# **SpringBoot2数据库连接池自动装配原理，以及如何配置使用其他的数据库连接池（druid）为例**

**SpringBoot2[数据库连接池](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E5%BA%93%E8%BF%9E%E6%8E%A5%E6%B1%A0&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/jike11231/article/details/)自动装配原理**

[一、SpringBoot的数据库连接池的相关默认](https://blog.csdn.net/jike11231/article/details/120722718" \l "SpringBoot_2" \t "https://blog.csdn.net/jike11231/article/details/_self)

[二、SpringBoot默认的数据库连接池，以及自动装配原理](https://blog.csdn.net/jike11231/article/details/120722718" \l "SpringBoot_8" \t "https://blog.csdn.net/jike11231/article/details/_self)

[三、使用其他的数据库连接池：例如Druid](https://blog.csdn.net/jike11231/article/details/120722718" \l "Druid_20" \t "https://blog.csdn.net/jike11231/article/details/_self)

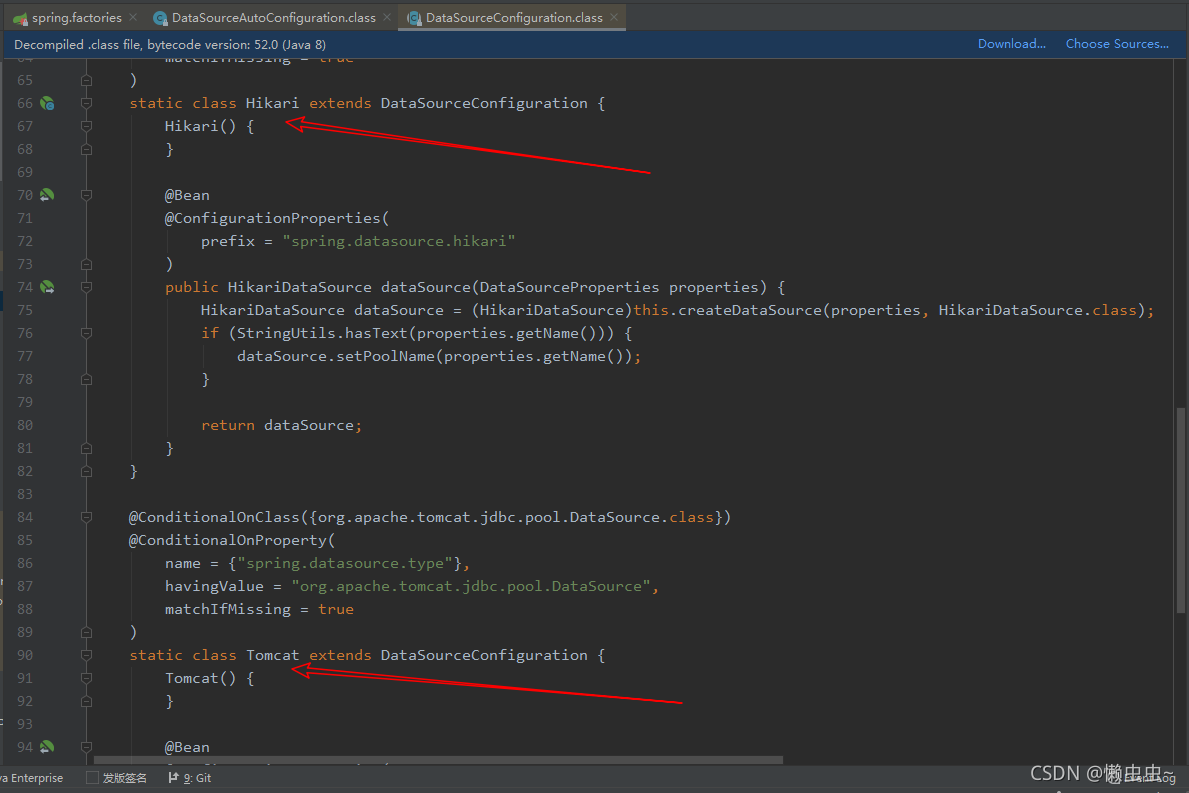
## 一、SpringBoot的数据库连接池的相关默认

SpringBoot之前的版本默认使用的是Tomcat的数据库连接池

较新的版本2.X，默认使用的是Hikari（我使用的是2.2.1），具体可以通过DataSourceConfiguration查看

默认支持数据库连接池：

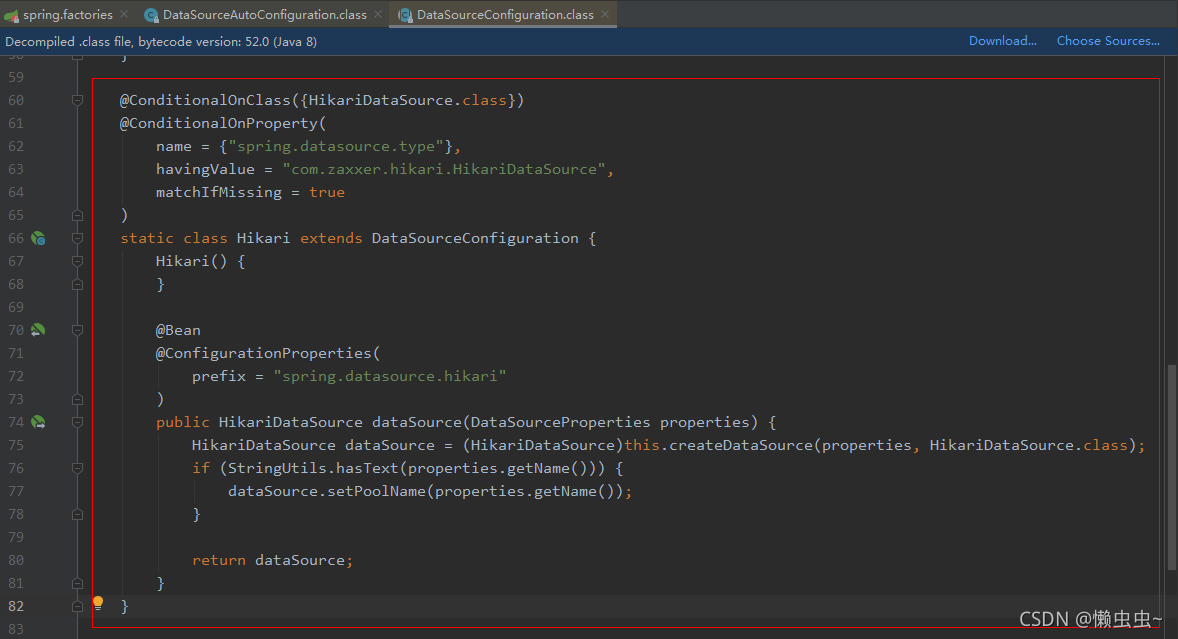
Tomcat、Hikari、Dbcp2



## **二、SpringBoot默认的数据库连接池，以及自动装配原理**

通过查看数据库自动装配类：org.springframework.boot.autoconfigure.jdbc.DataSourceConfiguration

我们截取其中的一个Hikari的代码装配过程，其他两个默认支持的数据源原理一样：



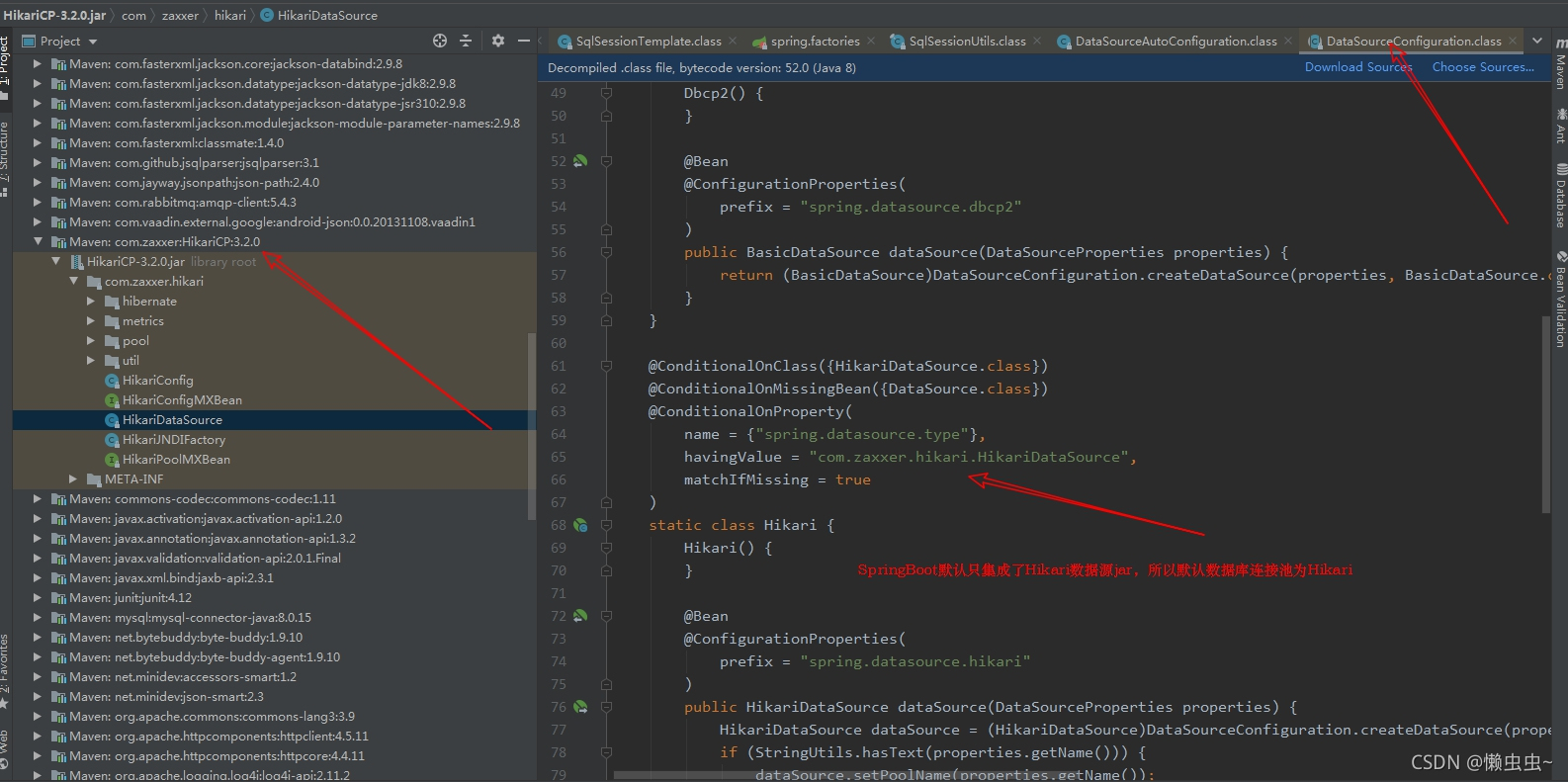
2.1 首先这是个静态成员内部类Hikari，并使用了@Configuration，里面只有一个方法dataSource并且该方法使用@Bean；于是我们知道该类目的是生成一个Hikari连接池的bean。

2.2 @ConditionalOnClass({HikariDataSource.class})：HikariDataSource在类路径上存在时，@Configuration才生效。

@ConditionalOnMissingBean({DataSource.class})：DataSource的bean不存在时,@Configuration才生效，避免生成多个dataSource。

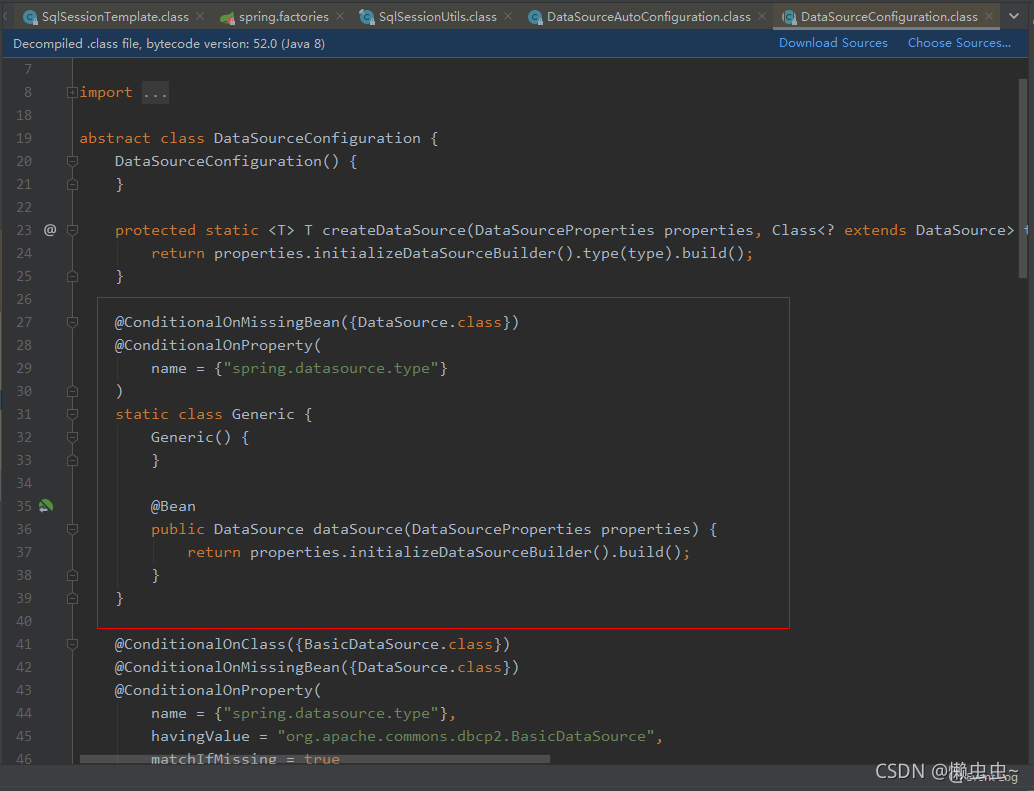
@ConditionalOnProperty：从配置文件读取name指定的属性的值与havingValue相等时，@Configuration才生效，但havingValue = "true"使得当读取的值为空时该判断条件返回true，也就是说你不配置name指定的属性，该条件返回true，如果配置了name指定的属性，则与havingValue的值比较返回true或false。

2.3 DataSourceConfiguration中只集成了Hikari的jar包，SpringBoot默认使用Hikari数据库连接池



2.4 @ConfigurationProperties(prefix = “spring.datasource.hikari”)：自动为生成的bean（DataSource）注入配置文件的属性配置，如初始连接数，最大最小等等。

## **三、使用其他的数据库连接池：例如[Druid](https://so.csdn.net/so/search?q=Druid&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/jike11231/article/details/_blank)**



当我们在配置文件配置使用其他非默认数据库连接池时，SpringBoot默认使用的数据库连接池Hikari不会生成使用，转而会生成我们配置使用的数据库连接池；

3.1 添加相关的依赖

<dependency>

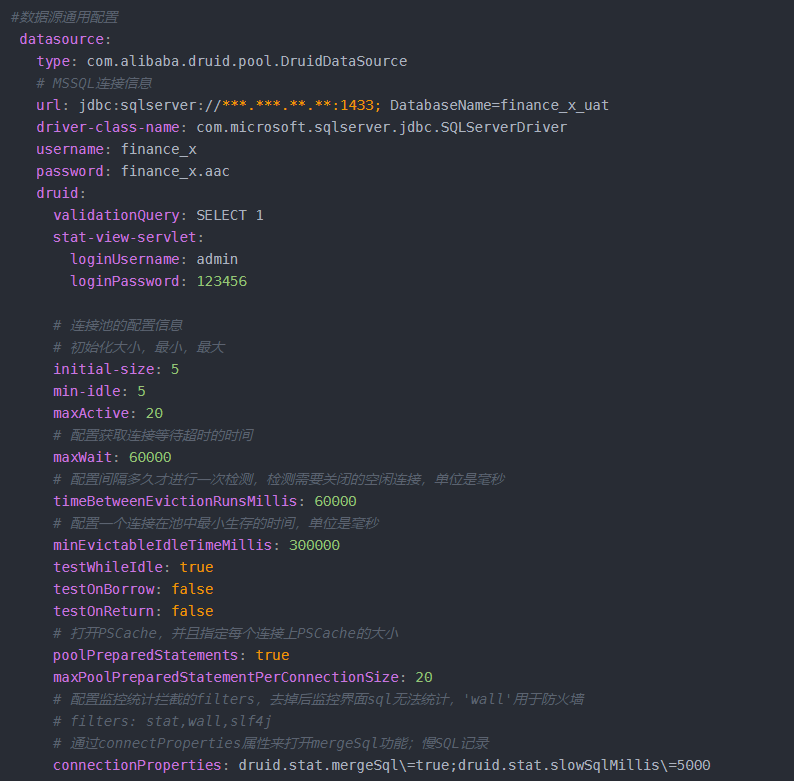
<groupId>com.alibaba</groupId>

<artifactId>druid-spring-boot-starter</artifactId>

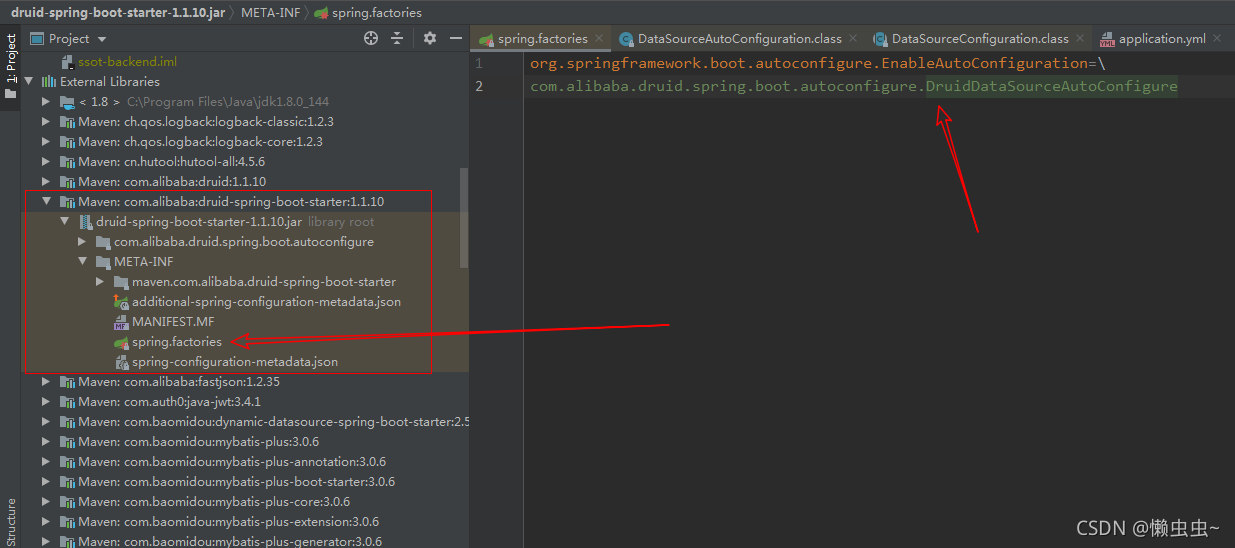
<version>1.1.10</version>

</dependency>

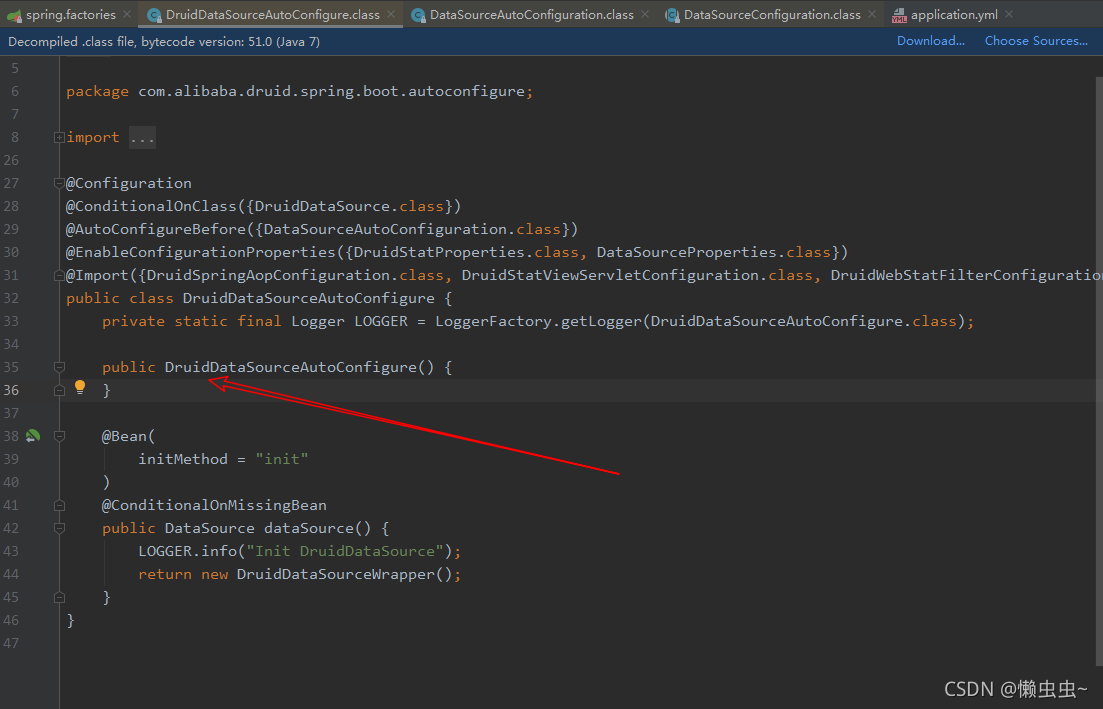
3.2 配置文件



3.3 添加完依赖完成后  
找spring.factories的自动配置类



分析Druid的DruidDataSourceAutoConfigure类



(1)@ConditionalOnClass(DruidDataSource.class)必须有DruidDataSource包才加载。

(2)@AutoConfigureBefore(DataSourceAutoConfiguration.class) 必须在DataSourceAutoConfiguration之前配置。因为 DataSourceAutoConfiguration默认是配置的HikariDataSource数据源，所以这里必须在之前配置。

(3)@EnableConfigurationProperties({DruidStatProperties.class, DataSourceProperties.class}) 开启了一些配置信息的数据项绑定。

(4)@Import下的4个Configuration

DruidSpringAopConfiguration Druid对SpringAOP的支持自动配置 可以自定匹配一些Spring组件的SQL监控规则

DruidStatViewServletConfiguration 自动化配置StatServlet视图

DruidWebStatFilterConfiguration 监控规则相关自动化配置

DruidFilterConfiguration 防火墙相关的自动化配置

@Bean(initMethod = “init”) //初始化方法init，在父类的DruidDataSource中,该方法定义了Druid的各种初始化操作并返回。