Rails Engine 组件化开发实践

覃明圆

- PPT 使用 markdown 编写
- https://github.com/marp-team

Work Design

https://github.com/work-design

- 3年
- 30 个

为什么要造轮子?

- 提升开发效率，降低开发成本
- 一次撸码，到处运行 (Write Once, Run Everywhere)
  DRY(Dont Repeat Yourself)
- 更完善、更成熟的业务逻辑
  如登陆基于 auth_token，相对于 user_id 更安全和支持更多功能

现有的轮子难以满足需求

- 难以 override: 如 Devise
- 难以配置，如：SimpleForm
- 难以迁移，沉没成本高，如： ActionAdmin

我们要造的轮子

- 集成简单，大部分安装后即可直接使用
- 尽可能减少配置
- 尽可能减少 DSL，DSL 即是学习成本

怎么造

- 从现有项目中剥离，避免纸上谈兵
- 尽可能理解业务的本质
- 主动出击，影响老板和产品经理

造轮子痛点

- 交付优先：造轮子产生的成本与交付项目之间的平衡
- 二八法则：用 20% 的开发成本满足 80% 的场景
- 易于 Override：基于二八法则，ruby在这方面极具优势

什么是Rails Engine

- Model、Controller、View 的集合体

Rails Egnine 要解决的问题

- 如何使用：在项目中引入，尽量避免改动祖传代码
- 如何覆写 Override

Model

如何定义模型

```
rails_auth
  app
    models
      rails_auth
        user.rb
      user.rb
```

定义 model 里的方法

```ruby
# rails_auth/app/models/rails_auth/user.rb

module RailsAuth::Account
  extend ActiveSupport::Concern
  included do
    attribute :identity, :string

    belongs_to :user
  end

  def send_token
    puts 'implement this in project'
  end
end
```

定义 model

```ruby
# rails_auth/app/models/account.rb
class Account < ApplicationRecord
  include RailsAuth::Account
end unless defined? Account
```

如何 Override

```ruby
class Account < ApplicationRecord
  include RailsAuth::Account

  attribute :identity, :integer

  def send_token
    SmsHelper.send(self.identity, '666666')
  end
end
```

此举意义

1. 易用，什么都不用干，即可使用现成的 Model
2. 易扩展，易override
3. 易理解，很容易知道相应的 model 在哪些 engine 里有定义（`Account.ancestors`）

不需要写 Migration

实现了 Django 引以为傲的 自动迁移功能

如何使用

bin/rails g rails_com:migrations

此举的意义?

- 懒：程序员第一生产力
- DRY：只需要在一个地方定义 model 的属性
- Engine 开发更容易，不用 install migrations，可以放心大胆的去调整 Model

```ruby
class Adminer < ApplicationRecord
  include RailsAuth::User
end
```

```
bin/rails g rails_com:migrations
```

- 使用 rails_com:migrations 后：

```ruby
class RailsComMigration < ActiveRecord::Migration[6.0]
  def change
    create_table :adminers do |t|
      t.string :name
      t.timestamps
    end
  end
end
```

Controller / View

Controller

```ruby
class Auth::Admin::UsersController < Auth::Admin::BaseController

  def create
    @user = User.new(user_params)

    unless @user.join(params)
      render :new, locals: { model: @user }, status: :unprocessable_entity
    end
  end

end
```

跟平常所见的 Controller 有什么区别?

- 没有 redirect_to
- 没有 respond_to
- 几乎没有逻辑

此举的意义?

- 尽可能将控制能力转移到 View 层
- 尽可能将业务逻辑转移到 Model 层

转移控制能力到 View 层

- 模板语言的学习成本更低：erb, jbulder
- View 层 Override 更灵活，更强大
- 极大的减少了 View 层代码量

Write View

- 可取的示例变量：instance_variables - _protected_ivars
- 条件判断和循环
- 可使用的 helper 方法：link_to，等常用方法
- form_build: default_form

RailsDoc 即将完成

- 将 controller 的对象 的属性进行输出

Override View

view 查找路径

```
My::AgenciesController.ancestors
```

我们的改造

- 改造前：
  模板文件的正则匹配
  依据搜索路径匹配到 模板名 符合要求即渲染
- 改造后：
  提升了 format 格式的优先级
  匹配到 模板名+ format 才进行渲染

View 提供的默认模板

https://github.com/work-design/rails_com/tree/master/app/views/application

```
# create.erb.html / update.erb.html / destroy.erb.html
<script>
  if (typeof Turbolinks === 'undefined') {
    window.location = document.referrer
  } else {
    Turbolinks.clearCache()
    Turbolinks.visit(document.referrer, {action: 'replace'})
  }
</script>
```

```
# new.json.jbuider / edit.json.jbuilder
json.error model.errors.as_json(full_messages: true)
json.message model.error_text
```

常规只涉及到增删改查

```
# maintain_logs
_filter.html.erb  # 搜索
_form.html.erb  # 表单
_maintain_log.json.jbuilder
_show.json.jbuilder
_show_table.html.erb
index.html.erb
index.json.jbuilder
```

Override Controller

重新定义路由

- override url
- override helper 方法

```ruby
Rails.application.routes.draw do
  scope :my, module: 'agency/my', as: :my do
    resources :agencies
  end
end
```

```ruby
# In Project
Rails.application.routes.draw do
  namespace :my do
    resources :agencies, only: [:new] do
      collection do
        get :search
      end
    end
  end
end
```

Assets

- webpacker 支持 Engine
- js / css 依据 controller / action 分离，代码清晰

示例

https://github.com/work-design/rails_auth/tree/master/app/assets/javascripts/controllers

Override Assets

项目中同路径

Override I18n

- 项目中同路径
- enum 支持

Q & A

Rails Engine 为什么不用 namespace 隔离

- 意义：
  - 可以从各个层次，各个姿势进行复用
  - 懒：不用每个 model 都带 namespace
- 解决：
  - 路由 _path / _url helper 方法冲突，忍忍吧，override一下
  - model 冲突，已解决，自己定义model, engine里自动失效
  - controller 已有 namespace，不会冲突

代码分布在各个 engine 里，是否加大了读代码的难度；

- 用 engine:
  我不知道去哪找相关的定义的代码
- 不用 engine:
  如果一个 model 进行了拆分，你可能也不知道去哪读代码

解决方案

- 使用 RubyMine
  - 看源码的功能
  - 把多个项目放到同一个工作区

入坑指南

- 三无产品！
  - 无测试
  - 无文档
  - 无备注

这是我们接下来的工作重点