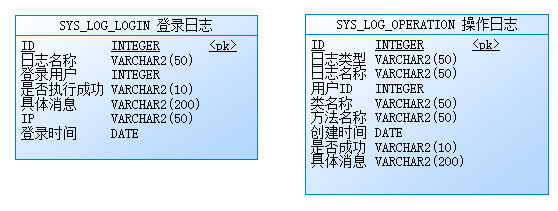
平台日志

日志作为MIS系统的共性需求，在之前的基础平台中也做了相关的设计实现。在这轮的平台重构中，我们结合各项目组提出的建议和需要，对日志做了新的设计和实现，让开发人员在使用平台的日志时更加方便。

## 数据库设计调整



日志对象关系相对简单，对于平台的日志只区分登录和操作两种。对于业务日志的扩展，可以借助操作日志中的日志类型来实现。

## 详细核心设计

**日志工厂类——LogFactory**

由于日志生产过程相对统一和频繁，因此设计了日志工厂LogFactory来创建日志。该工厂类有两个静态方法createLoginLog，createOperaionLog分别用来创建登录日志和操作日志。该工厂类相对简单，请大家参考代码自行了解。

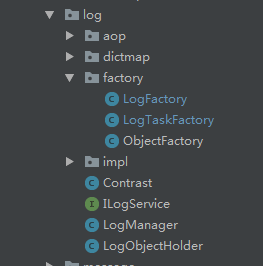
**日志管理器——LogManager**

日志的记录过程由于跟业务逻辑不是紧耦合的关系，所以一般在实现上使用异步的方式，这样能将对系统运行效率的影响降到最低。设计**LogManager**就是为了统一管理异步的实现方案。**LogManager**是一个单例的实现，只有一个最核心的方法executeLog——执行记录日志的任务。该方法会从线程池中获取空闲的线程，然后执行记录任务。

**日志任务工厂类——LogTaskFactory**

上面日志管理器在执行记录任务时，需要任务对象的参数（TimerTask实例），因此设计日志任务工厂类来生产日志任务。该工厂类也相对简单，提供了5个静态返回来创建不同类型的日志任务，请大家参考代码自行了解。

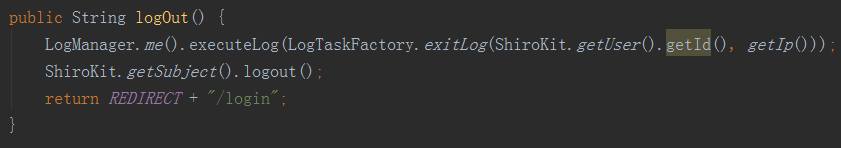
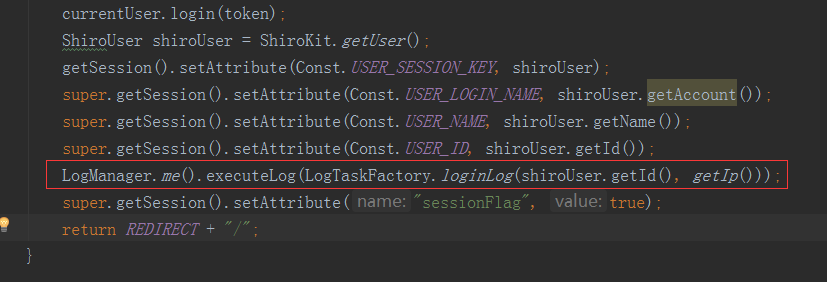
日志的核心设计主要就是上面三个类，在具体的使用场景下还有其他的设计，下文会继续介绍，下图为日志组件的目录结构。



## 案例

**登录日志**

这是使用相对单一的场景，一般在实现系统登录和登出时主动记录的日志场景，代码实现如下图。



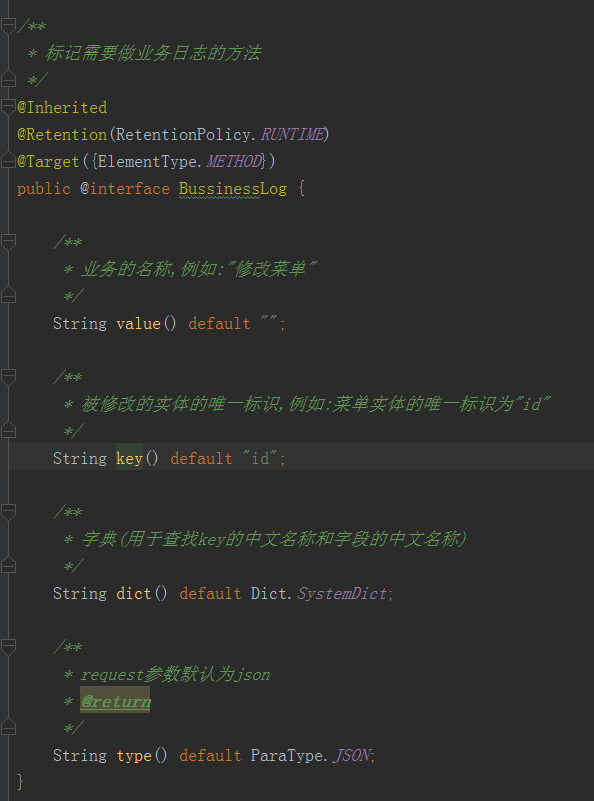
**操作日志**

操作日志相对登录日志要复杂的多，因为操作日志会涉及不同的业务，需要记录的内容也不同，使用的场景和频率也较高。但是我们发现其实大部分的操作日志也有它内在的相同之处，对于相同的地方，我们可以利用抽象地方式，提供统一的实现，而对于个性的地方进行参数或者配置化的设定。

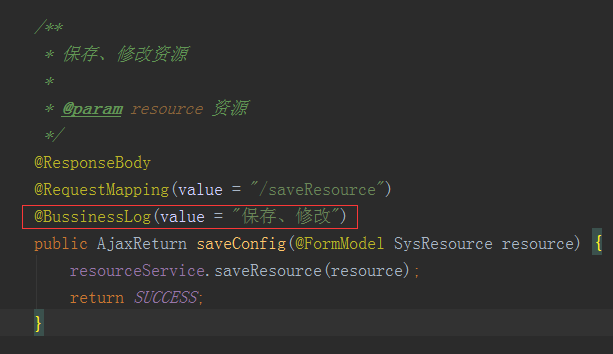
对于业务日志，平台提供了通过注解的方式快速生产日志的方式。

@BusinessLog

具体的参数说明见下图。



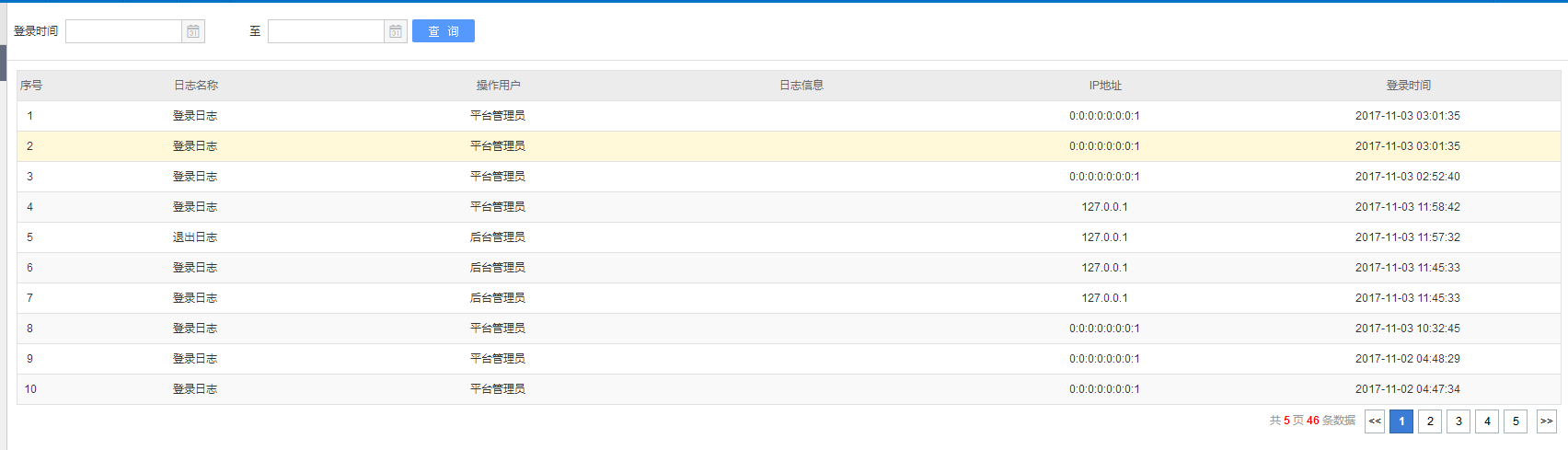
对于简单的业务记录，只需要指定value值来指认日志的名称。

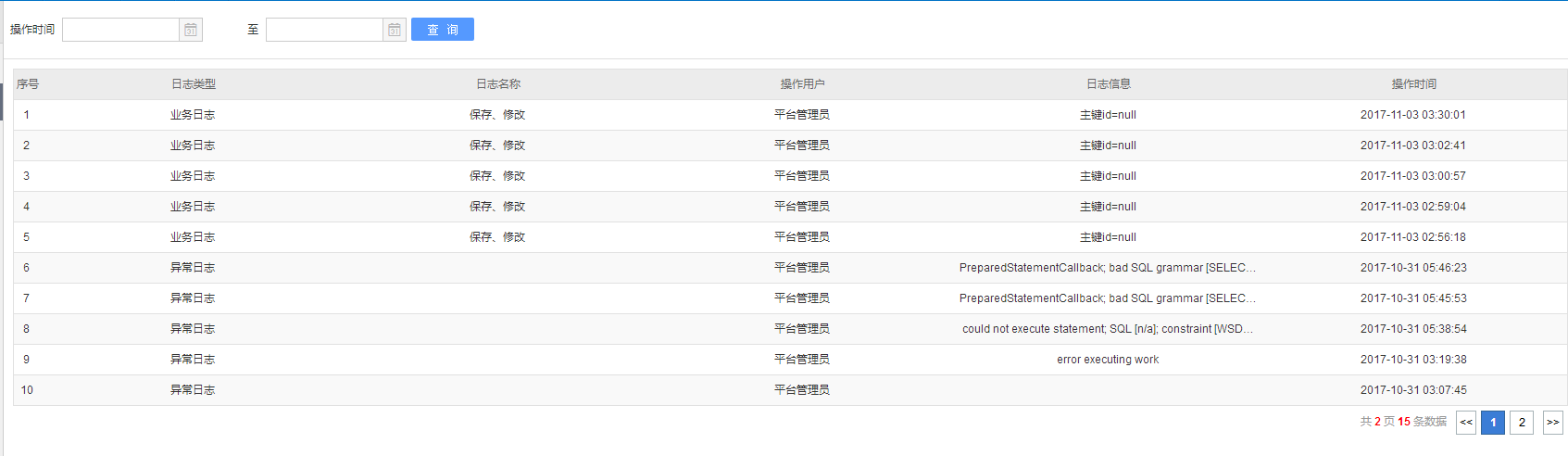


如上图，在controller的保存方法上加入了业务日志的注解，执行完saveConfig的逻辑后，会另起线程保存一条日志名称是“保存、修改”的业务日志。具体解析生产日志的过程在LogAop这个切面类中实现。当然平台只提供了，最常见的业务日志的记录实现，如果项目中对于业务日志的记录有个性化的需求，可以改造扩展LogAop的实现。

## 页面展示

在平台的后台中，实现登录日志和业务日志的两个列表展示页面并提供了简单的查询功能。





大家可以根据项目需求调整实现。