**1. Spring 拦截器**

<http://blog.csdn.net/tonytfjing/article/details/39207551>

**项目中使用拦截器，配合Java注解，实现用户行为跟踪**

**postHandle方法里面关键代码：**

if (handler instanceof HandlerMethod) {

HandlerMethod method = (HandlerMethod) handler;

Desc desc = method.getMethodAnnotation(Desc.class); //获取方法上的注解

if(desc!= null && StringUtils.isNotEmpty(desc.value())

&& StringUtil.isNotEmpty(desc.descType())) {

uriDesc = desc.value();

descType = desc.descType();

}

}

record(request, uri, uriDesc, descType, source); //用户行为记录到数据库（访问过哪些url，行为描述等）

**2. Spring 过滤器**

**3. Spring Aop**

**4. 缓存（Redis）**

**5. 全局错误捕获**

**6. 发邮件**

**7. 发短息**

**8. 定时任务**

**9. Spring 实现多数据库及读写分离方法**

**10. Druid连接池实现web监控**

第1步：pom文件

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>${druid-version}</version>  
</dependency>

第2步：打开Druid的监控统计功能：

Druid的监控统计功能是通过filter-chain扩展实现，如果你要打开监控统计功能，需要配置StatFilter

<!—自己定制的StatLogger、可以将druid记录到文件和日志打印等操作-->

<bean id="statLogger" class="com.lizhihui.druid.StatLogger"></bean>

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"  
 init-method="init" destroy-method="close">  
 <property name="url">  
 <value>${jdbc.url}</value>  
 </property>  
 <property name="username">  
 <value>${jdbc.username}</value>  
 </property>  
 <property name="password">  
 <value>${jdbc.password}</value>  
 </property>  
 <property name="initialSize">  
 <value>${jdbc.initialSize}</value>  
 </property>  
 <property name="maxActive">  
 <value>${jdbc.maxActive}</value>  
 </property>

<!—关键： 监控统计拦截的filters -->  
 <property name="proxyFilters">  
 <list>  
 <ref bean="stat-filter" />  
 <ref bean="wall-filter" />  
 </list>  
 </property>  
</bean>  
  
<bean id="stat-filter" class="com.alibaba.druid.filter.stat.StatFilter">  
 <property name="slowSqlMillis" value="1000" />  
 <property name="logSlowSql" value="true" /> <!-- 慢SQL记录-->   
 <property name="mergeSql" value="true" />  
</bean>  
  
<bean id="wall-filter" class="com.alibaba.druid.wall.WallFilter">  
 <property name="dbType" value="mysql" />  
 <property name="config" ref="wall-filter-config" />  
</bean>  
  
<bean id="wall-filter-config" class="com.alibaba.druid.wall.WallConfig">  
 <property name="multiStatementAllow" value="true" />  
</bean>

<!-- 配置druid监控spring jdbc -->    
<bean id="druid-stat-interceptor" class="com.alibaba.druid.support.spring.stat.DruidStatInterceptor"></bean>  
<bean id="druid-stat-pointcut" class="org.springframework.aop.support.JdkRegexpMethodPointcut" scope="prototype">  
 <property name="patterns">  
 <list>  
 <value>io.renren.modules.\*.controller.\*</value>  
 <value>io.renren.modules.\*.service.\*</value>  
 </list>  
 </property>  
</bean>  
<aop:config>  
 <aop:advisor advice-ref="druid-stat-interceptor" pointcut-ref="druid-stat-pointcut"/>  
</aop:config>  
  
<bean id="transactionManager"  
 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource">  
 <ref bean="dataSource" />  
 </property>  
</bean>  
  
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="configLocation" value="classpath:mybatis.xml" />  
 <property name="mapperLocations" value="classpath:mapper/\*\*/\*.xml" />  
 <property name="dataSource" ref="dataSource" />  
</bean>  
  
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <property name="basePackage" value="io.renren.modules.\*.dao" />  
</bean>

第3步：在web.xml文件后，添加

<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>

之后，就可以打开 <http://10.206.2.248:8082/druid/sql.html> 看到数据库各种信息了

此处还可以配置登录账户和密码，如果不配置，则无账户密码登录监控页面，如下：

<servlet>  
 <servlet-name>DruidStatView</servlet-name>  
 <servlet-class>com.alibaba.druid.support.http.StatViewServlet</servlet-class>  
 <!—

deny优先于allow，如果在deny列表中，就算在allow列表中，也会被拒绝。  
 如果allow没有配置或者为空，则允许所有访问

-->  
 <init-param>  
 <param-name>allow</param-name>  
 <param-value>128.242.127.1/24,127.0.0.1</param-value>  
 </init-param>  
 <init-param>  
 <param-name>deny</param-name>  
 <param-value>192.168.1.118</param-value>  
 </init-param>  
 <!-- 在StatViewSerlvet输出的html页面中，有一个功能是Reset All，执行这个操作之后，会导致所有计数器清零，重 新计数 -->  
 <init-param>  
 <param-name>resetEnable</param-name>  
 <param-value>false</param-value>  
 </init-param>  
  
 <init-param>  
 <param-name>loginUsername</param-name>  
 <param-value>druid</param-value>  
 </init-param>  
 <init-param>  
 <param-name>loginPassword</param-name>  
 <param-value>druid</param-value>  
 </init-param>  
</servlet>  
  
<servlet-mapping>  
 <servlet-name>DruidStatView</servlet-name>  
 <url-pattern>/druid/\*</url-pattern>  
</servlet-mapping>

第4步：可以自定义类，实现数据库信息log记录，只要在第1步的配置：

<bean id="statLogger" class="com.lizhihui.druid.StatLogger"></bean>

然后编写这个类的代码

/\*\*

\* druid监控sql信息(activeCount)日志打印

\* Created by lizhihui on 2017/10/17.

\*/

public class StatLogger extends DruidDataSourceStatLoggerAdapter {

private static Log LOG = LogFactory.getLog(StatLogger.class);

private Log logger = LOG;

@Setter

@Getter

private Map<String, Object> map = new LinkedHashMap<>();

public StatLogger(){

this.configFromProperties(System.getProperties());

}

/\*\*

\* @since 0.2.21

\*/

@Override

public void configFromProperties(Properties properties) {

String property = properties.getProperty("druid.stat.loggerName");

if (property != null && property.length() > 0) {

setLoggerName(property);

}

}

public Log getLogger() {

return logger;

}

@Override

public void setLoggerName(String loggerName) {

logger = LogFactory.getLog(loggerName);

}

@Override

public void setLogger(Log logger) {

if (logger == null) {

throw new IllegalArgumentException("logger can not be null");

}

this.logger = logger;

}

public boolean isLogEnable() {

return logger.isInfoEnabled();

}

public void log(String value) {

logger.info(value);

}

@Override

public void log(DruidDataSourceStatValue statValue) {

if (!isLogEnable()) {

return;

}

int activeCount = statValue.getActiveCount();

map.put("activeCount", activeCount);

String maxActive = PropertyUtils.getProperty("hibernate.properties", "jdbc.max\_size");

if (StringUtil.isNotEmpty(maxActive) && (activeCount == Integer.valueOf(maxActive))) {

String text = JSONUtils.toJSONString(map);

log(text);

}

}

}

**11. MyBatis**

**12. Spirng多环境打包方法**

**13. 权限管理（shrio）**

**14. Http协议**

**15. Linux基本使用**

**16. 项目发布脚本**

**17. Mysql知识及Mysql压测方法**

**18. poi导出百万数据的方法**

**19. spring mvc中常用的封装类**

**19.1 BeanUtils**

**20.如何区分ajax请求**

<http://blog.csdn.net/javajiawei/article/details/50563154>

1、以X打头的头域作为非HTTP标准协议，一般是某种技术的出现而产生或者某个组织指定的， X-Requested-With头域就是用来判断一个请求是传统的HTTP请求，还是Ajax请求。也就是说Ajax的请求一般都会带上X-Requested-With头域。

2、可以利用下面的代码区分是否是ajax请求：

if(request.getHeader("X-Requested-With") != null

&&request.getHeader("X-Requested-With").equalsIgnoreCase("XMLHttpRequest")){

out.print("该请求是 AJAX 异步HTTP请求。");

} else {

out.print("该请求是传统的 同步HTTP请求。");

}