**《筹Mi》代码规范文档**

编写人：修峻青

**对于本文档予以规定的命名方式应按照命名方式进行命名，本文档未予以规定的命名方式可由命名者进行规定，并更新文档向其他开发者予以通告执行**

1. **目录结构**

筹Mi项目以com.cn包为根包，下设common包和project包。

common包用于放置公共类，下设controller，util，interceptor和entity包，分别放置公共的控制器，工具类，拦截器类和实体类。

project类用于放置项目模块类，存储项目的各个模块包，模块命名在附表中进行查询，在模块包下设置功能包，包名采用英文全拼自命名的方式。功能包下设置controller，service，repository，entity四个包，分别用于放置功能所需的控制器，业务逻辑类，持久层类和实体类。

JSP页面的目录以webapp为根目录，下设jsp，js，image，css和WEB-INF文件夹，jsp文件夹用于存放JSP页面文件，js文件夹用于存放js文件及其支持文件，image文件夹用于存放网页所需的图片，css文件夹用于存放项目所应用的外部css文件

WEB-INF文件夹用于存放配置文件，存放有web.xml文件和conf文件夹，web.xml配置框架的核心入口类，conf文件夹放置spring的配置文件和数据库及其连接池的配置文件。

1. **文件命名规范**
2. **Java类命名规范**

筹Mi项目的类名基本规范为以对用功能的英文全拼的大驼峰命名法进行命名。为了区分不同用途的类，请按照类的用途和以下规范加前缀或后缀。

控制器类后缀Controller，类名为\*Controller.java

业务逻辑类后缀Server,类名为\*Server.java

持久层类后缀Repository，类名为\*Repository.java

工具类后缀Util，类名为\*Util.java

拦截器类后缀Interceptor，类名为\*Interceptor.java

实体类后缀Entity，类名为\*Entity.java

接口类在以上的基础上前缀I，类名为I\*.java

1. **JSP页面命名规范**

JSP页面以页面内容的英文全拼的大驼峰命名法进行命名，要求能够直观的显示页面的主要内容

1. **包命名规范**

包命名除约定外，其他的均以所对应的功能英文全拼全小写命名，约定名如下

控制器包controller

业务服务类包server

持久层包repository

实体类包entity

工具类包util

拦截器包interceptor

1. **JS，CSS文件和图片文件命名规范**

JS文件和CSS文件命名以其所属的JSP页面为名进行命名，对于全局的CSS和JS文件以common为名进行命名。图片除约定俗成的命名外均以其具体用途的汉语拼音进行命名

1. **文档命名规范**

文档命名以“模块名+功能描述”为名进行命名，内部格式参照设计文档

1. **设计图命名规范**

设计图以其表现的“功能或模块+图种类”为名命名，格式统一为jpg

1. **代码规范**
2. **Java代码规范**

Java代码规范应以良好的阅读性为前提。

1. **对象命名，**所有的对象名均以所对应名的首字母小写来命名，不得随意删减。
2. **变量命名**，对于有实际意义的变量以其值意义的英文全拼以大驼峰命名法进行命名，所有的变量除特殊需要外访问权限都应设置为私有。对于boolen类型的数据应前缀is（is无须大写）。对于计数器变量除了约定俗成的m,n,i,j,k外应后缀Counter，标识是计数器变量。对于某些特殊的无意义变量，请不要采用m,n,i,j,k来命名，以免引起混淆
3. **常量命名，**均以其功能或内容的英文全拼全大写命名，单词之间以下划线隔开，除特殊外，建议类型均为String类型。数据访问权限应设置为public static final
4. **方法命名，**除特殊的全局方法以外Controller层和Service层的方法均应后缀其层名，命名以方法功能的英文全拼大驼峰命名法予以命名，函数的参数以英文小写，能够充分表达参数的意义为要求，不做其它单独限制。另外构造方法理论上应当对异常进行处理，但如果确实无法处理的异常，请返回相应的返回码。
5. **注释规范，**注释分为三类，对于代码执行节点以//的方式进行注释，全局变量以/\*\*/的格式进行注释，对于方法和类的注释以/\*\* \*/的方式自动生成注释。为了提升代码的可读性并提高纠错效率，代码的注释比例至少达到文件代码总量的20%，对于关键节点，循环节点，判断节点应进行详细的注释。
6. **缩进换行规范，**Java代码缩进以便于阅读为基本前提，应尽量避免过多的缩进层级。换行建议以60-80个字符换一次行，对于过长的代码应予以换行以提高可读性，但换行不应分割一个连续的字符串，如对象名，方法名，参数名等，需保持代码阅读单元的连续性。
7. **逻辑处理建议，**编写业务逻辑时应注意避免以下的逻辑。可能无法结束的循环，对于这类循环应设置强制跳出条件。层级过多的分支层级，如果分支层级过多会显著的提升系统开销，降低系统执行效率，如果不可避免应当使用多条件联合判断，以降低业务逻辑的复杂程度。避免可能造成的返回值缺失方法，使用分支时务必要注意可能造成的返回值缺失问题。对于非数据库修改操作，建议以高效为前提，必要情况下，例如Server层没有必要的逻辑处理时，可不建立Service层方法
8. **方法设置，**方法的设置以注重性能同时兼顾便于阅读和保持连贯性为基本要求
9. **JSP代码规范**
10. JSP页面的Title以页面的主要功能的中文描述命名。
11. <script>和<style>标签均应设置type属性，并且采用双标签方式。<head>标签内的其他标签应视情况使用。
12. 页面的全部路径均应采用绝对路径，其中项目目录的根路径可以从Session当中使用EL表达式直接获取，获取方式为${sessionScope.ctx}
13. 页面应采用合理的缩进以提高代码的可读性，对于成对的标签应处于同一缩进位置，对于单例标签应以/结尾，保持标签代码的完整性，并对于其中的重要<div>标签，表单标签应在标签开始的位置进行注释。
14. 对于可以多次复用的CSS和JS建议采用外部式，对于缺少复用价值的CSS和JS外部式的，可以采用内部式。涉及数据提交的JS，不论异步还是同步都应使用内部式，以便于修改和调整。
15. JS的业务逻辑建议与Java业务逻辑基本相同，前端涉及到用户友好性，过于复杂的JS处理逻辑会延迟页面的加载和执行速度，应当在一定程度上予以优化，对于数据加载类的JS不建议采用同步处理，推荐使用异步处理以减少等待时间；对于需要输入隐私信息的数据提交类请求，不应使用异步使之可以被重复。
16. CSS属性的使用需要注意浏览器的兼容性，本次我们采用的浏览器为Chorme与IE11，因此如有需要，请注意准备多套适配于不同浏览器的CSS属性。
17. **JS和CSS代码规范**
18. CSS和JS应在保证代码的完整性的基础上提高代码的复用程度，不应为了复用CSS属性和JS代码而拆散本应连贯的重要部分。
19. CSS和JS不建议使用内联式，某些情况下为了解决属性的优先级问题也可以使用，但应当在注释中表明。
20. 页面或页面组的外部式专有CSS文件的命名以相对应的页面或页面组的名进行命名，并放入相对应的分级目录当中。
21. **代码建议，此段无强制性要求**
22. **Java代码建议**

之前我们提到代码结构的精简性和高效性，为了实现这一点，我在这里有一些小的建议，可以提高代码的可读性。首先，对于同功能的不同逻辑，建议采取分离模式。比如登录功能，我们把登录分做了普通登录，单点登录，手机验证码登录三种，那么我们就针对于登录的接口写三个实现类，并将一些需要复用的代码抽象到一个抽象类当中，由抽象类继承接口，当使用的时候。并根据需要实例化相应的实现类对象。这样能够使得你的不同处理逻辑保持一定的独立性，并且保证你的代码不会碎片化并且也保证了代码的复用率。

另外一点是当处理以上问题，即需要在执行中使用多态来决定实例化哪一个实现类对象，我们建议将判断逻辑也放置到相应的实现类当中，此时我们的代码分布是这样的：接口提供了公共函数代码，也就是判断函数和入口函数的函数签名，然后由一个抽象类继承接口，将一些能够复用的公共代码做成public或者protected方法，提高代码的利用率，但对于公共函数不做实现。最后是不同的实现类，在实现类当中我们实现其各自的公共函数，将判断函数的判断条件加以设置实现，还有入口类我们基于从抽象类当中继承而来的方法，予以调用并根据各自逻辑的特点再加以补充，实现功能。