Ruby on Rails 購物車實作

使用歐付寶（卷一）

by @tonytonyjan

大兜/簡煒航/tonytonyjan

- tonytonyjan.net
- Ruby on Rails 自習手冊作者
- TJDict Chrome 擴充套件作者
- 腦袋有動工作室負責人
- 微廣科技技術長
- Rails Girls Taipei 1~5 屆教練

專長

- Ruby (Rails)
- C (Qt)
- Java (Swing)
- Network Programming
- Information Retrieval

近況

- RubyConf Brazil 2015 講者（未來式）
- Confoo Canada 2015 講者
- Ruby Kaigi Japan 2014 講者
- Yahoo Hack Taiwan 2013 冠軍

本課前置知識假設學生已具備 RoR 基礎知識

```
task '本課' => 'Rails'
task 'Rails' => %w[Ruby CLI RDBMS 計網概 Git]
task '計網概' => '計概'
```

這堂課你會學到

- 購物車的實作與其相關知識
- 歐付寶金流
- 巢狀表單使用
- 後台製作技巧

且不需安裝任何 gem

User Story

User Story - 訪客

- 瀏覽商品
- 將商品放入購物車
- 調整購物車商品數量
- 建立訂單時需輸入付款資訊
- 將購物車商品移到下次再買清單
- 需要檢查庫存

User Story - 管理員

- 管理商品、訂單
- 可依照不同狀態瀏覽商品、訂單
- 商品狀態包括：新訂單、已出貨、已取消
- 金流狀態包括：等待付款、已付款

撿現成？

- Spree
  - 功能最成熟且全面。
- ror_ecommerce
  - 教學最詳盡。
- Piggybak
  - 以掛載（mount）實現，與 carrierwave 的設計哲學相同。

從何開始規劃網站？

設計網址

設計網址

- 瀏覽商品 -> /products
- 商品單頁 -> /products/:id
- 購物車頁 -> /cart
- 建立訂單頁 -> /orders/new
- 訂單顯示 -> /orders/:id
- 後台管理頁面 -> /admin/*

前置作業

```
git clone https://github.com/tonytonyjan/my_cart
cd my_cart
rails s
```

練習

在創造以下路由：

```
$ bin/rake routes
 Prefix Verb   URI Pattern                Controller#Action
   root GET    /                          products#index
product GET    /products/:id(.:format) products#show
   cart GET    /cart(.:format)            carts#show
```

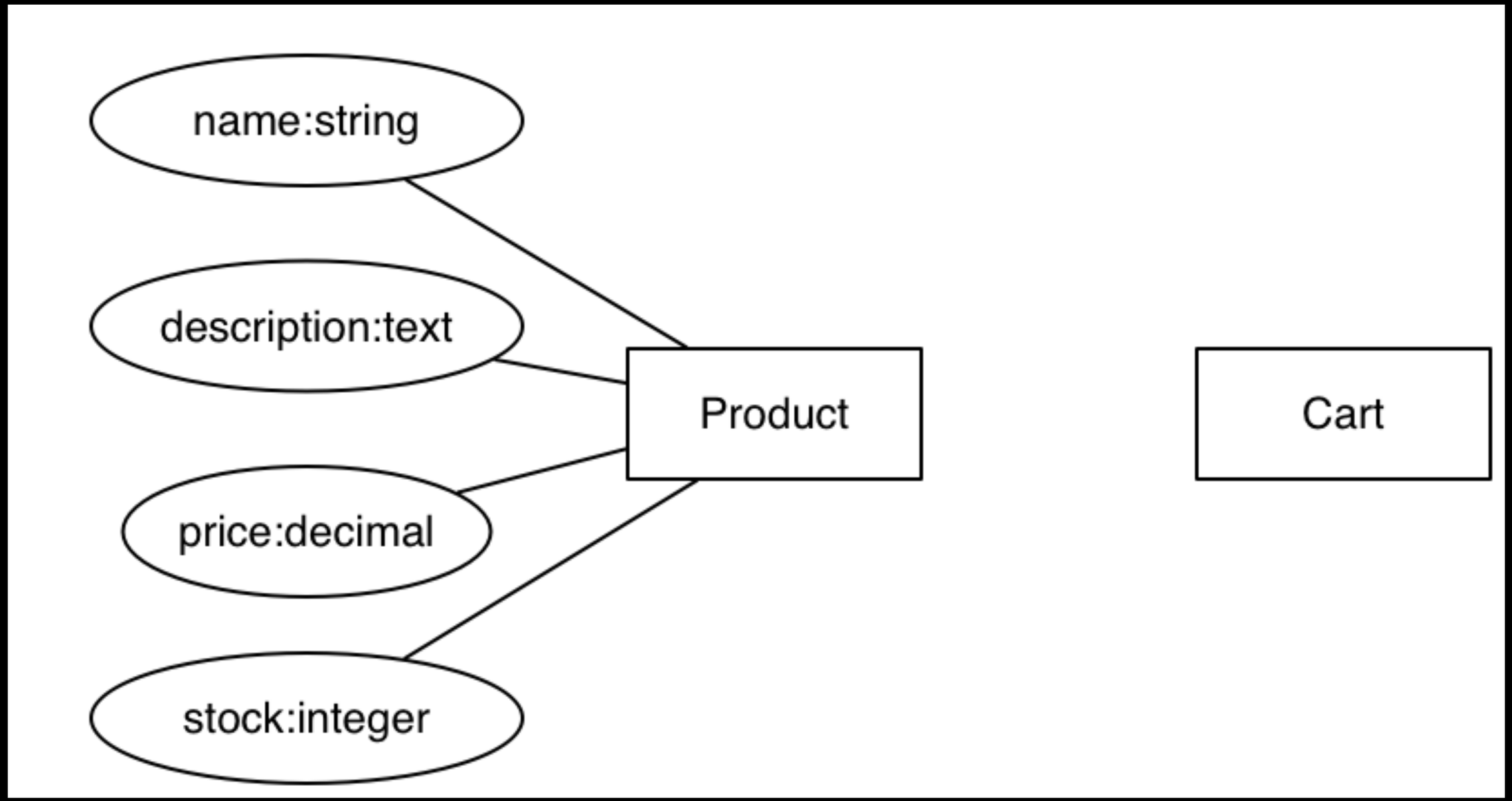
```ruby
# config/routes.rb
root 'products#index'
resources :products, only: :show
resource :cart, only: :show
```

設計資料（ERD）

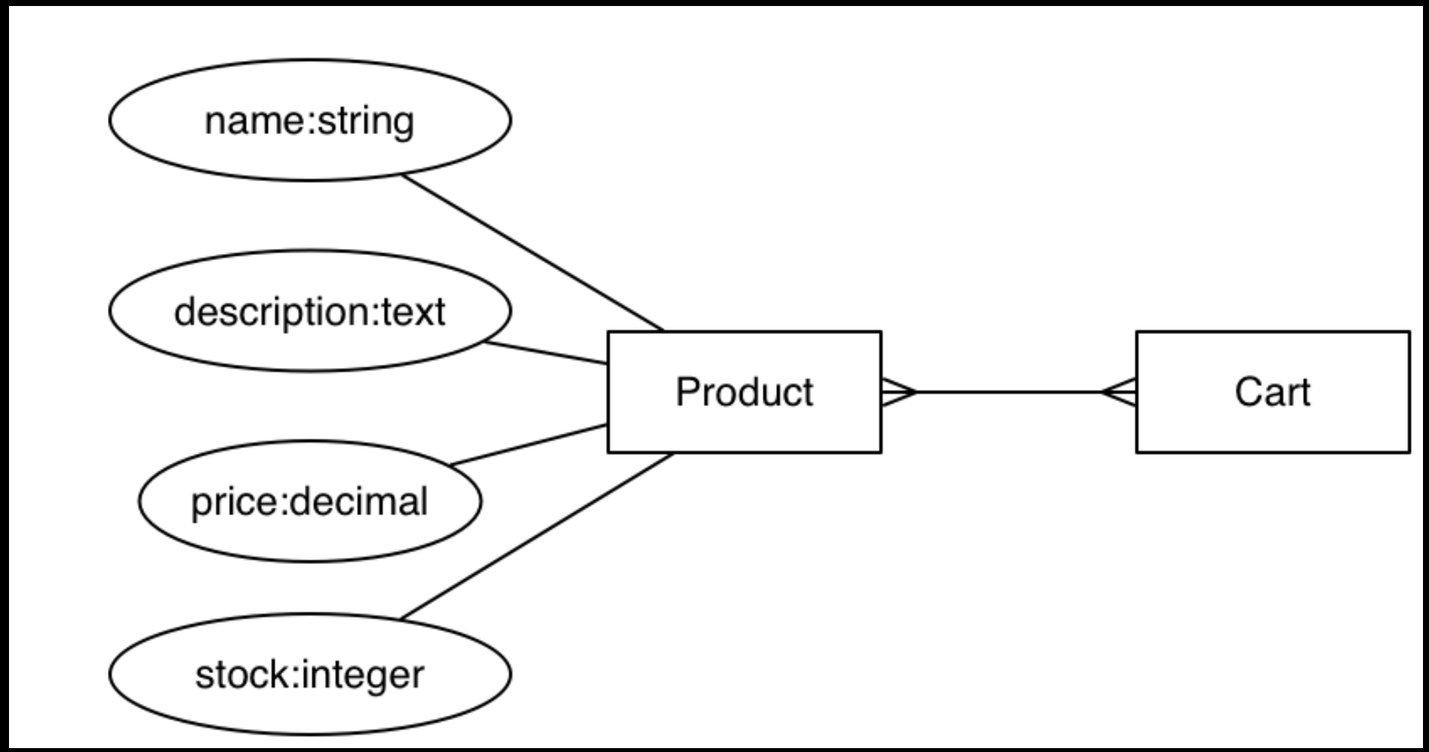
設計資料（ERD）

現實生活中的例子

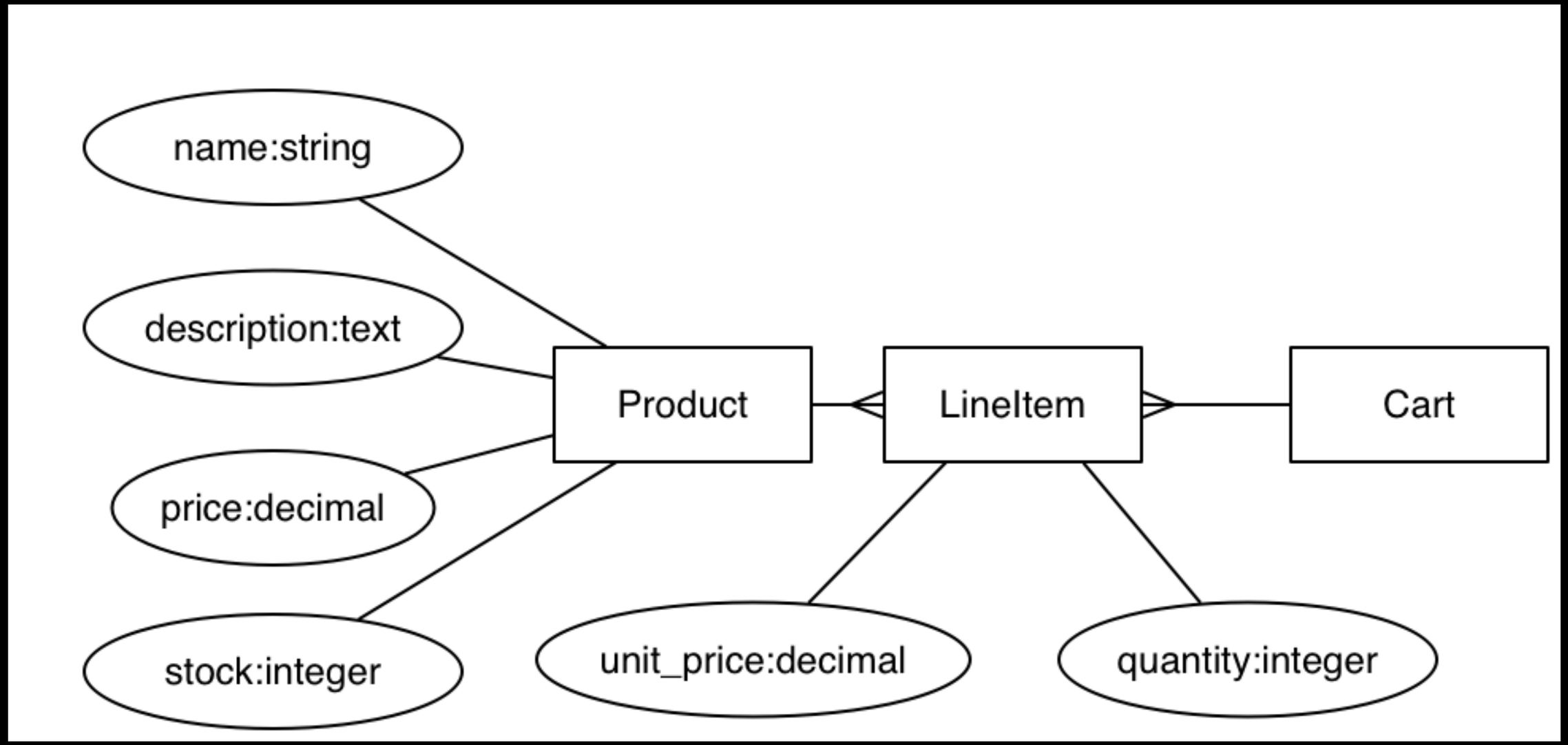
1. 逛賣場
2. 將商品放入購物車
3. 結帳


name:string
description:text
price:decimal
stock:integer
Product
Cart

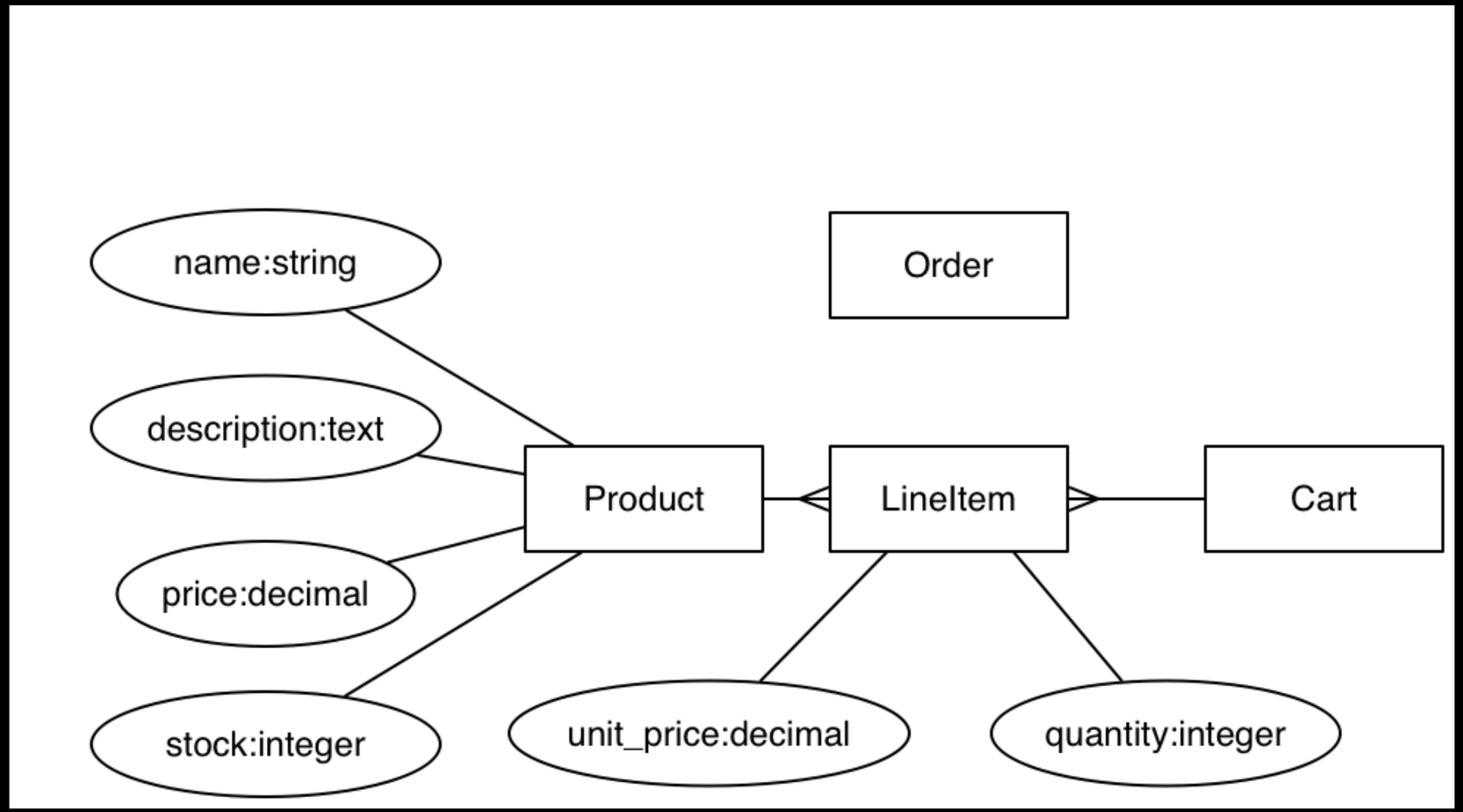
想一想

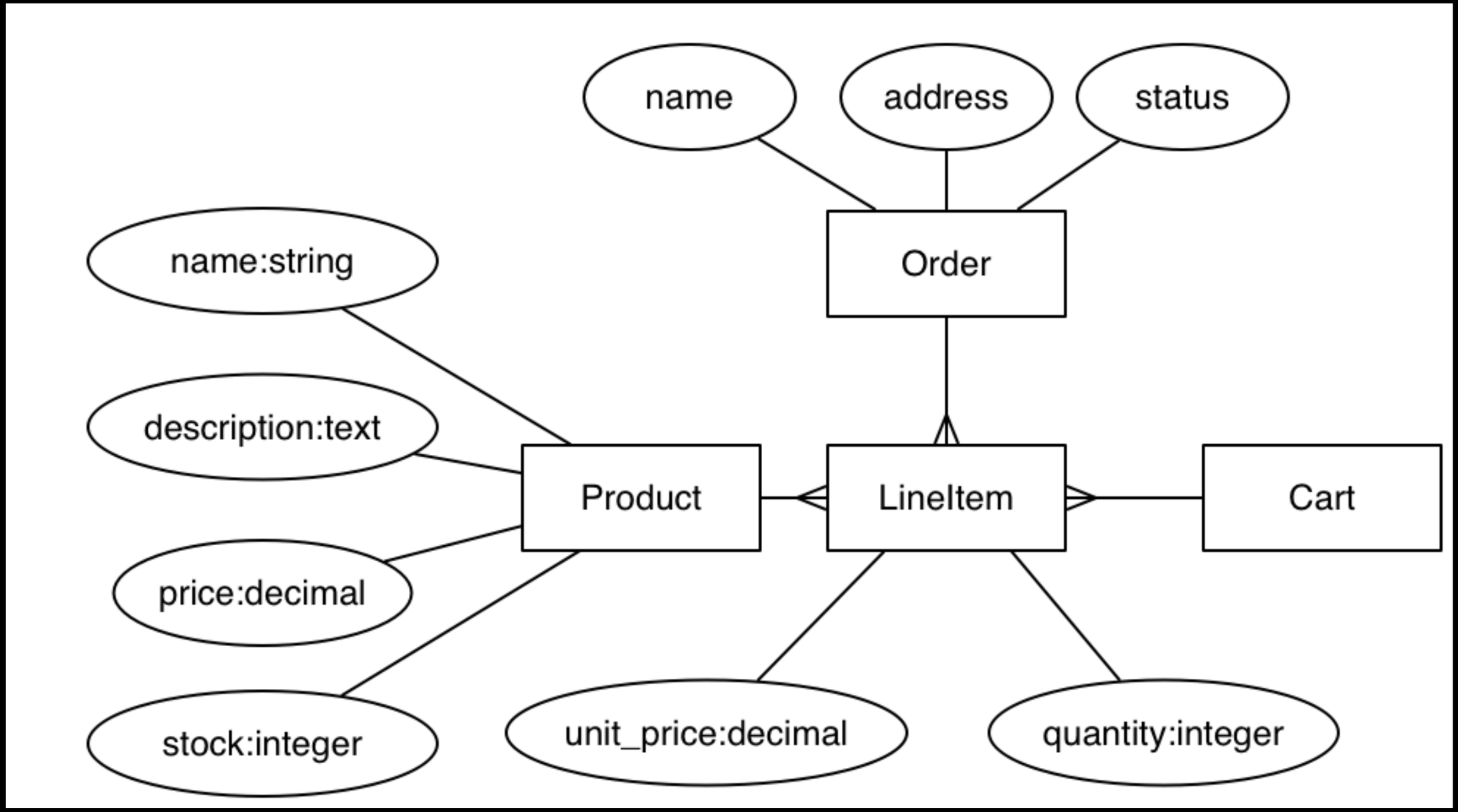


這張圖有什麼問題？


name:string
description:text
price:decimal
stock:integer
Product
LineItem
Cart
unit_price:decimal
quantity:integer

怎麼設計？


name:string
Order
description:text
Product
LineItem
Cart
price:decimal
stock:integer
unit_price:decimal
quantity:integer


name
address
status
Order
name:string
description:text
Product
LineItem
Cart
price:decimal
stock:integer
unit_price:decimal
quantity:integer

其他人怎麼做？

- Agile Web Development with Rails 書中採用此架構
- ror_ecommerce 使用此架構
- Spree、Piggybak 沒有 Cart entity，是靠 Order 上的狀態判定

執行：

```
rails g model product name description:text price:decimal stock:integer
rails g model order name address status payment_method
rails g model cart
rails g model line_item \
order:references cart:references product:references \
unit_price:decimal quantity:integer
```

建立關聯

```ruby
# app/models/cart.rb
class Cart
  has_many :line_items
end

# app/models/product.rb
class Product
  has_many :line_items
end

# app/models/order.rb
class Order
  has_many :line_items
  has_many :products, through: :line_items
end
```

修改資料庫遷移檔

```ruby
# db/migrate/VERSION_create_line_items.rb
t.decimal :unit_price, null: false
t.integer :quantity, null: false, default: 1

# db/migrate/VERSION_create_products.rb
t.string :name, null: false
t.text :description, null: false
t.decimal :price, null: false, default: 0
t.integer :stock, null: false

# db/migrate/VERSION_create_carts.rb
t.string :status, null: false, default: '新訂單'
```

rake dev:setup

rails g task dev fakeup

```ruby
namespace :dev do
  desc "Generate fake data"
  task fakeup: ['db:schema:load', :environment] do
    20.times do |i|
      Product.create(
        name: "product no.#{i}", description: "description no.#{i}",
        price: (rand(10) + 1) * 50, stock: rand(91) + 10
      )
    end
    cart = Cart.create
    Product.all.sample(5).each do |product|
      cart.line_items.create product: product, unit_price: product.price, quantity: rand(4) + 1
    end
  end
end
```

練習

實作以下方法：

- Cart#total
- Cart#empty?
- Cart#clear
- Order#total
- LineItem#subtotal

產生 controller

```
 Prefix Verb URI Pattern                Controller#Action
   root GET  /                          products#index
product GET  /products/:id(.:format)    products#show
   cart GET  /cart(.:format)            carts#show
```

```
rails g controller products index show
rails g controller carts show
```

current_cart

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
  before_action :set_current_cart, if: ->{ Rails.env.development? }
  helper_method :current_cart

  def current_cart
    @current_cart ||= Cart.find_or_create_by(id: session[:cart_id])
    session[:cart_id] = @current_cart.id
    @current_cart
  end

private

  def set_current_cart
    session[:cart_id] = 1
  end

end
```

練習

- 產品列表頁
- 產品單頁
- 購物車頁

加入購物車

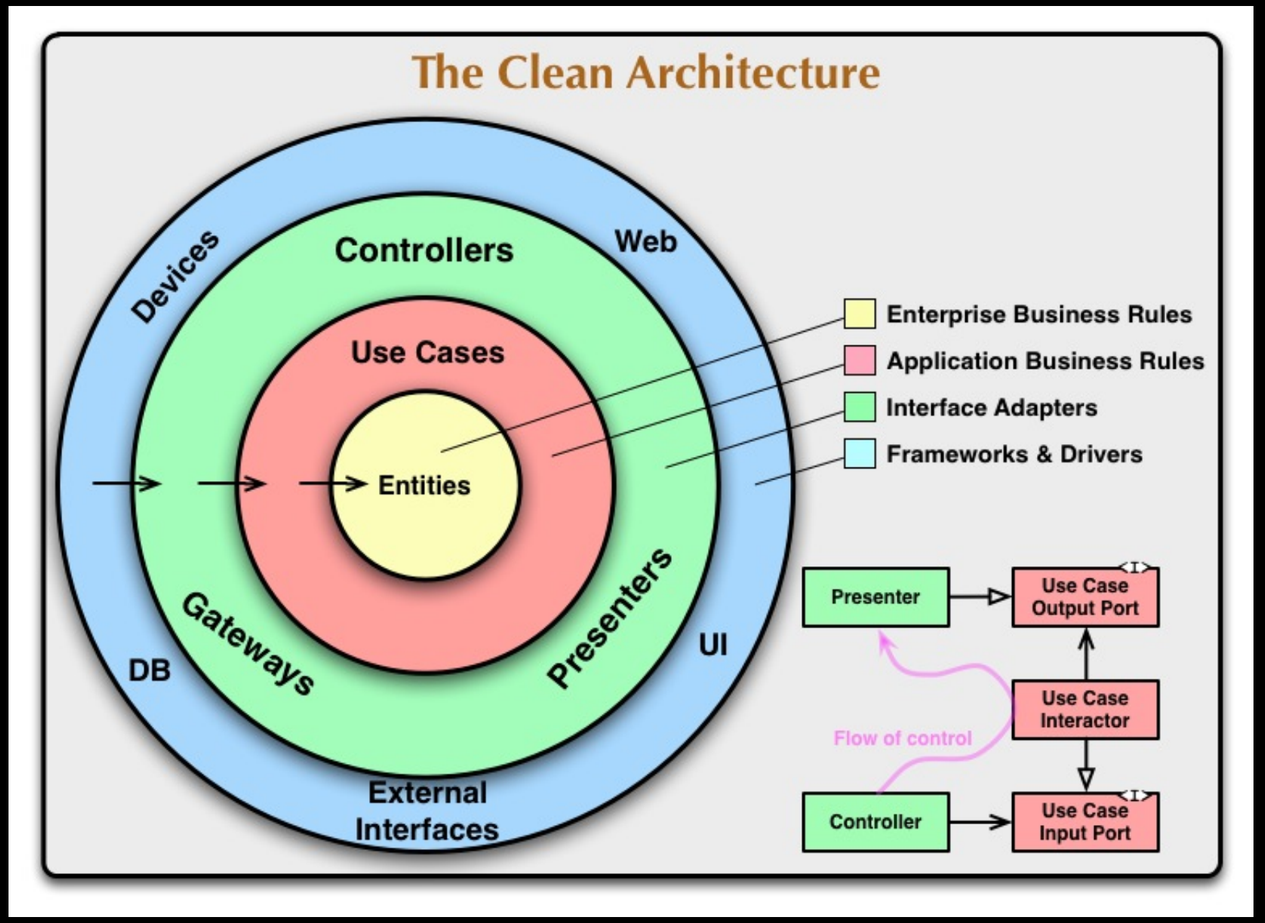
想一想

```
@cart.add_product @product
@product.add_to_cart @cart
```

哪一個正確？

Rails 的邏輯應該在哪裡，已經戰很久了

Clean Architecture by Uncle Bob



其實加入購物車、編輯購買數量等功能可視為 LineItem 的 CRUD

顯示加入購物車按鈕

```ruby
# config/routes.rb
resources :line_items, only: %i[create update destroy]


# app/controllers/line_items_controller.rb
class LineItemsController < ApplicationController
  def create
    @line_item = current_cart.line_items.new line_item_params
    if @line_item.save
      redirect_to cart_path, notice: '成功更新'
    else
      @product = @line_item.product
      render 'products/show'
    end
  end

private

  def line_item_params
    params.require(:line_item).permit(:quantity, :product_id)
  end
end
```

檢查庫存、初始化價錢

```ruby
# app/models/line_item.rb
validate :check_stock
before_save :set_unit_price

def check_stock
  errors.add(:quantity, 'out of stock') if quantity > product.stock
end

def set_unit_price
  self.unit_price = product.price
end
```

顯示按鈕

```ruby
# app/controllers/products_controller.rb
def show
  @product = Product.find params[:id]
  @line_item = current_cart.line_items.find_or_initialize_by(product: @product)
end
```

```erb
<!-- app/views/products/show.html.erb -->
<h1><%= @product.name %></h1>
<p>敘述<%= @product.description %></p>
<p>價錢：<%= @product.price %></p>
<p>庫存：<%= @product.stock %></p>

<%= form_for @line_item do |f| %>
  <%= f.number_field :quantity %>
  <%= f.hidden_field :product_id %>
  <%= f.submit '加入購物車' %>
<% end %>
```

練習

- LineItemsController#destroy
- LineItemsController#update

更新購物車

設定巢狀賦值

```ruby
# app/models/cart.rb
accepts_nested_attributes_for :line_items
validates_associated :line_items
```

會新增特定的抽象屬性，例如：

```ruby
@cart.line_items_attributes = {
  0 => {quantity: 10, id: 123},
  1 => {quantity: 10, id: 123}
}
```

練習

rails console

```
cart = Cart.first
cart.line_items_attributes = {0 => {quantity: 10, product_id: __}} # create
cart.line_items_attributes = {0 => {id: __, quantity: 10}} # update
cart.save
```

使購物車可更新

```ruby
# config/routes.rb
resource :cart, only: %i[show update]


# app/controllers/carts_controller.rb
class CartsController < ApplicationController
  def update
    if current_cart.update cart_params
      redirect_to cart_path, notice: '已更新'
    else
      render :show
    end
  end

private

  def cart_params
    params.require(:cart).permit(line_items_attributes: [:id, :quantity])
  end
end
```

```
<!-- app/views/carts/show.html.erb -->
<%= f.fields_for :line_items do |ff| %>
  <% line_item = ff.object %>
  <tr>
    <td><%= link_to line_item.product, line_item.product %></td>
    <td><%= ff.number_field :quantity %></td>
    <td></td>
  </tr>
<% end %>
```

會產生許多：

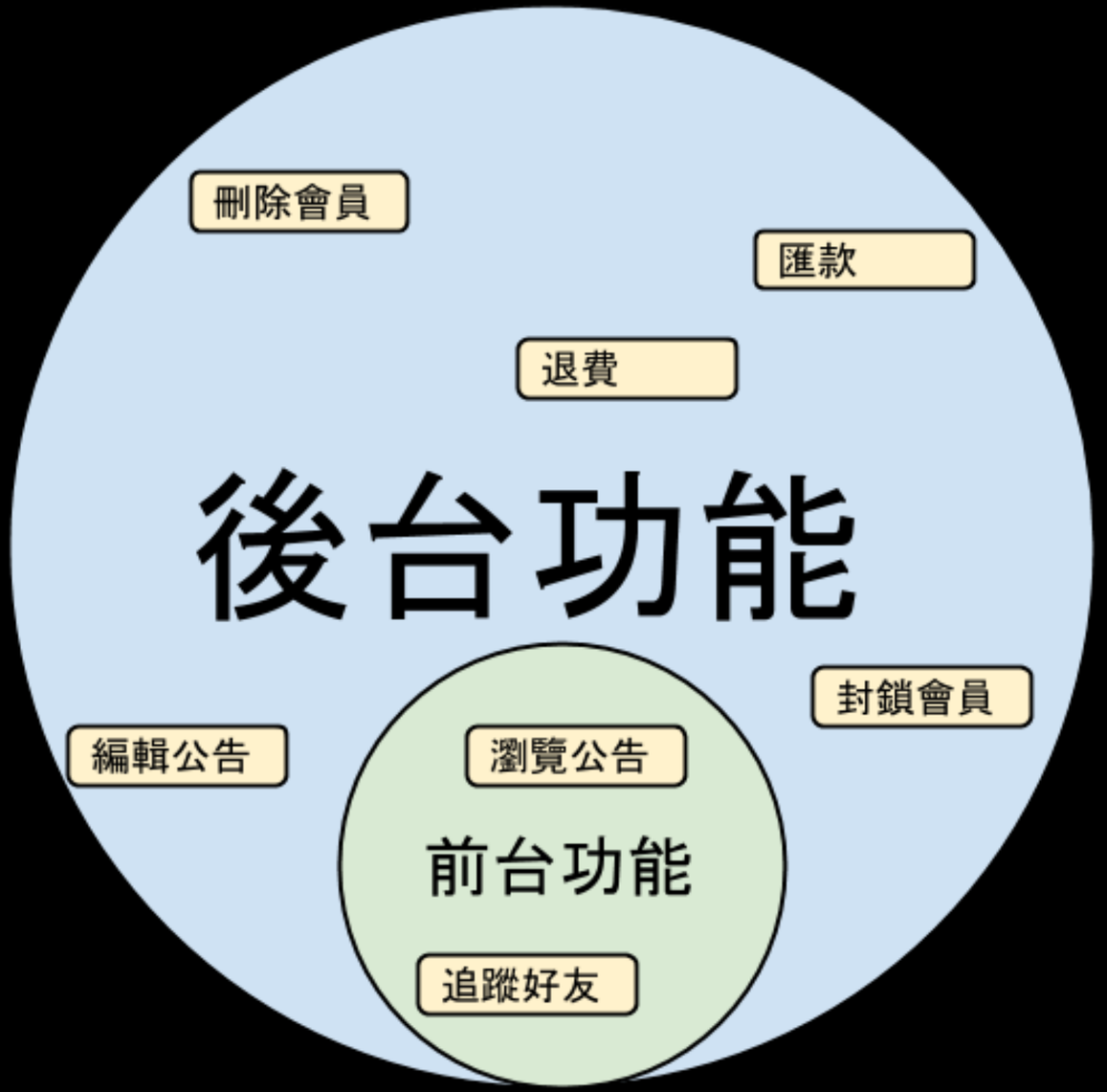
```
<input type="number" name="cart[line_items_attributes][1][quantity]"/>
<input type="hidden" name="cart[line_items_attributes][1][id]"/>
```

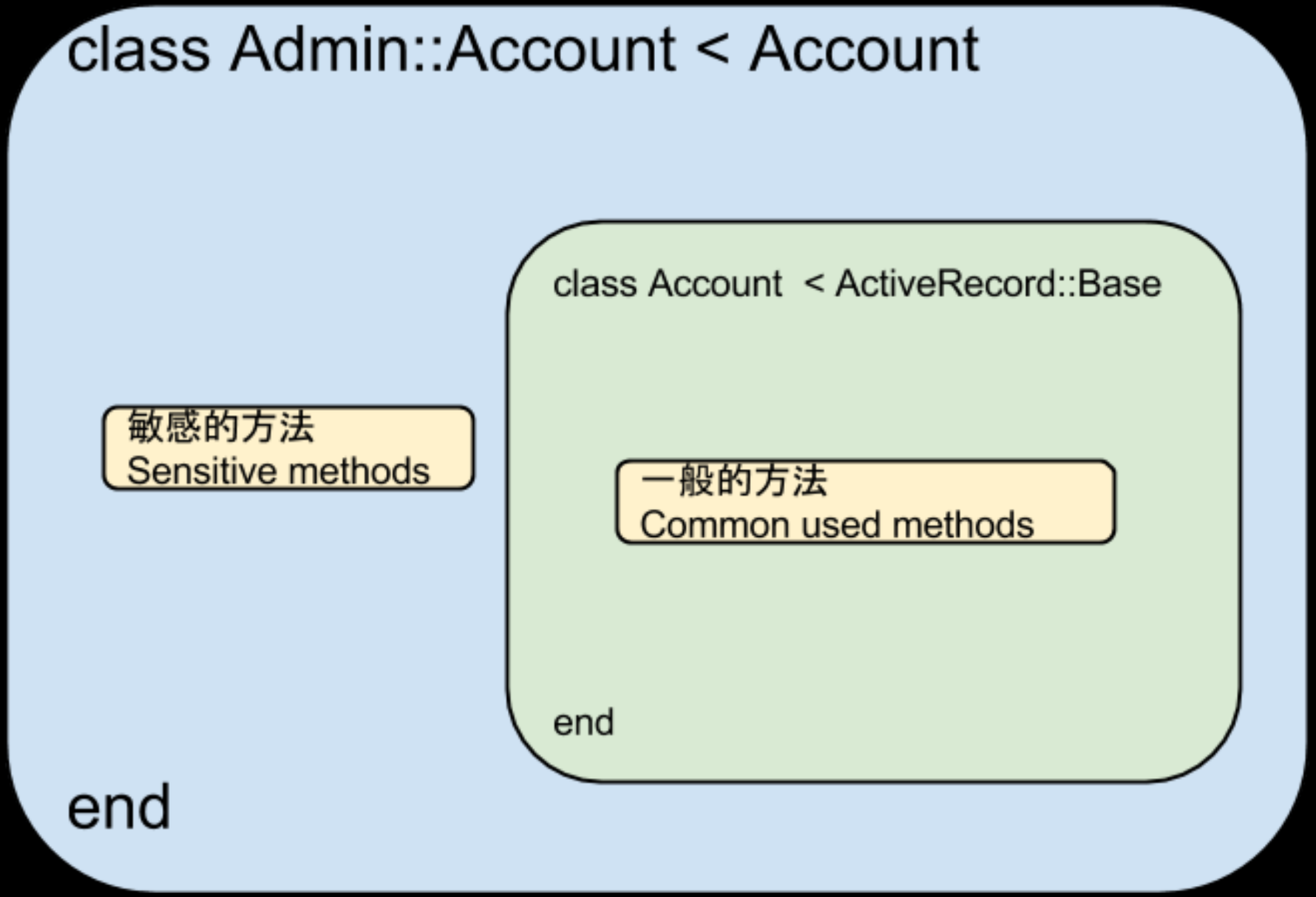
簡易後台

RubyConf Taiwan 2014 講題

打造漂亮的 Rails 後台

slides/video


刪除會員
匯款
退費
後台功能
封鎖會員
編輯公告
瀏覽公告
前台功能
追蹤好友


class Admin::Account < Account
敏感的方法
Sensitive methods
class Account < ActiveRecord::Base
一般的方法
Common used methods
end
end

```ruby
# app/models/account.rb
class Account < ActiveRecord::Base
  ...
end

# app/models/admin/account.rb
class Admin::Account < Account
  ...
  def transfer_money(account)
    ...
  end
  ...
end
```

- 所有 Account 可以做的事情，Admin::Account 都可以做
- 敏感的方法被放在 Admin 名稱空間下
- Admin::Account 只會在後台的 controller 用到

路徑問題

前台：

```
link_to('foo', @post) # href="/posts/:id"
form_for(@post)       # action="/posts" or "/posts/:id"
```

後台：

```
link_to('foo', [:admin, @post])                # href="/admin/posts/:id"
link_to('foo', admin_post_path(@post))         # href="/admin/posts/:id"
form_for(@post, url: admin_posts_path)         # action="/admin/posts"
form_for(@post, url: admin_post_path(@post)) # action="/admin/posts/:id"
```

使用名稱空間

如果 @post 是一個 Admin::Post：

```
link_to('foo', @post) # href="/admin/posts/:id"
form_for(@post)       # action="/admin/posts/:id" or "/admin/posts"
```

實作流程

假設我們有一個 Post model，製作後台流程為：

1. 修改 `config/routes.rb`
2. 新增 `app/models/admin/post.rb`
3. `app/controllers/admin/posts_controller.rb`
4. `app/views/admin/posts/{index,new,edit}.html.erb`

scaffold

利用 scaffold 製作後台

執行：

```
rails g scaffold Admin::Product name description:text --parent=Product
rails g scaffold Admin::Order name address status --parent=Order
```

搭配樣板：

```
lib
└── templates
    └── erb
        └── scaffold
            ├── _form.html.erb
            ├── edit.html.erb
            ├── index.html.erb
            ├── new.html.erb
            └── show.html.erb
```

課後作業

- 美化網站。
- 補上快閃訊息、導覽列。
- 新增 `lib/templates/erb/scaffold/show.html.erb`，使用 table 呈現資料。
  - 參考 `bundle show railtie`/lib/rails/generators/erb/scaffold/templates/show.html.erb。
- 想一想，`Product.stock` 應該在購物的哪個階段扣除？