1. 目的

使用统一编码约定集的主要原因，是使应用程序的结构和编码风格标准化，以便于阅读和理解这段编码。好的编码约定可使源代码严谨、可读性强且意义清楚，与其它语言约定相一致，并且尽可能的直观。

一组通用目的的编码约定应该定义完成上述目的所必需的、能让程序员自由地创建程序逻辑和功能流程的最小的要求。编码约定的目的是使程序易于阅读和理解，而不是用过份的约束和绝对的限制来束缚程序员本身的创造性。

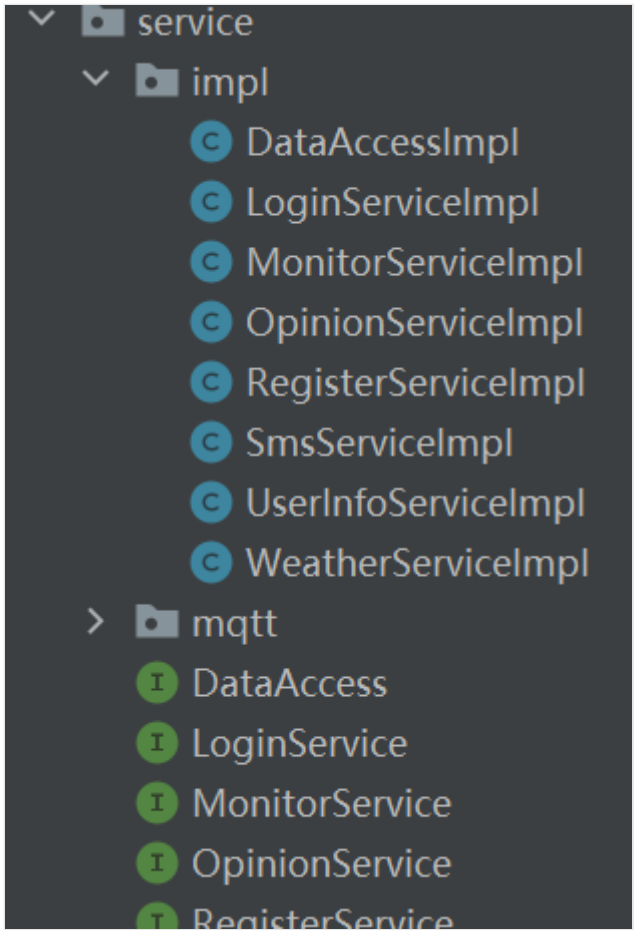
2. 命名规范

2.1. 类的命名

类的名字必须由大写字母开头而单词中的其他字母均为小写，例如Circle；如果类名称由多个单词组成，则每个单词的首字母均应为大写例如TestPage；如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如XMLExample，还有一点命名技巧就是由于类是设计用来代表对象的，所以在命名类时应尽量选择名词。

1. 服务的命名：将接口和实现类分开，实现类为接口名+impl
2. 功能主界面的弹窗类命名：以描述功能 +后缀 Dialog结尾
3. 实体类即用SQL查询需要填入的类命名： 以功能描述 + 后缀（POJO/Bean/Entity）结尾
4. 使用TableViewer进行解析的的类命名： 以功能描述 +后缀LabelProvider 结尾
5. 用于主界面Form测试的以： 功能描述主界面+后缀Run结尾
6. 用于DS表的字段和表名存放：使用I 开头 + 功能描述 命名
7. 模块化的界面（主界面各个模块不在一个类中）使用 ： 功能描述 +Composite（即模块继承或者使用的组件类型）结尾
8. 操作UI界面的一些类：要求封装在以 主界面+ Util的类名中

下图是命名的java类的例子（做到看到类名知道这个类的作用）：



2.2. 其他命名

如：包、接口、文件编码、方法、变量、常量、参数、数据变量、命名要求、命名禁止命名规范详见文档《HT-OC-JAVA开发规范说明书》。

3. 排版规范

3.1. IDEA 设置

默认 IDEA 设置

4. 日志输出

4.1. Slf4j 的使用

常用的日志输出级别有error、info、debug，输出级别error > info > debug。当输出级别为debug时, error、info、debug级别的日志都会打印；当输出级别为info时, error、info级别的日志会打印; 当输出级别为error时, 只有error级别的日志会打印。

系统上线初期日志会设置为debug级别，以便跟踪非常详细的日志，但是日志的量会非常大，占用大量磁盘空间；上线稳定后，一般会设置为info级别，跟踪一般的日志信息。

日志输出时按如下规则划分：

1. 以下日志使用log.error输出
    - 后台捕获Exception后，需要提示的错误信息
2. 以下日志使用log.info输出
    - JDBC调用的SQL语句

    - 程序执行过程中，必要的警示信息
3. 以下日志使用logdebug输出
    - 输出调试信息，如循环中打印的日志信息

4.2. Log日志管理

日志文件标准的方式是按日期生成，由系统管理员定时做清理。

4.3. 异常处理规范

重新抛出的异常必须保留原来的异常，即throw new NewException(“message”, e); 而不能写成throw new NewException(“message”)，更不能不继续往上层抛出异常。

针对重要的可捕获的业务相关异常，需创建异常处理类，在方法中捕获到异常后，反馈到用户界面上，提示用户。

5. 国际化

5.1. 资源包命名

- 对于基础资源包，通常使用以下命名约定：`basename_language_COUNTRY.properties`，其中：
    - `basename` 表示资源包的基本名称，通常与应用程序或模块的名称相关。
    - `language` 表示语言代码，采用小写的两个字母表示（如en、zh等）。
    - `COUNTRY` 表示国家/地区代码，采用大写的两个字母表示（如US、CN等）。
- 例如，一个针对英语（美国）的基础资源包可以命名为：`messages_en_US.properties`。
- 对于其他语言的资源包，根据相应的语言和国家/地区代码进行命名（如 `messages_fr_FR.properties`）。

5.2. 类命名

- 国际化类通常是用于加载和访问资源包中的文本消息的辅助类。
- 类名通常与资源包相关联，可以使用类似的命名约定：`<Basename>Messages`。
- 例如，如果基础资源包的名称是`messages`，则国际化类可以命名为`Messages`，如 `Messages.java`。

5.3. 键命名：

- 资源包中的文本消息通常使用键值对的形式存储，其中键（Key）用于唯一标识消息。
- 键的命名应该是具有描述性的，可以根据具体的应用场景和消息内容进行命名。
- 建议使用小写字母、下划线和数字的组合，避免使用特殊字符和空格。
- 例如，对于一个欢迎消息的键可以命名为`welcome_message`。

6. 注释说明

详见文档《HT-OC-JAVA开发规范说明书》。

7. 反面示例

7.1. 日志输出不符合规范要求

错误示例1（直接使用print打印日志）

```
System.out.println("log: "+sql);
```

正确示例1（使用 log 输出日志）

```
log.info("mqtt 客户端已订阅， 主题："+topic);
```

错误示例2：数据还未删除成功就记录操作日志

```
try {
    userMapper.register(phoneNumber,pwd);
    log.info("用户注册成功，phone："+phoneNumber);
}catch(Exception e){
    log.error(e.getMessage());
    throw new RuntimeException("用户注册失败");
}
```

正确示例2：应在数据删除成功后记录操作日志

```
try {
    userMapper.register(phoneNumber,pwd);
}catch(Exception e){
    log.error(e.getMessage());
    throw new RuntimeException("用户注册失败");
}
log.info("用户注册成功，phone："+phoneNumber);
```