Maven pom.xml添加依赖

<!--druid-->  
<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.0.29</version>  
</dependency>  
<!--h2数据库驱动-->  
<dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <version>1.4.194</version>  
</dependency>

配置数据源相关信息

# 数据源配置  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.url=jdbc:h2:file:D:/WorkSpaceIDEA/springboot-learning-demo/db/demo  
spring.datasource.username=admin  
spring.datasource.password=admin  
###################以下为druid增加的配置###########################  
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource  
# 下面为连接池的补充设置，应用到上面所有数据源中  
# 初始化大小，最小，最大  
spring.datasource.initialSize=5  
spring.datasource.minIdle=5  
spring.datasource.maxActive=20  
# 配置获取连接等待超时的时间  
spring.datasource.maxWait=60000  
# 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  
spring.datasource.timeBetweenEvictionRunsMillis=60000  
# 配置一个连接在池中最小生存的时间，单位是毫秒  
spring.datasource.minEvictableIdleTimeMillis=300000  
spring.datasource.validationQuery=SELECT 1 FROM DUAL  
spring.datasource.testWhileIdle=true  
spring.datasource.testOnBorrow=false  
spring.datasource.testOnReturn=false  
# 打开PSCache，并且指定每个连接上PSCache的大小  
spring.datasource.poolPreparedStatements=true  
spring.datasource.maxPoolPreparedStatementPerConnectionSize=20  
# 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙  
spring.datasource.filters=stat,wall,log4j  
# 通过connectProperties属性来打开mergeSql功能；慢SQL记录  
spring.datasource.connectionProperties=druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000  
# 合并多个DruidDataSource的监控数据  
#spring.datasource.useGlobalDataSourceStat=true  
###############以上为配置druid添加的配置########################################

数据源配置：

DruidDataSourceConfig.java

@Configuration  
@ConditionalOnClass({DruidDataSource.class})//（某个class位于类路径上，才会实例化一个Bean）  
@ConditionalOnProperty(//指定的属性是否有执行的值  
 name = {"spring.datasource.type"},  
 havingValue = "com.alibaba.druid.pool.DruidDataSource",  
 matchIfMissing = true  
)  
public class DruidDataSourceConfig {  
  
 @Bean(name = "dataSource", initMethod = "init", destroyMethod = "close")  
 @Primary  
 @ConfigurationProperties(prefix = "spring.datasource")  
 public DataSource druidDataSource() {  
 DruidDataSource druidDataSource = new DruidDataSource();  
 return druidDataSource;  
 }  
}

监控有两种配置方式：

方式一：

DruidStatViewServlet.java文件

*/\*\*  
 \* druid数据源状态监控.  
 \*/*@WebServlet(urlPatterns = "/druid/\*",  
 initParams = {  
 @WebInitParam(name = "allow", value = "192.168.1.72,127.0.0.1"),// IP白名单 (没有配置或者为空，则允许所有访问)  
 @WebInitParam(name = "deny", value = "192.168.1.73"),// IP黑名单 (存在共同时，deny优先于allow)  
 @WebInitParam(name = "loginUsername", value = "admin"),// 用户名  
 @WebInitParam(name = "loginPassword", value = "admin"),// 密码  
 @WebInitParam(name = "resetEnable", value = "false")// 禁用HTML页面上的“Reset All”功能  
 })  
public class DruidStatViewServlet extends StatViewServlet {  
 private static final long *serialVersionUID* = -7799874061866519411L;  
}

DruidStatFilter.java文件

*/\*\*  
 \* druid过滤器.  
 \*/*@WebFilter(filterName = "druidWebStatFilter", urlPatterns = "/\*",  
 initParams = {  
 @WebInitParam(name = "exclusions", value = "\*.js,\*.gif,\*.jpg,\*.bmp,\*.png,\*.css,\*.ico,/druid/\*")// 忽略资源  
 }  
)  
public class DruidStatFilter extends WebStatFilter {  
}

最后在启动类上添加@ServletComponentScan使spring能够扫描到我们自己编写的servlet和filter。

@SpringBootApplication  
@ServletComponentScan//用于扫描servlet和filter  
public class SpringbootDruidApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringbootDruidApplication.class, args);  
 }  
}

访问地址<http://localhost:8080/druid/>

第二种方式

使用代码注册Servlet

@Configuration  
public class DruidStatConfig {  
  
 */\*\*  
 \* 注册一个StatViewServlet  
 \*  
 \** ***@return*** *\*/* @Bean  
 public ServletRegistrationBean druidStatViewServle() {  
 // org.springframework.boot.context.embedded.ServletRegistrationBean提供类的进行注册.  
 ServletRegistrationBean servletRegistrationBean = new ServletRegistrationBean(new StatViewServlet(), "/druid/\*");  
 // 添加初始化参数：initParams  
 servletRegistrationBean.addInitParameter("loginUsername", "admin");// 用户名  
 servletRegistrationBean.addInitParameter("loginPassword", "admin");// 密码  
 servletRegistrationBean.addInitParameter("resetEnable", "false");// 禁用HTML页面上的“Reset All”功能  
 servletRegistrationBean.addInitParameter("allow", "127.0.0.1"); // IP白名单 (没有配置或者为空，则允许所有访问)  
 servletRegistrationBean.addInitParameter("deny", "192.168.20.38");// IP黑名单(存在共同时，deny优先于allow)  
 return servletRegistrationBean;  
 }  
  
 */\*\*  
 \* druid过滤器  
 \*  
 \** ***@return*** *\*/* @Bean  
 public FilterRegistrationBean filterRegistrationBean() {  
 FilterRegistrationBean filterRegistrationBean = new FilterRegistrationBean(new WebStatFilter());  
 // 添加过滤规则.  
 filterRegistrationBean.addUrlPatterns("/\*");  
 // 添加不需要忽略的格式信息.  
 filterRegistrationBean.addInitParameter("exclusions", "\*.js,\*.gif,\*.jpg,\*.bmp,\*.png,\*.css,\*.ico,/druid/\*");  
 return filterRegistrationBean;  
 }  
}

启动应用直接访问就行 <http://localhost:8080/druid/>