1

Your can use DataSourceBuilder for this purpose.

```
@Primary
@Bean(name = "dataSource")
@ConfigurationProperties(prefix = "spring.datasource")
public DataSource dataSource(Environment env) {
    final String datasourceUsername =
env.getRequiredProperty("spring.datasource.username");
    final String datasourcePassword =
env.getRequiredProperty("spring.datasource.password");
    final String datasourceUrl =
env.getRequiredProperty("spring.datasource.url");
    final String datasourceDriver =
env.getRequiredProperty("spring.datasource.driver-class-name");
    return DataSourceBuilder
            .create()
            .username(datasourceUsername)
            .password(datasourcePassword)
            .url(datasourceUrl)
            .driverClassName(datasourceDriver)
            .build();
}
```

Share Improve this answer Follow

answered Nov 27, 2020 at 14:11

makson

1,465 3 19 30

0

```
database.properties:

jdbcUrl=jdbc:mysql://localhost:3306/candelete
dataSource.user=root
dataSource.password=
dataSource.cachePrepStmts=true
dataSource.prepStmtCacheSize=250
dataSource.prepStmtCacheSqlLimit=2048
```

Share Improve this answer Follow

answered Dec 31, 2021 at 19:47

Maninder

740 6 11