CodeReview

袁剑

最主要的一些问题

- 注释：不写注释或者注释不规范
- Git提交记录：随意且不规范
- 代码可读性

好的代码需要

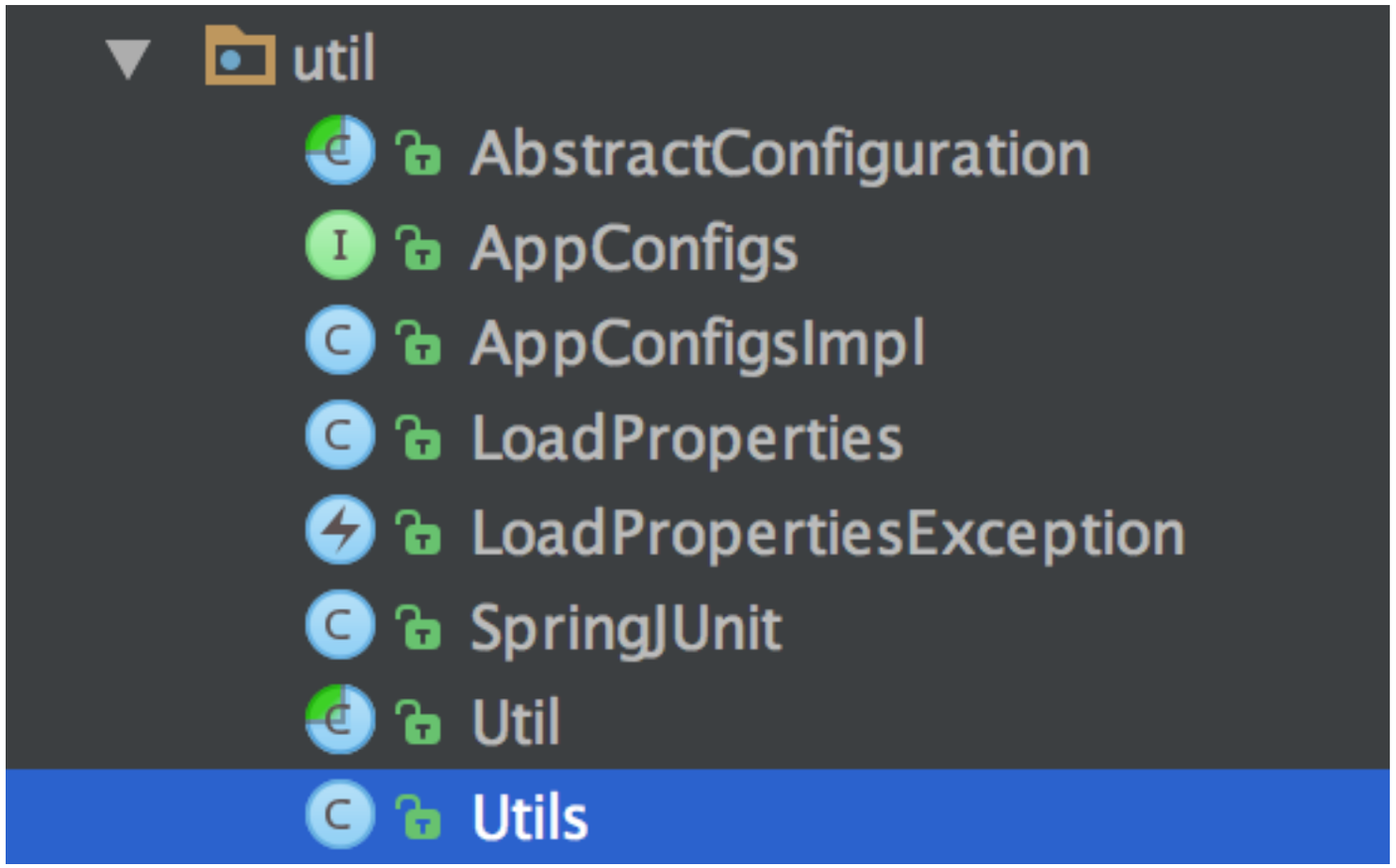
- 可阅读性
- 可维护性
- 遵守规范
- 逻辑清晰+良好的变量（方法）命名

一些基本原则

- Kiss
  - keep it simple, stupid
- Dry
  - Don’t repeat your self
- 沟通
  - 不懂没关系，不问就是你不对了

问题1：类名重复

util目录下有Util和Utils两个类让人混淆且无法理解



问题2：换行

```
AddressAdminProto.UpdateAddressRequest request = AddressAdminProto.UpdateAddressRequest.newBuilder()
        .setId(query.getId()).setNameCN(query.getNameCN()).setNameEN(query.getNameEN())
        .setNamePY(query.getNamePY()).setAlias(query.getAlias()).setSortOrder(query.getSortOrder())
        .setUpdateUser(0).setLevel(query.getLevel()).build();
```

不注意换行的代码会造成阅读障碍，对比上下两段代码

```
AddressAdminProto.UpdateAddressRequest requestRefactor = AddressAdminProto.UpdateAddressRequest.newBuilder()
        .setId(query.getId())
        .setNameCN(query.getNameCN())
        .setNameEN(query.getNameEN())
        .setNamePY(query.getNamePY())
        .setAlias(query.getAlias())
        .setSortOrder(query.getSortOrder())
        .setUpdateUser(0)
        .setLevel(query.getLevel())
        .build();
```

问题3：提交记录

Git的提交记录无法直观的看到修改了什么内容

Git提交记录规范

什么时候提交Git：完成一个功能/修改完1个BUG

提交时写什么：告诉别人你改了什么内容

- 简单的提交记录，如改动很简单将来也不会变化的用一行进行说明，如：解决生成的DTO代码中重复引用java.util.List的问题

- 复杂的提交记录，在标题用简短文字总结，用正文解释是什么，如：

  聚合订单查询服务修改

  1、增加接口：GetOrder
  2、CreateOrder增加参数OrderId
  ……

写好 Git Commit 信息的 7 个建议：http://blog.jobbole.com/92713/

问题4：长方法

- 可读性及可维护性差

```java
@Override
public AuthFrontProto.SignInResponse signIn(AuthFrontProto.SignInRequest request) {
    String requestId = request.getRequestId();
    String cacherequestid = Cache.get(String.class, UserConstant.CACHED_KEY_SIGNIN, request.getMobile());
    logger.debug("传进来的requestid 为 :"+requestId +"     缓存中的requestid 为： "+cacherequestid);
    if (!requestId.equals(cacherequestid)) {
        //如果没有requestid已经过期或不存在，验证不通过
        throw new FaultException(UserConstant.EXPIRE_OR_NOTEXIST,"requestid已经过期过不存在");
    }
    //验证短信验证码是否正确需要调用短信服务去验证
    if (!captchaService.validate(requestId, request.getChkCode())) {
        //如果验证码已经过期、输入错误或不存在，验证不通过
        throw new FaultException(UserConstant.EXPIRE_OR_NOTEXIST,"验证码错误");
    }
    //判断是否已经注册，先从缓存中获取，如果没有再从数据库中获取
    Long userid = Cache.get(Long.class,UserConstant.CACHED_KEY_MOBILE_USERID , request.getMobile());
    if (userid == null || userid == 0){
        userid = this.AuthFrontBiz.getUserIdFromMap(request.getMobile(), UserType.FRONT_USER.value());
    }
    if (userid == null || userid == 0) {
        //没有注册 走注册流程
        userid =idService.getId(UserConstant.ID_CATEGORY_USER); ////生成userid
        logger.debug("获取的用户userid =====   :"+userid);
        AuthFrontBiz.save(AuthFrontMapper.of(userid, request.getMobile()));
        userFrontService.addUser(AuthFrontMapper.ofadd(userid, request.getMobile()));
        sendNSQ(SolrConstants.NSQTOPIC_FRONT,AuthFrontMapper.ofnsq(userid, request.getMobile()));
    }
    //把用户mobile与userid映射关系存入缓存（没有永久缓存）
    Cache.set(userid, UserConstant.CACHED_KEY_MOBILE_USERID, request.getMobile());

    //已经注册 直接生产token 返回
    String token = Guid.get();   //生产token ， 需要调用生成token的服务
    logger.debug("获取的token =====   :"+token);
    //设置登录状态
    setFirstLoginStatus(userid, request.getMobile(), token, request.getDeviceToken());

    //记录经纬度，deviceToken，ip，并处理用户devictToken表，此逻辑在Task中实现。
    sendNSQ(UserHandlerConstant.NSQTOPIC_FRONT_LOGINLOG, AuthFrontMapper.of(userid, request.getMobile(), requ
    return AuthFrontProto.SignInResponse.newBuilder().setResult(token).build();
}
```

问题4：长方法-重构1

```
106     public AuthFrontProto.SignInResponse signInRefactor(AuthFrontProto.SignInRequest request) {
107         String requestId = request.getRequestId();
108         String checkCode = request.getChkCode();
109         String mobile = request.getMobile();
110         //
111         logger.debug("signIn: requestId:{}, checkCode:{}, mobile:{}", requestId, checkCode, mobile );
112         //
113         if ( requestId == null || requestId.isEmpty() ) {
114             throw new FaultException(UserConstant.REQUEST_ID_EMPTY,"requestid不合法");
115         }
116         if ( checkCode == null || checkCode.isEmpty() ) {
117             throw new FaultException(UserConstant.CHECK_CODE_EMPTY,"checkCode不合法");
118         }
119         if ( mobile == null || mobile.isEmpty() ) {
120             throw new FaultException(UserConstant.MOBILE_EMPTY,"mobile不合法");
121         }
122         //检查请求是否合法
123         boolean valid = checkRequest( requestId, checkCode, mobile );
124         if ( !valid ) {
125             throw new FaultException(UserConstant.EXPIRE_OR_NOTEXIST,"requestid已经过期或验证码错误");
126         }
127         //判断是否已经注册,没有则注册
128         long userId = isRegisterUser( mobile );
129         if ( userId == 0 ) {
130             userId = registerUser( mobile );
131         }
132         //
133         String token = generateToken( userId, mobile, request.getDeviceToken() );
134         //登录后的一些异步处理
135         afterSignIn( userId, mobile, request );
136         return AuthFrontProto.SignInResponse.newBuilder().setResult(token).build();
137     }
```

问题4：长方法-重构2

```java
/**
 * 检查登录请求是否合法
 * @param requestId
 * @param checkCode
 * @return boolean
 */
private boolean checkRequest( String requestId, String checkCode, String mobile ) {
    //如果没有requestid已经过期或不存在，验证不通过
    String cacheRequestId = Cache.get(String.class, UserConstant.CACHED_KEY_SIGNIN, mobile );
    if (!requestId.equals(cacheRequestId)) {
        return false;
    }
    //如果验证码已经过期、输入错误或不存在，验证不通过
    if (!captchaService.validate(requestId, checkCode )) {
        return false;
    }
    //
    return true;
}
```

```java
/**
 * 根据手机号判断是否是用户
 * @param mobile
 * @return boolean
 */
private long isRegisterUser( String mobile ) {
    Long userId = Cache.get(Long.class,UserConstant.CACHED_KEY_MOBILE_USERID, mobile );
    if ( userId == null || userId == 0){
        //这里以后应该还会判断用户状态
        userId = this.AuthFrontBiz.getUserIdFromMap(mobile, UserType.FRONT_USER.value());
    }
    //
    return userId;
}
```

```java
/**
 * 注册
 * @param mobile
 * @return boolean
 */
private long registerUser( String mobile ) {
    //TODO:事务性保证
    long userId = idService.getId( UserConstant.ID_CATEGORY_USER );
    AuthFrontBiz.save(AuthFrontMapper.of( userId, mobile ));
    userFrontService.addUser(AuthFrontMapper.ofadd( userId, mobile ));
    sendNSQ(SolrConstants.NSQTOPIC_FRONT,AuthFrontMapper.ofnsq( userId, mobile));
    //把用户mobile与userid映射关系存入缓存（没有永久缓存）
    Cache.set( userId, UserConstant.CACHED_KEY_MOBILE_USERID, mobile);
    return userId;
```

```java
/**
 * 产生token
 * @param userId
 * @param mobile
 * @return String
 */
private String generateToken( long userId, String mobile, String deviceToken ) {
    //已经注册 直接生产token 返回
    String token = Guid.get();   //生产token ， 需要调用生成token的服务
    logger.debug("获取的token =====   :"+token);
    //设置登录状态
    setFirstLoginStatus( userId, mobile, token, deviceToken );
    return token;
}
```

```java
/**
 * 设置登录状态
 * @param userId
 * @return
 */
private void afterSignIn( long userId, String mobile, AuthFrontProto.SignInRequest request ) {
    //记录经纬度，deviceToken，ip，并处理用户devictToken表，此逻辑在Task中实现。
    sendNSQ(UserHandlerConstant.NSQTOPIC_FRONT_LOGINLOG, AuthFrontMapper.of(userId, request.getM
}
```