开发笔记day3

# 分页:

## 框架的分页： Interceptor实现

1. 如果是springboot, 则可以直接引入 pagehelper-spring-boot-starter,
2. 使用的时候，只需在查询list前，调用 startPage 设置分页信息，即可使用分页功能。

String pageNum = request.getParameter("pageNum");

String pageSize = request.getParameter("pageSize");

try {

pageNow = Integer.parseInt(pageNum);

size = Integer.parseInt(pageSize);

} catch (NumberFormatException e) {

return BaseModel.getInstance(null);

}

PageHelper.startPage(pageNow, size);

1. 如果需要，可以重新实现这个分页控制器，通过实现Interceptor类:

@Component

@Intercepts({

@Signature(type = StatementHandler.class, method = "prepare",

args = {Connection.class, Integer.class}),

@Signature(type = ResultSetHandler.class, method = "handleResultSets",

args = {Statement.class})})

public class PageHelper implements Interceptor {

}

## Mybatis分页使用pagehelper:

第一步 配置分页插件

//分页插件

@Bean

public PaginationInnerInterceptor paginationInnerInterceptor(){

return new PaginationInnerInterceptor();

}

第二步 在需要时使用:

Page<User> page = new Page<>();

//把所有数据封装到Page里

mapper.selectPage(page,null);

# 逻辑删除和物理删除

**物理删除**： 对数据库里的一张表中数据实现delete操作时，这行数据确确实实的被删除了（无法查到，恢复不便）

但当希望只是对这行数据进行“隐形操作”，并非确实删除时，需要**逻辑删除**。

逻辑删除的简单实现：

1. 在数据库和实体层中加一个标识字段：isDelete 为tinyint类型，0表示没有逻辑删除，1表示从逻辑上删除了.
2. 字段上加上注解: @Tablelogic 和 @TableFiled(fill = FiledFill.INSERT)
3. 3.1.1版本以下的mp配置逻辑删除插件，新版本不需要

@Bean

public ISqlInjector sqlInjector(){

return new LogicSqlInjector();

}

之后在对这个表进行删除时，其实做的是更新操作（把isDeleted字段变1）

那如果需要把逻辑删除的数据显示出来，或者从被删除状态恢复，则需要在xml手写SQL

# SQL性能分析

**Mybatisplus在3.2版本以下：**

在application.properties插入

spring.profiles.active=dev

采用Mybatisplus的插件PerformanceInterceptor分析性能。两个参数：

maxTime：执行sql如果超过这个时间，自动停止，帮助debug

format：SQL是否格式化？默认为false

在配置类中

@Bean

@Profile({“dev”,”test”})

Public PerformanceInterceptor performanceInterceptor(){

PerformanceInterceptor p = new PerformanceIntercepetor();

p.setMaxTime(100);//超过100ms不执行

p.setFormat(true);//格式化SQL

return p;

}

**Mybatisplus 3.2版本以上这个插件已经移除，可以使用p6spy**

pom.xml：

<dependency>

<groupId>p6spy</groupId>

<artifactId>p6spy</artifactId>

<version>3.8.0</version>

</dependency>

添加spy.properties文件在resource下

module.log=com.p6spy.engine.logging.P6LogFactory,com.p6spy.engine.outage.P6OutageFactory

# 自定义日志打印

logMessageFormat=com.baomidou.mybatisplus.extension.p6spy.P6SpyLogger

#日志输出到控制台

appender=com.baomidou.mybatisplus.extension.p6spy.StdoutLogger

# 使用日志系统记录 sql

#appender=com.p6spy.engine.spy.appender.Slf4JLogger

# 设置 p6spy driver 代理

deregisterdrivers=true

# 取消JDBC URL前缀

useprefix=true

# 配置记录 Log 例外,可去掉的结果集有error,info,batch,debug,statement,commit,rollback,result,resultset.

excludecategories=info,debug,result,batch,resultset

# 日期格式

dateformat=yyyy-MM-dd HH:mm:ss

# 实际驱动可多个

#driverlist=org.h2.Driver

# 是否开启慢SQL记录

outagedetection=true

# 慢SQL记录标准 2 秒

outagedetectioninterval=2

application.properties:

加上

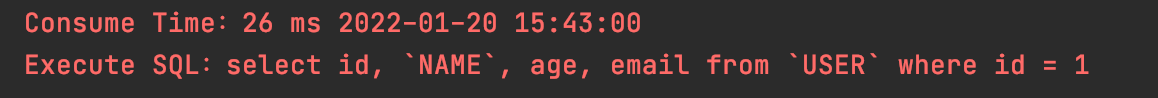
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

切换连接数据库引擎：

spring.datasource.driver-class-name=**com.p6spy.engine.spy.P6SpyDriver**

spring.datasource.url=jdbc:**p6spy**:mysql://localhost:3306/mybatis\_plus?serverTimezone=GMT%2B8

测试结果:



今日解决的bug：

1. 修复了BaseMapper无法使用的问题

报错：Invalid bound statement (not found):

原因：在调用包的时候，以前使用的包为

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.2</version>

</dependency>

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatisplus-spring-boot-starter</artifactId>

<version>1.0.5</version>

</dependency>

修正后：

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>3.3.1</version>

</dependency>

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus</artifactId>

<version>3.5.0</version>

</dependency>

统一改成了mybatis-plus. 注：之前mybatisplus-spring-boot-starter无法选择SqlSession或者SqlFactory，所以无法使用会报错。之前使用的mybatis-spring-boot-starter来暂时代替.

现在改成了mybatis-plus-spring-boot-starter.