## ****DRUID介绍****

    DRUID是阿里巴巴开源平台上一个数据库连接池实现，它结合了C3P0、DBCP、PROXOOL等DB池的优点，同时加入了日志监控，可以很好的监控DB池连接和SQL的执行情况，可以说是针对监控而生的DB连接池(据说是目前最好的连接池,不知道速度有没有BoneCP快)。

## ****配置参数****

和其它连接池一样DRUID的DataSource类为：com.alibaba.druid.pool.DruidDataSource，基本配置参数如下：

|  |  |  |
| --- | --- | --- |
| 配置 | 缺省值 | 说明 |
| name |  | 配置这个属性的意义在于，如果存在多个数据源，监控的时候可以通过名字来区分开来。  如果没有配置，将会生成一个名字，格式是："DataSource-" + System.identityHashCode(this) |
| jdbcUrl |  | 连接数据库的url，不同数据库不一样。例如：  mysql : jdbc:mysql://10.20.153.104:3306/druid2  oracle : jdbc:oracle:thin:@10.20.149.85:1521:ocnauto |
| username |  | 连接数据库的用户名 |
| password |  | 连接数据库的密码。如果你不希望密码直接写在配置文件中，可以使用ConfigFilter。详细看这里：https://github.com/alibaba/druid/wiki/%E4%BD%BF%E7%94%A8ConfigFilter |
| driverClassName | 根据url自动识别 | 这一项可配可不配，如果不配置druid会根据url自动识别dbType，然后选择相应的driverClassName(建议配置下) |
| initialSize | 0 | 初始化时建立物理连接的个数。初始化发生在显示调用init方法，或者第一次getConnection时 |
| maxActive | 8 | 最大连接池数量 |
| maxIdle | 8 | 已经不再使用，配置了也没效果 |
| minIdle |  | 最小连接池数量 |
| maxWait |  | 获取连接时最大等待时间，单位毫秒。配置了maxWait之后，缺省启用公平锁，并发效率会有所下降，如果需要可以通过配置useUnfairLock属性为true使用非公平锁。 |
| poolPreparedStatements | false | 是否缓存preparedStatement，也就是PSCache。PSCache对支持游标的数据库性能提升巨大，比如说oracle。在mysql下建议关闭。 |
| maxOpenPreparedStatements | -1 | 要启用PSCache，必须配置大于0，当大于0时，poolPreparedStatements自动触发修改为true。在Druid中，不会存在Oracle下PSCache占用内存过多的问题，可以把这个数值配置大一些，比如说100 |
| validationQuery |  | 用来检测连接是否有效的sql，要求是一个查询语句。如果validationQuery为null，testOnBorrow、testOnReturn、testWhileIdle都不会其作用。 |
| testOnBorrow | true | 申请连接时执行validationQuery检测连接是否有效，做了这个配置会降低性能。 |
| testOnReturn | false | 归还连接时执行validationQuery检测连接是否有效，做了这个配置会降低性能 |
| testWhileIdle | false | 建议配置为true，不影响性能，并且保证安全性。申请连接的时候检测，如果空闲时间大于timeBetweenEvictionRunsMillis，执行validationQuery检测连接是否有效。 |
| timeBetweenEvictionRunsMillis |  | 有两个含义：  1) Destroy线程会检测连接的间隔时间2) testWhileIdle的判断依据，详细看testWhileIdle属性的说明 |
| numTestsPerEvictionRun |  | 不再使用，一个DruidDataSource只支持一个EvictionRun |
| minEvictableIdleTimeMillis |  |  |
| connectionInitSqls |  | 物理连接初始化的时候执行的sql |
| exceptionSorter | 根据dbType自动识别 | 当数据库抛出一些不可恢复的异常时，抛弃连接 |
| filters |  | 属性类型是字符串，通过别名的方式配置扩展插件，常用的插件有：  监控统计用的filter:stat日志用的filter:log4j防御sql注入的filter:wall |
| proxyFilters |  | 类型是List<com.alibaba.druid.filter.Filter>，如果同时配置了filters和proxyFilters，是组合关系，并非替换关系 |

* 使用方法

DB数据源的使用方法也就是2种，一种是在代码中写死通过NEW操作符创建DataSSource，然后set一些连接属性，这里不在累述;另外一种是基于SPRING的配置方法，然后让SPRING的Context自动加载配置（以下配置文件默认都在项目根目录下conf文件夹中）

1、属性文件:application.properties(DataSource连接参数)

jdbc.driverClassName=com.mysql.jdbc.Driver   
jdbc.url=jdbc:mysql://127.0.0.1:3306/test   
jdbc.username=root   
jdbc.password=1qaz!QAZ

2、SPRING配置文件：spring-base.xml

<?xml version="1.0" encoding="UTF-8"?>   
<beans xmlns=" <http://www.springframework.org/schema/beans>"   
 xmlns:xsi=" <http://www.w3.org/2001/XMLSchema-instance>" xmlns:batch=" <http://www.springframework.org/schema/batch>"   
 xsi:schemaLocation=" <http://www.springframework.org/schema/beans>   
           <http://www.springframework.org/schema/beans/spring-beans-4.0.xsd>">

 <bean id="propertyConfigure"   
  class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">   
  <property name="locations">   
   <list>   
    <value>./conf/application.properties</value>   
   </list>   
  </property>   
 </bean>

 <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"   
  init-method="init" destroy-method="close">   
  <property name="driverClassName" value="${jdbc.driverClassName}" />   
  <property name="url" value="${jdbc.url}" />   
  <property name="username" value="${jdbc.username}" />   
  <property name="password" value="${jdbc.password}" />   
  <!-- 配置初始化大小、最小、最大 -->   
  <property name="initialSize" value="1" />   
  <property name="minIdle" value="1" />   
  <property name="maxActive" value="10" />

  <!-- 配置获取连接等待超时的时间 -->   
  <property name="maxWait" value="10000" />

  <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->   
  <property name="timeBetweenEvictionRunsMillis" value="60000" />

  <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->   
  <property name="minEvictableIdleTimeMillis" value="300000" />

  <property name="testWhileIdle" value="true" />

  <!-- 这里建议配置为TRUE，防止取到的连接不可用 -->   
  <property name="testOnBorrow" value="true" />   
  <property name="testOnReturn" value="false" />

  <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->   
  <property name="poolPreparedStatements" value="true" />   
  <property name="maxPoolPreparedStatementPerConnectionSize"   
   value="20" />

  <!-- 这里配置提交方式，默认就是TRUE，可以不用配置 -->

  <property name="defaultAutoCommit" value="true" />

  <!-- 验证连接有效与否的SQL，不同的数据配置不同 -->   
  <property name="validationQuery" value="select 1 " />   
  <property name="filters" value="stat" />   
  <property name="proxyFilters">   
   <list>   
    <ref bean="logFilter" />   
   </list>   
  </property>   
 </bean>

 <bean id="logFilter" class="com.alibaba.druid.filter.logging.Slf4jLogFilter">   
  <property name="statementExecutableSqlLogEnable" value="false" />   
 </bean>

</beans>

上面红色标注部分为监控DB池连接执行监控，后面在做详细说明.

* 监控方式

1、WEB方式监控配置

<servlet>   
     <servlet-name>DruidStatView</servlet-name>   
     <servlet-class>com.alibaba.druid.support.http.StatViewServlet</servlet-class>   
 </servlet>   
 <servlet-mapping>   
     <servlet-name>DruidStatView</servlet-name>   
     <url-pattern>/druid/\*</url-pattern>   
 </servlet-mapping>   
 <filter>   
  <filter-name>druidWebStatFilter</filter-name>   
  <filter-class>com.alibaba.druid.support.http.WebStatFilter</filter-class>   
  <init-param>   
   <param-name>exclusions</param-name>   
   <param-value>/public/\*,\*.js,\*.css,/druid\*,\*.jsp,\*.swf</param-value>   
  </init-param>   
  <init-param>   
   <param-name>principalSessionName</param-name>   
   <param-value>sessionInfo</param-value>   
  </init-param>   
  <init-param>   
   <param-name>profileEnable</param-name>   
   <param-value>true</param-value>   
  </init-param>   
 </filter>   
 <filter-mapping>   
  <filter-name>druidWebStatFilter</filter-name>   
  <url-pattern>/\*</url-pattern>   
 </filter-mapping>

把上面servlet配置添加到项目web.xml即可。然后运行Tomcat，浏览器输入 [http://IP:PROT/druid](http://ipprot/)

就可以打开Druid的监控页面了.

2、日志文件监控

Druid提供了多种日志文件监控 commons-logging、log4j等，这里我们主要使用slf4j和logback来进行日志监控配置。

首先要引入slf4j和logback相关的jar文件（从Maven公共仓库下载 <http://search.maven.org/>）

<slf4j.version>1.7.7</slf4j.version>   
<logback.version>1.1.2</logback.version>

<dependency>   
    <groupId>org.slf4j</groupId>   
    <artifactId>slf4j-api</artifactId>   
    <version>${slf4j.version}</version>   
 </dependency>   
<dependency>   
    <groupId>ch.qos.logback</groupId>   
    <artifactId>logback-access</artifactId>   
    <version>${logback.version}</version>   
</dependency>   
<dependency>   
    <groupId>ch.qos.logback</groupId>   
    <artifactId>logback-core</artifactId>   
    <version>${logback.version}</version>   
</dependency>   
<dependency>   
    <groupId>ch.qos.logback</groupId>   
    <artifactId>logback-classic</artifactId>   
    <version>${logback.version}</version>   
</dependency>

接下配置logback的配置文件(./conf/logback.xml)

<configuration>

 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">   
  <layout class="ch.qos.logback.classic.PatternLayout">   
   <Pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n   
   </Pattern>   
  </layout>   
 </appender>

 <appender name="FILE" class="ch.qos.logback.core.FileAppender">   
  <file>./logs/druid\_info.log</file>   
  <layout class="ch.qos.logback.classic.PatternLayout">   
   <Pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</Pattern>   
  </layout>   
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">   
   <level>debug</level>   
  </filter>   
 </appender>

 <root level="DEBUG">   
  <appender-ref ref="FILE" />   
 </root>   
</configuration>

最后就是写一个测试类进行测试

public class TestMain {

 public static void loadLoggerContext() {   
  System.getProperties().put("logback.configurationFile", "./conf/logback.xml");   
  LoggerContext lc = (LoggerContext) LoggerFactory.getILoggerFactory();   
  StatusPrinter.setPrintStream(System.err);   
  StatusPrinter.print(lc);   
 }

 public static void main(String[] args) {   
  try {   
   loadLoggerContext();   
   FileSystemXmlApplicationContext context = new FileSystemXmlApplicationContext("./conf/spring-base.xml");   
  
  } catch (Exception e) {   
   System.out.println(e);   
  }   
 }   
}