Code 2017-12-22
申思维

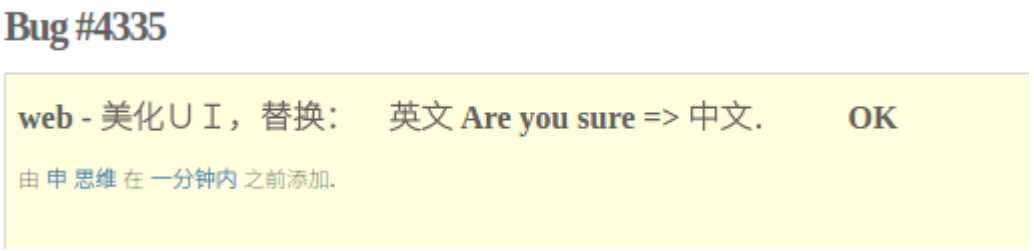
去掉不必要的尖括号

< %
    @ ke_fang1 = 0
    one_check_in_orders = ...
% >

```
<td>
  <% @ke_fang1 = 0%>
  <% one_check_in_orders = CheckInOrder.joins(room: :roomcategory).
      where("is_check_out = ? and check_in_orders.leave_date like ?
            true, "#{year}%", params[:company_id])%>
  <% one_check_in_orders.each do |c| %>
     <% @ke_fang1 += c.current_price.to_f * c.check_in_days%>
  <% end %>
  <%= @ke_fang1 %>
```

要使用中文

- Are you sure? => 您确定吗？



不要写废话代码

```
<td>
  <%# if User.find(user.id).is_enabled %>
  <% if user.is_enabled %>
    <%= link_to_modal to_set_sale_rate_users_pat
      class: '', title: "打折权限" do %>
       <%= "客房打折" %>
    <% end %>
```

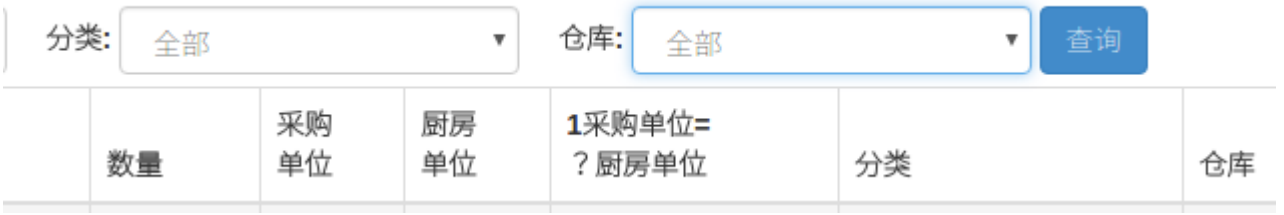
要使用 prepared condition

- where(“goods.name like ?”, params[:name])

```
@goods = @goods.where("goods.name like '%#{params[:name]}%'") if params
@goods = @goods.where(:goods_first_category_id => params[:category_id])
@goods = @goods.order('goods.created_at desc').page(params[:page])
```

下拉单的默认文字

-
```
("company_id = ?",
{:include_blank=>"请选择物品分类",
', :name => "category_id")%>
```
-
```
{:include_blank=>"请选择仓库", :sel
', :name => "storehouse_id")%>
```



要有四舍五入

2017-10-23	2288.4000000953674	0.0	0.0	0.0	0.0	0.0	0.0	404.0	0.0	572.5999999046326	2692.4000000953674
2017-10-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-10-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-10-20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-10-19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-10-18	0.0	524.7999877929688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	131.20001220703125	524.7999877929688

四舍五入数字

```
<td>
  <%= @dish_orders.where("payment = ?", "alipay").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "card").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "transfer").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "check").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "sign").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "hang").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= @dish_orders.where("payment = ?", "hui_yuan").sum(:actual_cost).to_f.round(2) %>
</td>
<td>
  <%= (@dish_orders.sum(:cost).to_f - @dish_orders.sum(:actual_cost).to_f).round(2) %>
```

酌情使用两种写法

```ruby
def operation_text
  case self.operation
  when 'in'
    '入库'
  when 'purchase_in'
    '采购订单入库'
  when 'out'
    '出库'
  when 'dish_out'
    '菜品出库'
  end
end

def chinese_operation_text operation
  case operation
  when 'in'
    '入库'
  when 'purchase_in'
    '采购订单入库'
  when 'out'
    '出库'
  when 'dish_out'
    '菜品出库'
  end
```

代码写在 controller 中

- 例如下面代码的第一行

```
<% GoodsFirstCategory.joins(goods: :purchase_orders).where("purchase_orders.purchase_l
  <% if category.restaurant_goods_second_categories.count != 0 %>
    <% category.restaurant_goods_second_categories.joins([{:goods => [:purchase_orders
    <label><%= category.name %></label> - <label><%= second_category.name %></label>
    <% purchase_orders = PurchaseOrder.joins(:good).where("purchase_orders.purchase_li
                                                          and goods.restaurant_goods_
                                                          @purchase_list.id, second_c
     <% purchase_orders.includes(:good).each do |order| %>
        <table>
          <thead>
            <tr>
              <th>物品名称</th>
              <th>采购数量</th>
              <th>库存数量</th>
              <% if @purchase_list.status == 'checking' || @purchase_list.status == 'a
                <th>
                  操作
                </th>
              <% end %>
              <th>
```

为啥不把表格合起来

餐厅 - 蔬菜

物品名称	采购数量	库存数量	状态
小芹菜	3.0/斤	12.0/斤	没有购买

物品名称	采购数量	库存数量	状态
小葱	2.0/斤	6.752/斤	没有购买

物品名称	采购数量	库存数量	状态
豆角	3.0/斤	-1.1/斤	没有购买

物品名称	采购数量	库存数量	状态
小油菜	2.0/斤	6.6/斤	没有购买

不要重复使用 joins

```
def index
  @in_and_out_storages = InAndOutStorage.joins(good: :goods_first_categor
  @in_and_out_storages = @in_and_out_storages.joins(:good).where("goods.s
                         params[:storehouse]) if params[:storehouse].pres
  @in_and_out_storages = @in_and_out_storages.joins(:good).where("goods.g
                         params[:good_category_id]) if params[:good_categ
  @in_and_out_storages = @in_and_out_storages.joins(:good).where("goods.r
                         params[:restaurant_goods_second_category_id]) if
  @in_and_out_storages = @in_and_out_storages.joins(:good).where("goods.n
                         "%#{params[:name]}%") if params[:name].present?
  @in_and_out_storages = @in_and_out_storages.order('in_and_out_storages.
  @company_id = current_user.company_id
```

不同表的外键不要占用同一个列

- 例如下面的 dish_order, purchase_list

```
class InAndOutStorage < ActiveRecord::Base
  belongs_to :staff, :class_name => Staff, :foreign_key => 'staff_id'
  belongs_to :staff, :class_name => Staff, :foreign_key => 'ling_qu_staff_id'
  belongs_to :good
  belongs_to :storehouse
  belongs_to :purchase_list, :class_name => PurchaseList, :foreign_key => 'order_id'
  belongs_to :dish_order, :class_name => DishOrder, :foreign_key => 'order_id'
```

includes 要写在根源那里

- 下方第二行的 includes 要写在　controller 中

```
<% if @in_and_out_storages.present? %>
  <% @in_and_out_storages.includes(:good).each do |in_and_out_storage| %>
    <tr>
      <td><%= in_and_out_storage.good.name %></td>
      <td><%= in_and_out_storage.quantity %>
       <%= in_and_out_storage.good.unit %></td>
      <% staff = Staff.find(in_and_out_storage.staff_id)%>
      <td><%= in_and_out_storage.dish_order.serial_number %></td>
      <td><%= in_and_out_storage.dish_order.dinner_table.name %></td>
      <td>
      </td>
    </tr>
  <% end %>
```

尽量写在 controller 中

- 特殊情况下，写了也可以 .

```
      <% @day_array.each do |day|%>
        <% dish_orders = DishOrder.includes(:dish_order_details => :dish).joins(:dinner
e).where("dish_orders.created_at >= ? and
        dish_orders.created_at <= ? and dinner_tables.company_id = ? and is_paid = ?",
        day.beginning_of_day, day.end_of_day, current_user.company_id, true)
        dish_orders.each do |dish_order|
```

太重复了　使用元编程简化

```ruby
shui_guo_purchase_orders.each do |order|
  @shui_guo_month += order.quantity.to_f * order.unit_price.to_f
end
gan_huo_purchase_orders.each do |order|
  @gan_huo_month += order.quantity.to_f * order.unit_price.to_f
end
tiao_liao_purchase_orders.each do |order|
  @tiao_liao_month += order.quantity.to_f * order.unit_price.to_f
end
liang_you_purchase_orders.each do |order|
  @liang_you_month += order.quantity.to_f * order.unit_price.to_f
end
dan_purchase_orders.each do |order|
  @dan_month += order.quantity.to_f * order.unit_price.to_f
end
dou_purchase_orders.each do |order|
  @dou_month += order.quantity.to_f * order.unit_price.to_f
end
guan_tou_purchase_orders.each do |order|
  @guan_tou_month += order.quantity.to_f * order.unit_price.to_f
end
chu_fang_yong_ju_purchase_orders.each do |order|
  @chu_fang_yong_ju_month += order.quantity.to_f * order.unit_price.to_f
end
```

```
ri_yong_pin_purchase_orders = PurchaseOrder.joins(good: :goods_first_category).
  where("goods_first_categories.company_id =? and goods_first_categories.name like ? ar
        params[:company_id], "%日用品%","#{params[:month]}%")
wei_xiu_purchase_orders = PurchaseOrder.joins(good: :goods_first_category).
  where("goods_first_categories.company_id =? and goods_first_categories.name like ? ar
        params[:company_id], "%维修%","#{params[:month]}%")
shui_chan_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_cat
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%水产%","#{params[:month]}%")
shui_guo_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_cate
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%水果%","#{params[:month]}%")
gan_huo_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_categ
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%干货%","#{params[:month]}%")
tiao_liao_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_cat
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%调料%","#{params[:month]}%")
liang_you_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_cat
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%粮油%","#{params[:month]}%")
dan_purchase_orders = PurchaseOrder.joins(:good => [{:restaurant_goods_second_category
  where("goods_first_categories.company_id =? and restaurant_goods_second_categories.na
        params[:company_id], "%蛋%","#{params[:month]}%")
```

使用元编程

- [‘shui_guo’, ‘gan_huo’...].each do |name|
  eval(“#{name}_purchase_orders”.....)
  end

判断放在前端

```ruby
def return
  if params[:return_count].present?
    return_count = params[:return_count].to_i
    if return_count > @dish_order_detail.count
      flash[:danger] = "退菜数量大于点菜数量 请重新输入"
    else

    end
      count = @dish_order_detail.count - return_count
      @dish_order_detail.update_attributes(
        :return_count => return_count,
        :count => count
      )
      # 更新订单总价
      dish_order = @dish_order_detail.dish_order
      dish_order.update_attributes :cost => dish_order.total_cost
  end
  redirect_to :back
end
```