享洗自助洗衣系统代码规范文档

模块名称	享洗自助洗衣系统设计文档
所属系统	
项目负责人	王子杰
文档提交日期	2017/5/25

修改记录

No	修改后版本号	修改内容简介	修改日期	修改人
1	1.0	技术规范	22/3/2017	王子杰
2	2.0	技术代码规范	2017/5/25	王子杰

Ruby On Rails 代码规范

1 排版

1-1：程序块要采用缩进风格编写，缩进的空格数为4个。
说明：对于由开发工具自动生成的代码可以有不一致。

1-2：相对独立的程序块之间、变量说明之后必须加空行。
示例：

```
def new
    @user = Space.find_by_passport(cookies[:passport])
    @entry_categories = EntryCategory.find(:all, :conditions => ["space_id=?", @user.id])
 end

 def index
    @user = Space.find_by_passport(cookies[:passport])
    @entries = Entry.paginate(:page => params[:page], :per_page => 30,
          :conditions => ["space_id = ? and status = 0", @user.id])
  end
```

1-3：较长的语句（>80字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读。

示例：

```
@entries = Entry.paginate(:page => params[:page], :per_page => 30,

         :conditions => ["space_id = ? and status = 0", @user.id],

         :order => "updated_at desc")
```

1-4：循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分，长表达式要在低优先级操作符处划分新行，操作符放在新行之首。

示例：

```
<% if @album.permission == 0 %>

       <input type="radio" name="album[permission]" value="0" checked>公开

       <input type="radio" name="album[permission]" value="1">隐私

 <% else %>

       <input type="radio" name="album[permission]" value="0">公开
```

```
        <input type="radio" name="album[permission]" value="1" checked>隐私

  <% end %>
```

1-5：不允许把多个短语句写在一行中，即一行只写一条语句。

1-6：参数中的逗号前面不加空格，后面空一格。

示例：

```
@hot_all = Entry.find(:all, :conditions => ["status = 0 and permission !=
          2"], :order => "reading_count desc"
```

1-7：操作符前后各空一格

示例：

```
@entry.recommend_count += 1
```

1-6：在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格；进行非对等操作时，如果是关系密切的立即操作符（如—>），后不应加空格。

2 注释

2-1：边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除。

2-2：注释的内容要清楚、明了，含义准确，防止注释二义性。
示例：

```
@@PUBLIC = 0    #公开
@@FRIEND = 1    #好友可见
@@PRIVATE = 2   #隐藏
```

2-3：全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时注意事项等的说明。

2-4：控制器头部应进行注释，列出：控制器的目的、功能以及一些常量的作用。
示例：

```
# 日志后台    实现日志的创建、存为草稿、修改、删除等
#日志权限
@@PUBLIC = 0    #公开
@@FRIEND = 1    #好友可见
@@PRIVATE = 2   #隐藏
#日志状态
@@DELETE = 1   #删除
@@DRAFT = 2    #草稿
```

3 标识符命名

3-1：标识符的命名要清晰、明了，有明确含义，变量用小写字母以及下划线组成。
示例：

```
@entries @entry_comments @entry_category
```

3-2：对于变量命名，禁止取单个字符（如 i、j、k...），建议除了要有具体含义外，还能表明其变量类型、数据类型等，但 i、j、k 作局部循环变量是允许的。
示例：

```
for entry in @entries
end
```

3-3：对于常量的命名应大写。
示例：

```
@@PUBLIC
```

3-4：可以使用局部变量替代的地方，一律使用局部变量，不允许使用实例变量等替代局部变量的情况。

4 可读性

4-1：注意运算符的优先级，并用括号明确表达式的操作顺序，避免使用默认优先级。
示例：

```
if comment.passport or (cookies[:passport] and @blog.user.passport == cookies[:passport])
```

4-2：避免使用不易理解的数字，用有意义的标识来替代。
示例：

```
@entry.status = @@DELETE
```

4-3：源程序中关系较为紧密的代码应尽可能相邻。

4-4：不要使用难懂的技巧性很高的语句，除非很有必要时。

5 方法、结构

5-1：并明确定义公共变量的含义、作用、取值范围及公共变量间的关系

5-2：视图中不应出现数据库查询语句，应封装到 helper 中。
示例：

```
<%= logged_in_user.passport %>
def logged_in_user
    if cookies[:passport]
```

```
      return Space.find_by_passport(cookies[:passport])
    else
      return nil
    end
  end
```

5-3：对于比较复杂的业务处理过程，应封装到 model 中。
示例：

```
def Space.create_widget(space_id)
    for widget in Widget.find(:all)
      space_widget = SpaceWidget.new
      space_widget.widget_id = widget.id
      space_widget.space_id = space_id
      space_widget.col = widget.init_col
      space_widget.row = widget.init_row
      space_widget.save
    end
 end
```

5-4：方法后应加括号。
示例：

```
@entry = Entry.new(params[:entry])
```

5-5：控制器名字应以复数的形式。
示例：

```
entries_controller
```

5-6：避免方法中的参数与方法中的变量重名。

5-7：控制器中所有自定义方法统一放到该控制器最下方，并限制这些方法的访问权限。

5-8：如需扩展现有类，例如：String 类，需将该扩展统一放到库里，所有的扩展类统一管理。禁止出现将该扩展类置入某个控制器中，或者某几个控制器中同时出现该类的扩展方法。

5-9：如需根据某个值，取出另一个值，不允许出现类似 if..else..的判断。

```
if card_type == Card::CARD_TYPE[:YEAR]
  年卡
elsif card_type == Card::CARD_TYPE[:HALF_YEAR]
  半年卡
end
```

替换成：

```
CARD_TYPE = [["年卡", '01'], ["半年卡", '02']]
Card::CARD_TYPE.rassoc(card_type)[0]
```

5-10：控制器中的方法可以封装的都封装到模型中，尽量减少每个方法的代码行。

5-11：判断是否为空，不允许出现使用 size 或者 length 的情况。

```
if @books and @books.size > 0
   @books.each do |book|
   end
end
```

替换成：

```
(@books || []).each do |book|
end
```

5-12：方法体结构或者 sql 类似的，能够封装的都进行封装，不允许出现一个控制器中多个方法类似的情况。

5-13：项目中自定义的方法，不允许出现多个方法实现相同功能或者类似功能，所有类似的方法必须合并，冗余的方法都去掉。

5-14：图片显示不允许使用，替换成标签形式，必要时重写 image_tag 方法，解决 firefox 下 alt 不能显示的问题。

5-15：页面上尽量减少 css 代码，所有的 css 代码必须统一放入样式表，加快页面的呈现速度。

5-16：控制器中过期的或者确定无用的方法必须删除，不允许出现大片注释代码。

5-17：不允许跳过 CSRF 防范。

类似 skip_before_filter :verify_authenticity_token 和 protect_from_forgery :except => ["batch_destroy", "stock"]之类的代码不允许出现。

5-18：所有的查询语句尽量都采用预加载子表的模式。

5-19：数据更新除非需要跳过验证，否则必须经过模型层验证，不允许出现直接跳过模型层验证而更新数据的代码。

5-20：所有的表单必须使用标签形式，禁止出现<form action=””></form>形式。