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

**KIẾN TRÚC ỨNG DỤNG, TỔ CHỨC MÃ NGUỒN**

**VÀ MỘT SỐ KỸ THUẬT CƠ BẢN**

Biên soạn: Phan Minh Tài

Founder of CafeT.vn|ChuyenTin.vn

**MỞ ĐẦU**

Khi đọc bài viết này chắc các bạn đang quan tâm đến việc tìm hiểu về ASP.NET MVC và muốn tìm kiếm một mô hình nào đó “tương đối” hiệu quả cho việc xây dựng các ứng dụng ASP.NET MVC để triển khai trong thực tế. Đây cũng là vấn đề mà khi mới làm quen với ASP.NET MVC tôi cũng quan tâm để tìm kiếm. Thực tế thì có khá nhiều bài viết đề cập đến chủ đề này, tuy nhiên các ví dụ minh họa bên trong các bài viết đó có thể hơi khó để chúng ta thử nghiệm sử dụng và dùng thường xuyên để suy nghĩ lâu hơn. Do đó, tôi viết lại bài này với mục đích chia sẻ kinh nghiệm trong quá trình xây dựng CafeT.vn|ChuyenTin.vn|ChuyenToan.vn|… để các bạn có thể tham khảo và sẵn sàng sử dụng hơn.

Kèm theo đó là mã nguồn mẫu, tôi sẽ chia sẻ tại GitHub (của CafeT.vn) để các bạn tham khảo.

**TỔNG QUAN KIẾN TRÚC ỨNG DỤNG ASP.NET MVC 5**

Như các bạn đã thấy, khi tạo các ứng dụng ASP.NET MVC 5 với Visual Studio 2017 (tôi đang sử dụng công cụ này), thì template mặc định của ứng dụng khi tạo sẽ có mô hình như sau:

Solution <Name>

* Project<Name>
  + Models
  + Controllers
    - AccountController
  + Views
    - Home

Đây là một mô hình tổ chức mã nguồn (rất cơ bản) tuân thủ mô hình kiến trúc ứng dụng ASP.NET MVC. Tuy nhiên để có thể xây dựng các ứng dụng lớn hơn (quy mô vừa phải mang tính thực tế) thì chúng ta còn cần phải bổ sung thêm nhiều thứ, chẳng hạn:

* ModelViews
* Mappers
* ServiceLayers

**TỔ CHỨC CÁC VIEWS**

Mô hình mẫu

* Views
  + <Model1> [View]
  + <Model2>
  + Shares
    - Layout
    - Layouts

**Layouts**

**Views cho các Model**

Ta thấy rằng, hầu hết các đối tượng chính mang ý nghĩa trực tiếp về mặt tính năng (nghiệp vụ) thường sẽ tương tác với người dùng (hay đúng hơn là người dùng tương tác với các model này) dưới dạng các thao tác chính:

* **Index:** Liệt kê toàn bộ các đối tượng với thông tin đủ dùng, có thể áp dụng một vài kỹ thuật tại đây như phân trang, phân loại, sắp xếp, tìm kiếm, …. Nhưng chính nhất vẫn là “Liệt kê các đối tượng”.
* **Details:** Xem chi tiết từng đối tượng
* **Edit:** Chỉnh sửa đối tượng
* **Delete:** Xóa đối tượng
* Search <từ khóa>
* Filter <Bộ lọc>

Tuy nhiên trong thực tế còn có thêm vài tình huống khác, lấy vị dụ cụ thể về việc xây dựng Views cho Article, ta thấy như sau:

* **Trang chủ:** Liệt kê các bài viết (không phải toàn bộ - mà là số lượng có hạn, hiển thị tin mới trước), kèm theo đó như các bạn thấy sẽ có một vài vùng bố trí những thông tin nổi bật như “Xem nhiều”, “Bình luận nhiều”, … Với những yêu cầu này ta sẽ thấy cần có các Views sau:

**Views**

* + **Article**
    - **Index**
    - **Edit**
    - **Delete**
    - **Details**
  + **Shares**
    - **Articles**
      * **\_ItemForHome**
      * **\_ItemsForHome**
      * **\_ItemMeta**
      * **\_ItemInfo**
      * **\_ItemAuthor**
      * **\_ItemByAuthor**
* **Articles**
* **Comments**
  + Index
  + Delete
  + Edit
  + Details
  + Shares
    - Comments
      * \_ItemsOfArticle
      * \_ItemsOfQuestion
      * \_ItemsOfCoure
      * \_ItemOfArticle
      * \_ItemOfQuestion
      * \_ItemOfCourse
      * \_ItemUserMenu
      * \_ItemAdminMenu

Nhìn thì rõ ràng khá mắc công, vì chúng ta tạo ra quá nhiều view, nhưng theo tôi nó cần thiết để tách bạch vấn đề và dễ dàng chỉnh sửa, tránh đụng chạm khi một PartialView dùng chung quá nhiều lần khi chỉnh sửa sẽ đụng chạm đến nhiều giao diện khác. Ngoài ra, nếu muốn làm nhanh thì ta nên bổ sung:

* \_Item
* \_ItemMenu
* \_ItemInfo
* \_ItemHeader
* \_ItemFooter