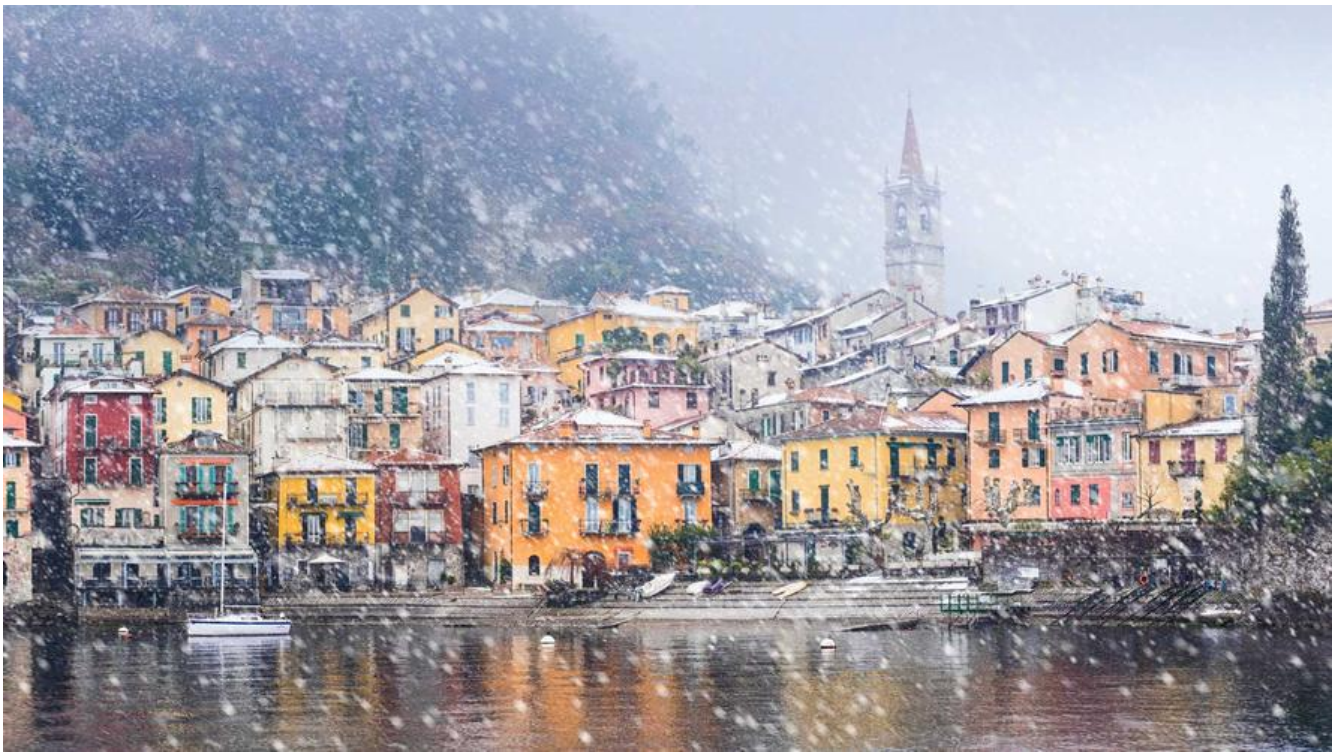
使用 p6spy 格式化日志输出

作者：jianzh5
原文链接：https://ld246.com/article/1576158481173
来源网站：链滴



P6Spy 是针对数据库访问操作的动态监测框架（为开源项目，项目首 页：www.p6spy.com）它使得据库数据可无缝截取和操纵，而不必对现有应用程序的代码作任何修改。P6Spy 分发包包括P6Log，是一 个可记录任何 Java 应用程序的所有JDBC事务的应用程序。其配置完成使用时，可以进行数据访性能的监测。

我们最需要的功能，查看sql语句，不是预编译的带问号的哦，而是真正的数据库执行的sql，更直观更简单。

Step1、引入相关jar包

```
<dependency>
  <groupId>p6spy</groupId>
  <artifactId>p6spy</artifactId>
  <version>3.6.0</version>
</dependency>
```

Step2、配置日志监控

```
<!--p6spy 监控-->
<bean id="dataSource" class="com.p6spy.engine.spy.P6DataSource">
    <constructor-arg ref="dataSourceTarget"></constructor-arg>
</bean>
<!--配置数据源-->
<bean id="dataSourceTarget" class="com.alibaba.druid.pool.DruidDataSource" init-method=
init"
     destroy-method="close">
    <!-- 基本属性 url、user、password -->
    <property name="driverClassName" value="${jdbc_driver}" />
    <property name="url" value="${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"/>
```

原文链接：使用 p6spy 格式化日志输出

```
    <property name="minIdle" value="1"/>
    <property name="maxActive" value="20"/>
    <!-- 配置获取连接等待超时的时间 -->
    <property name="maxWait" value="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 -->
    <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="validationQuery" value="SELECT 'x'"/>
    <property name="testWhileIdle" value="true"/>
    <property name="testOnBorrow" value="false"/>
    <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"/>
    <!-- 配置监控统计拦截的filters，去掉后监控界面sql无法统计 -->
    <property name="filters" value="stat"/>

</bean>
```

Step3、在classpath下加入配置文件 spy.properties

Step4、自定义日志输出格式

这里使用logback作为日志处理，可以通过修改spy.properties中的appender属性进行配置

```java
import com.p6spy.engine.logging.Category;
import com.p6spy.engine.spy.appender.FormattedLogger;
import com.p6spy.engine.spy.appender.P6Logger;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
* <p>
* <code>LogbackLogger</code>
* </p>
* 自定义查询日志输出
* @version 1.0
* @Date: 2016/5/6 14:24
* @since 1.0
*/
public class LogbackLogger extends FormattedLogger implements P6Logger{
    private static final Logger logger = LoggerFactory.getLogger("p6spy");

    public String getLastEntry() {
        return lastEntry;
    }

    public void setLastEntry(String lastEntry) {
        this.lastEntry = lastEntry;
    }
```

```
    protected String lastEntry;

    @Override
    public void logSQL(int connectionId, String s, long l, Category category, String s1,String sql)
{
        if (!"resultset".equals(category)) {
            logger.info(trim(sql));
        }
    }

    @Override
    public void logException(Exception e) {
        logger.error(e.getMessage(),e);
    }

    @Override public void logText(String s) {
        logger.info(s);
        this.setLastEntry(s);
    }

    @Override public boolean isCategoryEnabled(Category category) {
        return true;
    }

    private String trim(String sql){
        StringBuilder sb = new StringBuilder("\r\n");
        sb.append(sql.replaceAll("\n|\r|\t|  '"," "));
        return sb.toString();
    }

}
```

打印结果：