# 计划

## 进度计划

### 人力资源计划

### 人员构成

### 角色，任务，数量，时间

#### 培训、考试

# 配置

文档管理：

原始资料、需求、设计、测试、验收文档、用户手册的目录位置、命名约定、锁定规则

使用wiki?

使用缺陷跟踪系统？

# 沟通

设计开发规范的更新与发布

# 需求

业务描述

决策人员的愿景

表单解释

# 设计

原则：既要避免设计不足，不能指导开发，又要避免过度设计、导致工期过程。

## 总体设计

## 命名方案

许多地方都涉及到命名的问题，如：jsp，Controller，url。总体上来讲，有两种方案可供选择：拼音和英文。应使用一致的方案进行命名。

1. 拼音

一般情况下使用简拼，如遇到重复的情况，可以使用全拼形式，比如婚姻与怀孕的简拼都是hy，如果婚姻使用了hy，则怀孕可以使用hun\_yin或HunYin。优点：与中文有简单的对应关系，命名过程比较轻松；缺点：在使用英文比较简单的情况下，没有英文直观，不容易体现复数的概念。

1. 英文

如果使用英文，对于较长或较为难以翻译的词，不追求能够准确的体现中文的原意，只要能够容易与对应的中文建立关联即可，比如，对于城市低保对象的子类型比照低保标准发生活费对象来说，可以使用compare来代表，无论多复杂的中文短语，尽量不要超过3个单词

## 数据库设计方法

数据库设计应当整体进行，当核心业务的数据库设计完成后，再进入到界面及业务逻辑设计阶段

#### 表

1. 表的命名规范
2. 表注释
3. 主键的命名规范
4. “外键”的命名规范
5. 字段的命名规范：日期、时间等
6. 数据类型的使用约定：日期、时间
7. 默认值
8. 字段注释，如何处理字典
9. 字段命名的一致性

#### 视图

#### 函数

#### 存储过程

## 功能设计

功能设计，包括界面及业务逻辑设计。

功能设计使用word格式，内容分为一般规定、基本操作和功能模块三部分。

### 一般原则

一般原则部分给出本设计对一些具有普遍性的问题的处理原则，例如：

* 系统分层
* 命名规范：Jsp, url, Domain, Dao(Impl), Service(Impl), Controller的命名规范
* 对于录入字段，给出合适的默认值
* 编辑/查看页面，如果最后一行有空列，则使用空的<th /><td / >补充，而不是使用colspan的方式处理；对于只读的字段，直接显示文字，不使用控件
* url的命名规范：多个单词以下划线（\_）连接，ajax请求的url以ajax\_开头
* Controller及其方法的命名，Service(Impl)、Domain的命名
* jsp文件的命名：模块名\_页面描述.jsp，如果需要使用多个单词，则以下划线连接，还是采用java变量方式命名，同一模块的多个文件放在相同的模块目录下
* 对话框：使用jquery控件代替浏览器自带的alert、confirm的功能
* Service、Dao是否使用接口
* 在Service方法上使用@Transactional还是在Service类上使用
* Domain是否使用继承，一对多、多对一、一对一的配置原则
* $.blockUI的使用
* 表单验证过程

### 基本操作

基本操作部分给出查询、新增、修改、删除、查看的通用部分的处理方法，包括url，方法名，基本算法（业务逻辑）。

### 功能模块

功能模块部分依次对系统的各个功能进行设计。页面设计按照页面初始化、用户与界面的各种交互操作分别进行设计。用户与界面的交互包括选择下拉列表中的值、在文本框中输入值、点击按钮/链接等。

对于分页查询功能，需要指出以下信息：

1. 查询条件

标题、条件名、默认值、是否必填、控件、排序（升、降、无）

1. 查询结果

标题、字段、翻译、操作及条件；多个操作之间使用&nbsp;进行分隔；常用的、不危险的操作放在这里，最多不能超过三个，一般包括编辑、查看

### 命名规范

### 一般原则

### 最佳实践

#### Controller

#### Service

#### Domain

### 数据库

# 开发

开发必需执行较为严格的开发规范，项目启动前，项目组成员有必要就开发规范进行统一的学习、确认，在开发的过程中遇到现有规范没有明确的地方，应向架构师提出，架构师提出解决方案后、核心开发人员讨论通过后予以实施

开发规范应当简单明确，易于实施、检查，考虑能够以检查点的形式体现，以便于开发时进行执行

开发过程中，应当根据模块的特殊性程度、复杂程度按照一定的比例比对开发规范进行抽检，应该在计划中指定检查人员并预留相应的时间

Js：jsp私有的js应该放在模块目录下，每个jsp对应一个同名（后缀不同）的js文件，公共的js文件放在js/common目录下，第三方的js文件放在js/lib下，部分公用的js文件放在js/<子目录>中，子目录指出该部分公用js的用途

Css：jsp私有的css文件的命名规则：css/<模块名>.css，公共的css放在css/common目录下，部分公用的css文件放在css/<子目录>下，子目录指出该部分公用css的用途

验证过程

## 基本操作

字典的处理

## 查询

系统提供的分页查询的两种实现方式：

方式一：通过searchForm使用PageRequest方式

深入了解SearchForm

form属性：

key：[字段名]\_[操作类型]\_<数据类型>

value: 字段值

常用操作类型：lk, eq, nm

常用操作类型：string

解析时使用and进行连接

Map<String, Object[]> whereEntries:

key: 条件表达式，

value: 条件值数组

在进行解析时，会将条件表达式用括号括起来，并使用and与其他条件连接起来

pageRequest属性：

pageNo

pageSize

sort

orders：可以包含优先级最高的的排序字段

orderBy

order

orderBy: 优先级最高的排序字段

order

ParamEntry

paramName ：参数（条件字段）名

queryCase: 条件操作符

value: 条件值

WhereEntry

String whereClause: 条件表达式，

List whereParams: 条件值

Criteria

List< ParamEntry> entries

fetchList

List<WhereEntry> whereEntries

firstResult

maxResults

parseForm方法：

解析form中的键值对，保存到ParamEntry中

解析whereEntries

将前两步的结果封装到criteria中

最佳实践：

第一步：在jsp中构造符合SearchForm的命名规则的查询条件

第二步：处理隐含条件

例如：低保户查询的查询结果应当只包含未脱保的低保户，则需要通过SearchForm#addFormEntry(‘isOutOfLowSecurity’, YesOrNo.NO)方法添加一个条件

第三步：默认值

页面初始化时，可以设置默认值：

例如：

If(form.isEmpty()){

form.put(“areaCode”, “220104”);

}

第四步：对可能需要使用简拼来查询的条件，需要判断条件值的首字符是否为字母，如果是，则需要将相应的条件字段换成相应的简拼字段名加上“\_eq”，这样做应该会比使用or的方式快一些

第五步：设置排序

setOrderBy/order和/或addOrder方法增加排序字段

第六步：执行查询

Page page = findPage(searchForm)

第七步：处理结果

Page = DictionaryHelper.decodePage();

decodePage()方法内部调用decodeBean()方法，但是目前decodeBean()方法似乎会将对象类型的属性处理成一个代表对象的字符串，因此建议使用反射的Field.getValue()方法获取原对象的属性值

方式二：使用pageReqeust方式，如果框架通过searchForm的方式做了统一处理，则不能使用该方法。

了解history与redirect

了解常用的service方法

了解detach

了解事务

建议扩展系统提供的queryPage方法，支持简拼查询

## 增加

## 修改

## 查看

## 删除

## 其他

### 普通常量类vs继承常量类vs枚举

# 测试

编写测试用例

准备测试数据

# 实施/试运行

历史数据迁移

# 收尾

1. 从版本库下载工程
2. 导入下载的工程
3. 确定要开发的模块，如：低保户
4. 根据[minz-web/src/main/webapp/html/frame/民政页面整理.xlsx]确定模块的路径，如：[低保户]模块的路径为[/html/help/object/fixed/]
5. 根据模块路径确定模块view.xml文件的路径，如：[低保户]模块的view.xml文件的路径为[minz-web/src/main/webapp/WEB-INF/views/help/object/low\_security/views.xml]
6. 配置模块的view.xml文件

假设业务模块（与菜单一一对应）名为low\_security(低保)，则：

|  |  |  |  |
| --- | --- | --- | --- |
| 操作类型 | definition[name]=/help  /object/low\_security/ | Put-attribute[value]=/WEB-INF  /views/help/object/low\_security/ | Controller方法 |
| 查询 | list | low\_security\_list.jsp | list |
| 增加 | add | low\_security\_add.jsp | add(method=get) |
| 增加保存 | add |  | add(method=post) |
| 修改 | edit | low\_security\_edit.jsp | edit(method=get) |
| 修改保存 | edit |  | edit(method=post) |
| 删除 | remove |  | remove |
| 查看 | view | low\_security\_view.jsp | view |
| 其他操作 | {operate} | low\_security\_{operate}.jsp | operate(method=get) |
| 操作保存 | {operate} |  | operate(method=post) |

1. 根据view.xml文件的配置，将相关jsp从minz-web/src/main/webapp/html/目录复制到minz-web/src/main/webapp/WEB-INF/views/目录，子路径及文件名可根据view.jsp中的配置进行修改
2. 开发Controller

PathVariable,ModelAttribute,Model,SearchForm

1. 开发service

涉及到数据库的写操作的方式请在方法前增加 @Transactional 注解,否则会造成事务不提交的问题。嵌套的方法也需要添加该注解。

1. Jsp二次开发

关于事务：

系统配置不支持在Controller中使用Transactional注解，因此，如果请求的操作超过一个数据库更新语句，则需要放到Service中来实现，然后由Controller来调用

异常处理：

在Service层，如果出现已检查异常（checked exception），要想使事务回滚，则必须捕获该异常，然后抛出MinzRuntimeException,系统会自动转到相应的错误页面；Controller层可将已检查异常捕获后转换成MinzException抛出

使用相同版本的ide（eclipse），源代码在本地文件系统上的路径相同，maven的位置相同，总之，要保证svn中的.project、.classpath等完全一致，以避免各自环境不同导致的频繁冲突

url规范

Jsp命名规范

Controller命名规范

请求方法命名规范

Ajax方法

Service命名规范，service方法命名规范

文件上传、下载规范

常用的快捷键

数据库：

字典类型的字段注释：

<字段名称>：DICT\_TYPE=’<字典名称>’

字典名称：<表名>\_<字段名>

数据库管理制度：

ddl语句只能由专人执行，先在公共数据库中执行，经验证无误后，再在公共数据库备份中执行

公共数据库、个人数据库、公共数据库备份

个人测试在个人数据库中执行，系统测试在公共数据库中执行

**Javascript变量采用匈牙利命名法**

匈牙利命名法语法：

变量名＝类型＋对象描述

|  |  |
| --- | --- |
| **JavaScript变量类型** | **变量命名前缀** |
| Array 数组 | a |
| Float 浮点 | l |
| Function 函数 | f |
| Integer(int) 整型 | n |
| Object 对象 | o |
| Regular Expression 正则 | r |
| String 字符串 | s |
| Boolean | b |

编辑时，字典类型字段的页面显示：

使用文本还是控件？文本。

如果使用控件，则需要使用disable属性，后台更新时要注意先取得原值后再更新；或者将文本型空件设置为readonly，样式设置为disable状态，将select设置为伪disable

url、Jsp命名规范：**使用下划线**还是驼峰命名方式

页面常规提交（相对于ajax方式）后，应该使用blockui阻塞用户操作，如果使用了校验，则应该在校验通过后使用blockui进行页面的阻塞

查看时，应该显示更新日期、用户，创建日期、用户

最佳实践：

Domain:

可以从Domain继承rid, createUserFid, createUser, createTime, updateUserFid, updateUser, updateTime, isNew()，换言之，Domain已经实现了IEntity<String>和UpdateCreateSmart接口

常规操作：

查询：

增加：

设置默认属性

使用entityManager.persist()方法将对象插入到数据库中

修改：

从数据库中获取要查看的记录

使用BeanUtils.merge方法将页面的修改结果复制到从数据库库中获取的记录上

使用entityManager.merge更新从数据库中获取的记录

删除

查看：

从数据库中获取要查看的记录

使用distach方法将记录从持久化上下文中剥离出来

使用decodeSourcebean方法对字典值进行转译，这里没有使用decodeBean方法是因为decodeBean方法会把“复杂类型”的字段干掉

History的使用

界面阻塞的使用方式与时机

小窍门：

同一个jsp文件，如果在minz工程下打开，则没有自动完成功能，如果在minz-web下打开，则有