一、命名规范

1. 文件命名要以英文（不会拼的有道，谷歌之类的查查都行）为准，尽量简单明了，但是不要缩写，原则中文全拼也不能出现

2. 类名首字母大写，方法名和属性名首字母小写，统一遵循驼峰标识，常量统一大写用下划线进行分隔，并用 final 进行修饰

3. 接口类中的方法和属性不要加任何访问权限修饰符，尽量保持代码整洁；抽象类的属性或者方法如果只有子类能访问要用protected修饰

4. 新增的业务领域模型如没有特殊需求，要继承基类，并尽量调用基类现有的方法或者属性

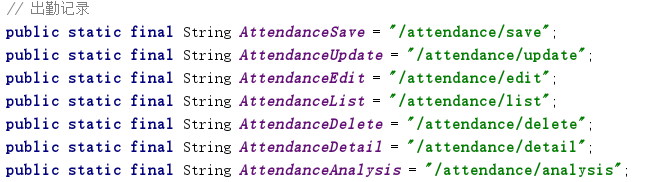
4-1. Entity（基类 BaseAssignedEntity id自定义， id自动递增）, 字段如果只有几个固定的值，尽量采用枚举类型；数据库建表统一小写，并根据驼峰标识用下划线分隔，如：JAVA字段名：private String carId，则数据库字段名：car\_id

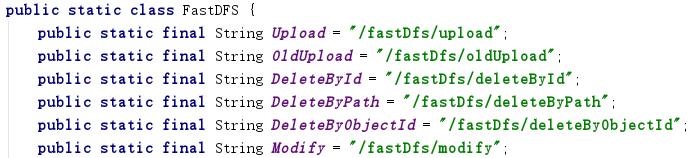
\* 字段注释必须加！！！字段注释必须加！！！字段注释必须加！！！



4-2. Service （基类 BaseJpaService），常用方法名定义的时候获取单个对象用get或者find，多个对象用list，分页用page，保存用save，更新用update或modify，删除用delete或remove做前缀

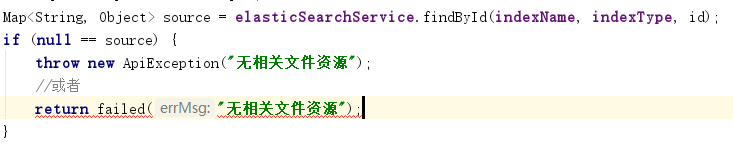
4-3. Controller（基类 EnvelopRestEndPoint），常用方法名定义的时候获取单个对象用get或者find，多个对象用list，分页用page，保存用save，更新用update或modify，删除用delete或remove做前缀。URL映射地址定义参考以下



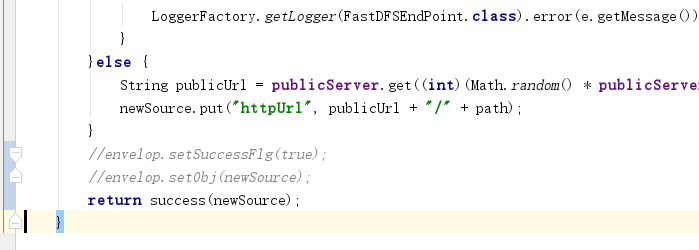


4-3-1. 返回结果处理，如果以Envelop为返回结果，可选择一下方式进行处理

（1）请求失败示例（以下代码的返回结果一样）尽量使用第二种，如果返回的结果要同时指定HTTP状态码，可使用第一种，详见具体带参构造函数



（2）请求成功示例

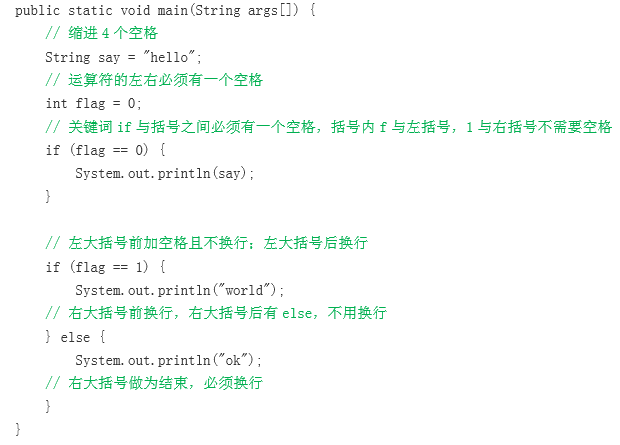


以上代码为简单示例，更多接口及方法请查看公共部分代码

4-4. Dao （目前项目数据访问层基本采用JPA的形式继承 PagingAndSortingRepository 即可），另外目前代码里面数据访问层的类名有用Dao结尾的有用Repository结尾的，个人建议用Dao结尾。还有开头不必加 X

二、格式规范

1. 参考以下



2. 调用方法的时候参数之间要有空格 service.method("a", "b", "c");

三、控制语句

1. 控制语句的大括号必须携带，不要用 if (condition) execute; 的形式;

2. if 语句能解决的问题，不必使用else， 如 if (condition) { ... return obj; }

3. switch 语句要用break或者return防止穿透，并以default结尾

四、注释规范

1. 非常用的新增方法名必须要加注释，注释使用 /\*\* content \*/ 的形式

2. 方法内的单行注释使用//，多行注释使用 /\* content \*/

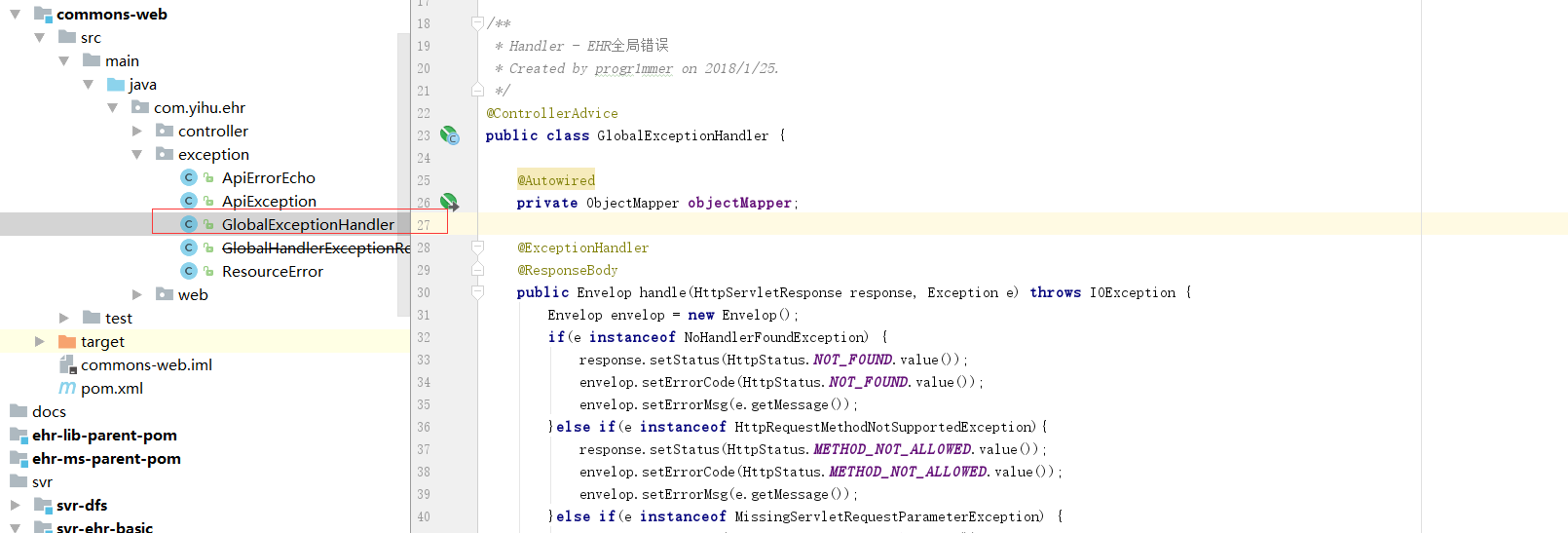
3. 修改原有代码的时候要加修改时间，修改者的相关信息注释

4. 特别需要注意的是，拷贝的代码，如果与现在实现的业务逻辑不相符，原有的注释、(方法名、参数名)一定要修改

五、异常处理

1. 可控的运行时异常一定要尽量避免，如空指针异常，类型转换异常

2. 如果不需要进行特殊处理的异常信息，可将其抛出，项目里面目前在Controller层有定义了一个全局错误处理器



该处理器在发生异常的时候会将错误信息封装成Envelop对象，并返回相应的Http请求状态码

3. 如果进行 try catch 的话，一定要将异常的堆栈信息打印出来，e.getMessage()能获取到的信息太少，不好定位错误

4. 处理示例

（补充：之前项目在Controller有做try..catch的处理但是并没有业务相关的代码，后续大家在开发的时候有看到相关的代码请顺手改一下）

可直接抛出示例（不必进行try...catch，有看到相关代码请顺手改一下）



不可直接抛出示例

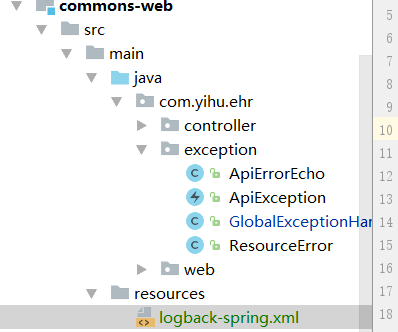


5. 如果不可避免的在Controller进行多表更新或者插入操作的时候要使用注解 @Transactional(rollbackFor = Exception.**class**, propagation = Propagation.***REQUIRED***)，进行事务控制，防止发生异常的时候导致数据不一致的情况发生

六、日志处理

1. EHR平台的日志统一用logback进行处理

2. 接口访问的日志已提供统一的日志配置



如果有其他业务日志需要日志记录可在项目底下新增日志配置覆盖该配置（参考svr-pack-resolve项目底下的logback-spring.xml），项目启动的时候如果是开发环境，会在项目的磁盘根路径创建相关的日志文件夹，如果是生产环境则会在 / 路径底下创建相关的文件夹

3. 大量输出无效日志，不利于系统性能提升，也不利于快速定位错误，记录日志的时候请思考：这些日志是否会有人看？看到这条日志能用来做什么？能不能给问题的排查带来好处？

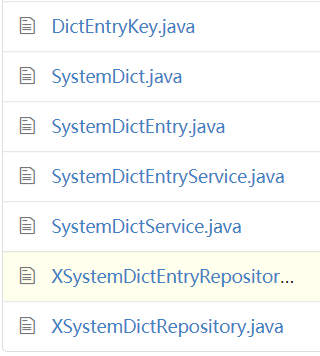
七、其他

1. 操作大量字符串拼接的时候用StringBuilder的append()方法， 不要用 str += str1;

2. 不要在一个循环里面调用参数格相同的Http请求，能一个接口一次请求解决的功能，尽量一次请求一次解决。

3. 目前有些后台必须要有的参数，前端有时为了渲染页面，会要求APP的Controller参数设置为非必传项，这时APP如果没有检查参数会导致后 台报错，大量无意义的日志会增加排查问题的难度，故在此要求有类似情况的时候，一定要在接口调用的时候检查参数是否合法

4. 各个领域模型类，如Entity、Dao、Service、Controller不能放在同一个包下（图1.1），几个领域模型就要划分几个包



5. 常用的工具类统一使用 commoms-util 模块底下的

6. （建议）如果微服务返回的数据前端可以直接展示的话，不需要进行模型转换。(RestModel)

7. 删除数据的时候一定要检查关联表是否存在相关数据，如果存在要一并删除，避免数据库存在垃圾数据

8. 前端错误信息处理（待讨论）