### 前端

页面结构以php 或者 html为基础, (不用jsp等, 这样后端实现的代码可以与页面分离，同时，php或者html修改后可直接覆盖，不需要重启), 所有数据都以ajax请求来获取，vagrant搭建本地开发环境，实现前端和后端的解耦，具体技术如下：

jquery,

require.js(<http://www.ruanyifeng.com/blog/2012/10/javascript_module.html>, <http://www.requirejs.cn/> 模块化编程),

art-template (<http://aui.github.io/artTemplate/> javascript 模板引擎),

bootstrap(<http://www.bootcss.com/> ),

vagrant, virtual box (<http://blog.csdn.net/jillliang/article/details/8251242>)

gulp(自动化构建工具，<http://www.techug.com/gulp> )

less or sass(CSS的扩展，让 CSS 语言更强大、优雅 http://www.w3cplus.com/sassguide/)

可选后端模版(<http://www.gbtags.com/gb/share/3309.htm> )

### 后端

以现有技术为基础，包括(spring mvc , spring, mybatis, zookeeper(curator), commons-pool, thrift, redis, quartz, guava, aviator, jackson)， 同时以spring-test为测试case

### 版本控制

git分支管理(<http://blog.jobbole.com/13916/>)

### 文档

api-doc(<http://apidocjs.com/>) -- 基于注解的方式建立api文档，同时也方便建立版本控制，不像doc文档不方便对比

### 代码规范

Google前端代码规范(<http://www.cnblogs.com/2050/archive/2012/04/26/2470947.html> )

Google后端代码规范(<http://www.cnblogs.com/lanxuezaipiao/p/3534447.html> )

建议：

所有controller需要给出请求方式，是get，还是post，返回对象统一为Response

所有的频率限制条件都在filter中完成,而不是放入各个方法中

所有的测试用例都需要放在 src/test/java下，不能在java类中写一个main方法

所有的返回异常信息都需要从resouces.properties中取，统一管理

分页传参以及后端model统一

Thrift定义的model只在srvHandler类中用，到了处理方法函数就转换为不需要序列化的model，以释放相关rpc model class的堆栈内存

缓存基类提取到Icache中，并在无数据的时候实现callback去加载数据，不需要每个地方都先从缓存里取然后判断是否为空，然后在反序列化为对象，最后还需序列化并存储这一系列动作

Controller层获取用户信息，不需要每次filter都从redis获取用户信息，并放入request中，可以在controller层真正获取的时候才去获取，同时放入thread local中

所有的redis key变量包括全局锁，前缀需统一放入变量文件中，方便key维护，同时提供相应的model，包含key和timeout参数以及是否持久

### 代码审查

建议每次合并代码之前需负责人审查代码，看是否符合代码规范，同时看是否有隐藏的bug