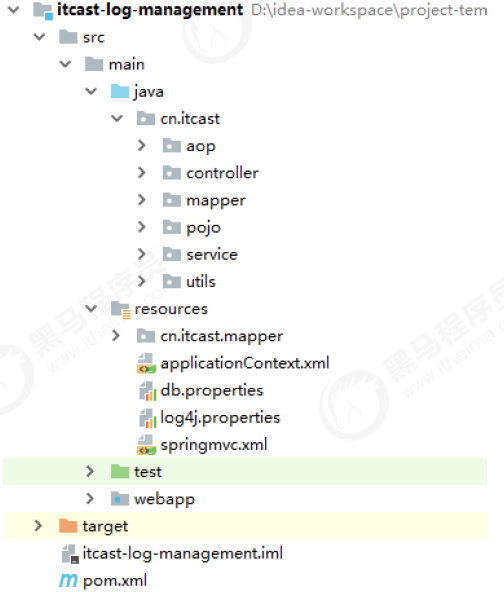
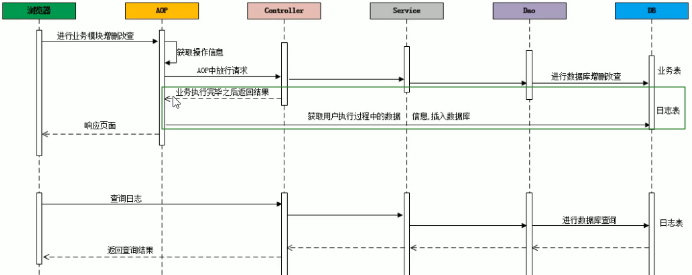
* 日志业务模块工程：



* 日志信息业务：

在业务系统中，需要记录当前业务系统的访问日志，该访问日志包含：操作人，操作时间，访问类，访问方法，请求参数，请求结果，请求结果类型，请求时长 等信息。记录详细的系统访问日志，主要便于对系统中的用户请求进行追踪，并且在系统 的管理后台可以查看到用户的访问记录。

记录系统中的日志信息，可以通过Spring 框架的AOP来实现。具体的请求处理流程，如下：



在AOP中拿到业务操作请求和结果，插入到日志表中并且响应页面。

不管是商品，品牌，还是用户的增删改查，都要记录到操作日志表中，这张表的日志量是非常巨大的。

* AOP拦截原理：

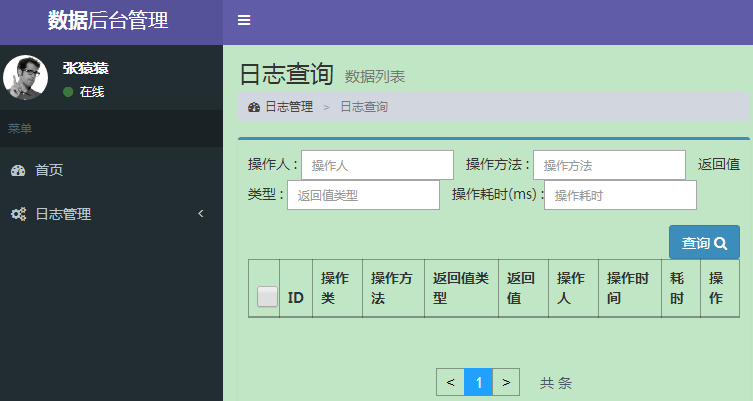
通过自定义注解@OperateLog，来标示方法需不需要进行记录日志。

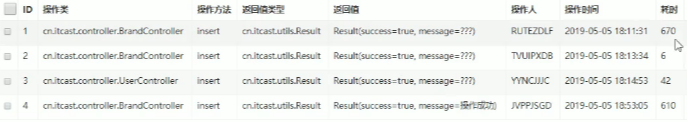
在AOP通知类中拦截指定包下的并且标注有这个自定义注解的方法。

所以，如果一个方法在访问时需要记录日志，则在该方法上标示该注解既可。

* 前端技术：

前端代码使用 BootStrap + AdminLTE 进行布局， 使用Vuejs 进行视图层展示。





* 性能优化方案：

系统中用户访问日志的数据量，随着时间的推移，这张表的数据量会越来越大，因此我们需要根据业务需求，来对日志查询模块的性能进行优化。

1） 分页查询优化

由于在进行日志查询时，是进行分页查询，那也就意味着，在查看时，至少需要查询两次：

A. 查询符合条件的总记录数。--> count 操作

B. 查询符合条件的列表数据。--> 分页查询 limit 操作

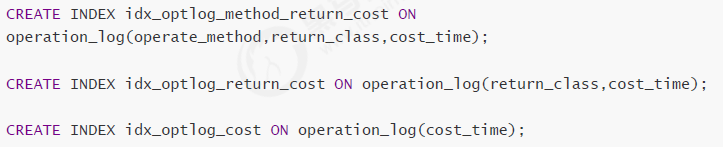
count() 都需要扫描大量的行（意味着需要访问大量的数据）才能获得精确的结果，创建一张log\_counter表用来记录日志表的总数据量。

每当添加新数据的时候，总数据量加一。

2） 条件查询优化

针对于条件查询及排序字段建立索引。

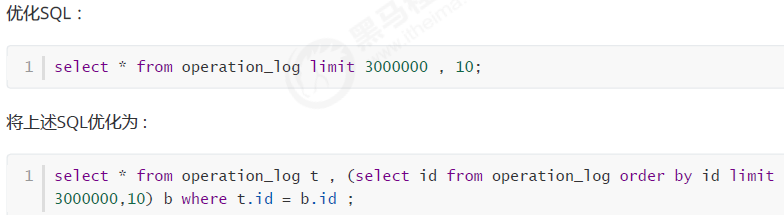
对 操作方法、返回值类型、操作耗时 等字段进行创建索引，以提高查询效率。



Limit优化：

在进行分页时，一般通过创建覆盖索引，能够比较好的提高性能。一个非常常见，而又非常头疼的分页场景就是 "limit 1000000,10" ，此时MySQL需要搜索出前1000010 条记录后，仅仅需要返回第 1000001 到 1000010 条记录，前1000000 记录会被抛弃，查询代价非常大。

当点击比较靠后的页码时，就会出现这个问题，查询效率非常慢。



3）读写分离

通过主从复制集群，来完成读写分离，使写操作走主节点， 而读操作，走从节点。

对于读写分离的实现，本项目通过Spring AOP 来进行动态的切换数据源。

* 表结构说明：

CREATE TABLE `brand` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) DEFAULT NULL COMMENT '品牌名称',

`first\_char` varchar(1) DEFAULT NULL COMMENT '品牌首字母',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `item` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '商品id',

`title` varchar(100) NOT NULL COMMENT '商品标题',

`price` double(10,2) NOT NULL COMMENT '商品价格，单位为：元',

`num` int(10) NOT NULL COMMENT '库存数量',

`categoryid` bigint(10) NOT NULL COMMENT '所属类目，叶子类目',

`status` varchar(1) DEFAULT NULL COMMENT '商品状态，1-正常，2-下架，3-删除',

`sellerid` varchar(50) DEFAULT NULL COMMENT '商家ID',

`createtime` datetime DEFAULT NULL COMMENT '创建时间',

`updatetime` datetime DEFAULT NULL COMMENT '更新时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品表';

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(45) NOT NULL,

`password` varchar(96) NOT NULL,

`name` varchar(45) NOT NULL,

`birthday` datetime DEFAULT NULL,

`sex` char(1) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

`phone` varchar(45) DEFAULT NULL,

`qq` varchar(32) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `operation\_log` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`operate\_class` varchar(200) DEFAULT NULL COMMENT '操作类',

`operate\_method` varchar(200) DEFAULT NULL COMMENT '操作方法',

`return\_class` varchar(200) DEFAULT NULL COMMENT '返回值类型',

`operate\_user` varchar(20) DEFAULT NULL COMMENT '操作用户',

`operate\_time` varchar(20) DEFAULT NULL COMMENT '操作时间',

`param\_and\_value` varchar(500) DEFAULT NULL COMMENT '请求参数名及参数值',

`cost\_time` bigint(20) DEFAULT NULL COMMENT '执行方法耗时, 单位 ms',

`return\_value` varchar(200) DEFAULT NULL COMMENT '返回值',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

create table log\_counter(

logcount bigint not null

)engine = innodb default CHARSET = utf8;