自：https://www.cnblogs.com/hfultrastrong/p/8595161.html

[**SpringBoot2 集成三种连接池 c3p0 hikari druid**](https://www.cnblogs.com/hfultrastrong/p/8595161.html)

**Hikari**

1、首先集成 hikari springboot默认集成，只需要简单的配置即可

1.1 首先导入包

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <dependency>      <groupId>com.zaxxer</groupId>      <artifactId>HikariCP</artifactId>  </dependency>  <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency> |

1.2 编写配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5 | # hikari  #spring.datasource.url=jdbc:mysql://127.0.0.1:3306/springboot2  #spring.datasource.username=root  #spring.datasource.password=root  #spring.datasource.driver-class-name=com.mysql.jdbc.Driver |

1.3 编写配置类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package com.archibaldwitwicke.springboot2.chapter05.dao;    import org.springframework.context.annotation.Configuration;    @Configuration  public class DataSourceConfiguration {        // Hikari 连接池  //    @Bean(name = "dataSource")  //    public DataSource dataSource(@Autowired Environment environment) {  //        HikariDataSource ds = new HikariDataSource();  //        ds.setJdbcUrl(environment.getProperty("spring.datasource.url"));  //        ds.setUsername(environment.getProperty("spring.datasource.username"));  //        ds.setPassword(environment.getProperty("spring.datasource.password"));  //        ds.setDriverClassName(environment.getProperty("spring.datasource.driver-class-name"));  //        return ds;  //    }  } |

1.4 然后即可即可 jdbctemplate模板使用。

**C3P0**

1 导入相应包

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <dependency>      <groupId>c3p0</groupId>      <artifactId>c3p0</artifactId>      <version>0.9.1.2</version>  </dependency>  <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency> |

2 编写相应配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | # c3p0  #c3p0.jdbcUrl=jdbc:mysql://127.0.0.1:3306/springboot2  #c3p0.user=root  #c3p0.password=root  #c3p0.driverClass=com.mysql.jdbc.Driver  #c3p0.minPoolSize=2  #c3p0.maxPoolSize=10  #c3p0.maxIdleTime=1800000  #c3p0.acquireIncrement=3  #c3p0.maxStatements=1000  #c3p0.initialPoolSize=3  #c3p0.idleConnectionTestPeriod=60  #c3p0.acquireRetryAttempts=30  #c3p0.acquireRetryDelay=1000  #c3p0.breakAfterAcquireFailure=false  #c3p0.testConnectionOnCheckout=false |

3 编写数据源类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package com.archibaldwitwicke.springboot2.chapter05.dao;    import org.springframework.context.annotation.Configuration;    @Configuration  public class DataSourceConfiguration {        // c3p0 连接池  //    @Bean(name = "dataSource")  //    @Qualifier(value = "dataSource")  //    @Primary  //    @ConfigurationProperties(prefix = "c3p0")  //    public DataSource dataSource(@Autowired Environment environment) {  //        return DataSourceBuilder.create().type(com.mchange.v2.c3p0.ComboPooledDataSource.class).build();  //    }  } |

4 然后即可注入 jdbctemplate 操作数据库

**Druid**

1 导入相应包（需要日志包）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <dependency>      <groupId>com.alibaba</groupId>      <artifactId>druid</artifactId>      <version>1.0.29</version>  </dependency>  <dependency>      <groupId>log4j</groupId>      <artifactId>log4j</artifactId>      <version>1.2.16</version>      <scope>compile</scope>  </dependency>  <dependency>      <groupId>mysql</groupId>      <artifactId>mysql-connector-java</artifactId>  </dependency>  <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency> |

2 编写配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # durid  # 驱动配置信息  spring.datasource.url = jdbc:mysql://127.0.0.1:3306/springboot2  spring.datasource.username = root  spring.datasource.password = root  spring.datasource.driverClassName = com.mysql.jdbc.Driver  #连接池的配置信息  spring.datasource.initialSize=5  spring.datasource.minIdle=5  spring.datasource.maxActive=20  spring.datasource.maxWait=60000  spring.datasource.timeBetweenEvictionRunsMillis=60000  spring.datasource.minEvictableIdleTimeMillis=300000  spring.datasource.validationQuery=SELECT 1 FROM DUAL  spring.datasource.testWhileIdle=true  spring.datasource.testOnBorrow=false  spring.datasource.testOnReturn=false  spring.datasource.poolPreparedStatements=true  spring.datasource.maxPoolPreparedStatementPerConnectionSize=20  spring.datasource.filters=stat,wall,log4j  spring.datasource.connectionProperties=druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000 |

还需要log4j配置：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | log4j.rootCategory=INFO, stdout  log4j.rootLogger=info, stdout    ### stdout ###  log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.Target=System.out  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p - %m%n    ### set package ###  log4j.logger.org.springframework=info  log4j.logger.org.apache.catalina=info  log4j.logger.org.apache.commons.digester.Digester=info  log4j.logger.org.apache.catalina.startup.TldConfig=info  log4j.logger.chb.test=debug |

3 编写数据源类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103 | package com.archibaldwitwicke.springboot2.chapter05.dao;    import com.alibaba.druid.pool.DruidDataSource;  import org.springframework.beans.factory.annotation.Value;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import org.springframework.context.annotation.Primary;    import javax.sql.DataSource;  import java.sql.SQLException;    @Configuration  public class DruidDataSourceConfiguration {  //    private Logger logger = LoggerFactory.getLogger(DruidDataSourceConfiguration.class);        @Value("${spring.datasource.url}")      private String dbUrl;        @Value("${spring.datasource.username}")      private String username;        @Value("${spring.datasource.password}")      private String password;        @Value("${spring.datasource.driverClassName}")      private String driverClassName;        @Value("${spring.datasource.initialSize}")      private int initialSize;        @Value("${spring.datasource.minIdle}")      private int minIdle;        @Value("${spring.datasource.maxActive}")      private int maxActive;        @Value("${spring.datasource.maxWait}")      private int maxWait;        @Value("${spring.datasource.timeBetweenEvictionRunsMillis}")      private int timeBetweenEvictionRunsMillis;        @Value("${spring.datasource.minEvictableIdleTimeMillis}")      private int minEvictableIdleTimeMillis;        @Value("${spring.datasource.validationQuery}")      private String validationQuery;        @Value("${spring.datasource.testWhileIdle}")      private boolean testWhileIdle;        @Value("${spring.datasource.testOnBorrow}")      private boolean testOnBorrow;        @Value("${spring.datasource.testOnReturn}")      private boolean testOnReturn;        @Value("${spring.datasource.poolPreparedStatements}")      private boolean poolPreparedStatements;        @Value("${spring.datasource.maxPoolPreparedStatementPerConnectionSize}")      private int maxPoolPreparedStatementPerConnectionSize;        @Value("${spring.datasource.filters}")      private String filters;        @Value("{spring.datasource.connectionProperties}")      private String connectionProperties;        @Bean     //声明其为Bean实例      @Primary  //在同样的DataSource中，首先使用被标注的DataSource      public DataSource dataSource() {          DruidDataSource datasource = new DruidDataSource();            datasource.setUrl(this.dbUrl);          datasource.setUsername(username);          datasource.setPassword(password);          datasource.setDriverClassName(driverClassName);            //configuration          datasource.setInitialSize(initialSize);          datasource.setMinIdle(minIdle);          datasource.setMaxActive(maxActive);          datasource.setMaxWait(maxWait);          datasource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);          datasource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);          datasource.setValidationQuery(validationQuery);          datasource.setTestWhileIdle(testWhileIdle);          datasource.setTestOnBorrow(testOnBorrow);          datasource.setTestOnReturn(testOnReturn);          datasource.setPoolPreparedStatements(poolPreparedStatements);          datasource.setMaxPoolPreparedStatementPerConnectionSize(maxPoolPreparedStatementPerConnectionSize);          try {              datasource.setFilters(filters);          } catch (SQLException e) {  //            logger.error("druid configuration initialization filter", e);              e.printStackTrace();          }          datasource.setConnectionProperties(connectionProperties);            return datasource;      }  } |

4 即可使用 jdbctemplate

以上所有数据源，根据需要导入数据驱动， 测试使用的为msyql