**KỸ THUẬT LẬP TRÌNH ASP.NET MVC 5 (C#)**

**PHÂN TÍCH, THIẾT KẾ VÀ CÀI ĐẶT GIAO DIỆN NGƯỜI DÙNG (VIEWS)**

**Phan Minh Tài** *| Founder of TitiSoft|CafeT.vn|ChuyenTin.vn|…*

**MỞ ĐẦU**

Hiện tại tôi không chuyên về thiết kế giao diện web (Web Design) cho nên cũng không thể bàn sâu vào các vấn đề như thiết kế giao diện phù hợp với người dùng (UX), hay các kỹ thuật sâu liên quan đến Photoshop, CSS, Html, … Ở đây tôi muốn bàn luận những kỹ năng cơ bản có thể thiết kế giao diện cho các ứng dụng ASP.NET MVC 5 ở mức chấp nhận được.

**ASP.NET MVC 5 | VIEWS**

Views – là tập hợp tất cả các “giao diện” của ứng dụng dùng để giao tiếp với người dùng (ở tất cả các role), nó sẽ bao gồm đầy đủ các trang (View) và các thành phần dùng để cấu tạo nên trang (PartialView), ngoài ra, để có thể xây dựng các View chúng ta cần có các đoạn mã Javascript, Jquery, Html, Css, … là những cấu phần cơ bản nhất để hình thành nên các “giao diện” người dùng.

View – đang nói đến từng trang cụ thể, chẳng hạn như Home – Trang chủ, hay Articles/Details – trang chi tiết bài viết, …

ASP.NET MVC 5 (Template) thường tổ chức các Views theo cấu trúc như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Views | |  |  |  |
|  | Articles |  |  |  |
|  |  | Index | Danh sách các bài viết |  |
|  |  | Details |  |  |
|  |  | Edit |  |  |
|  |  | Delete |  |  |
|  |  | Create |  |  |
|  |  | … | Có thể định nghĩa thêm các views khác tùy theo bối cảnh người dùng hoặc cách mà chúng ta thiết kế giao diện |  |
|  | Questions |  |  |  |
|  |  | CRUD | Bao gồm Index, Create, Edit, Delete |  |
|  |  | \_QuestionMenu | PartialView – thường thì bắt đầu bằng “\_” |  |
|  |  | \_QuestionHeader |  |  |
|  |  | \_QuestionFooter |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Shares |  |  |  |
|  |  | Layout |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**TỔ CHỨC CÁC VIEWS NHƯ THẾ NÀO ?**

Thông thường thì một website ASP.NET MVC 5 có bố cục như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Views | |  |  |  |
|  | Shares |  | Các trang dùng chung |  |
|  |  | \_Layout | Danh sách các bài viết |  |
|  |  | \_Header |  |  |
|  |  | \_Footer |  |  |
|  |  |  |  |  |
|  |  | Create |  |  |
|  |  | … | Có thể định nghĩa thêm các views khác tùy theo bối cảnh người dùng hoặc cách mà chúng ta thiết kế giao diện |  |
|  | Advs |  | Quảng cáo |  |
|  |  | CRUD | Bao gồm Index, Create, Edit, Delete |  |
|  |  | \_QuestionMenu | PartialView – thường thì bắt đầu bằng “\_” |  |
|  |  | \_QuestionHeader |  |  |
|  |  | \_QuestionFooter |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Articles |  |  |  |
|  |  | Index |  |  |
|  |  | Details |  |  |
|  |  | Edit |  |  |
|  |  | Delete |  |  |
|  |  | \_Header |  |  |
|  |  | \_Footer |  |  |
|  |  | \_Info |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Trang chủ (Homde/Index)**

Xét trang CafeT.vn (đang xây dựng - và chia sẻ bài viết này), trang chủ có các cấu phần sau đây:

* Layout: Đương nhiên phải có, và Layout = {\_Header, \_Footer, \_Advs, \_Articles, …}

Nhận xét:

* Để dễ tổ chức, ta nên tổ chức các views theo cấu trúc của các models, chẳng hạn:

Views

* + Articles
    - Index
    - Delete
    - Details
    - Create
    - \_Header
    - \_Footer
    - \_Info
    - \_Tags
* Do trên trang Articles/Details, có cấu trúc:
  + Articles/Details
    - Thành phần chính là nội dung bài viết
    - \_Header
    - \_Footer
    - \_Info
    - \_Menu
    - \_Comments
      * \_CommentItem
        + \_Comment
        + \_CommentMenu
    - \_Relateds: Các bài viết liên quan
      * \_ArticleItem
      * \_ArticleMenu
    - \_Questions
  + Articles/Index
    - ArticleItems
    - \_ItemMenu
    - \_ItemInfo
    - \_ItemHeader
    - \_ItemFooter
    - \_Paging
    - \_MenuForItem

Nhìn một cách tổng thể, ta thấy nên tổ chức như sau:

* Models (Views)
  + Index
  + Details
  + Delete
  + Edit
  + \_DetailMenu
  + \_DetailHeader
  + \_DetailFooter
  + \_DetailInfo
  + \_Paging
  + Shares\Models (view)
    - Model
      * \_Items (dùng cho trang Index)
      * \_Item (dùng cho \_Items)
      * \_ItemMenu (dùng cho \_Item)
      * \_ItemHeader (dùng cho \_Item)
      * \_ItemFooter (dùng cho \_Item)
      * \_ItemInfo (dùng cho \_Item)

**Tạo bản sao cho một đối tượng bất kỳ (Clone)**

* Đây cũng là một bài toán hay gặp, chẳng hạn trong ngữ cảnh của ứng dụng hiện nay tôi có nhu cầu khá rõ nét:
  + Giả sử bài viết ban đầu (do người dùng biên tập) sẽ được giữ nguyên và đánh dấu là Layer[0] – hình dung theo nghĩa trải bài viết lên một mặt phẳng.
  + Tôi muốn phân tích sâu hơn bài viết hoặc xử lý nó theo nhiều tình huống khác nhau tùy theo bối cảnh, chẳng hạn ở đây:
    - Layer[1] – Tôi hiệu chỉnh chính tả.
    - Layer[2] – Tôi đánh dấu chỉ số (dưới) của tất cả các từ có trong văn bản.
    - Layer[3] – Tôi tiến hành dịch nó sang một ngôn ngữ nào đó (chẳng hạn tiếng Anh) – thường thì tôi dùng công cụ dịch như Google hoặc Microsoft, sau đó lưu kết quả lại, khi cần thì dùng.
    - Layer[4] – Tôi đánh dấu tất cả các từ chưa biết (so khớp với từ điển cá nhân), …
    - ….

Như vậy thì các Layer này sẽ được tổ chức như thế nào ?



**Viewers – Người xem đối tượng**

* Đối với một đối tượng bất kỳ (mang tính bussiness) thường thì có mấy hành động cụ thể đối với nó:
  + Xóa
  + Sửa
  + Tạo
  + Xem
  + (và hành động không phải trực tiếp – Tìm kiếm)
* Việc theo dõi xem ai đã (Xem, Sửa, Xóa, Tạo) và đảm bảo an toàn khi thực thi các hành động trên đối với từng đối tượng thực sự cần thiết.
* Một biểu mẫu lấy yêu cầu cụ thể có thể như sau:
  + Lớp/Đối tượng
  + Ý nghĩa
  + Mục đích sử dụng
  + Tương tác
    - Xem: {Ai, như thế nào, ảnh hưởng ra sao}
    - Xóa {Ai, như thế nào, …}
* Như vậy về bản chất đối tượng sẽ phải có:
  + Các hành động kiểm tra trước khi tương tác với đối tượng nên để cho ObjectManager<T> thực thi.
  + Bản thân nội tại nó không tự nó thực thi được các hành động như CRUD, tuy nhiên nó có thể tự quyết định được việc (Xem) sẽ được thực hiện như thế nào cũng như quá trình xem nó thì xem được những gì, và khi bị xem nó phản ứng như thế nào?
    - ObjectManager<T>.GetToView()
    - ObjectManager<T>.GetToDelete()
    - ….

**Biến hình các đối tượng (Transform)**

**LỚP UrlObject**

Cũng là một lớp các đối tượng thường gặp trong các bài toán liên quan đến Data/Web Minning, và trong thực tế tôi cũng sử dụng UrlObject để khai thác dữ liệu cho các website.

UrlObject

* string url: Là địa chỉ đầu vào, việc khởi tạo nó cần kiểm tra, nếu url.IsUrl() (– hàm mở rộng tự xây dựng) thì mới khởi tạo thành công, còn nếu không thì không tạo được.

**BÀI 01**

**TRANG CHỦ (Home/Index)**