Nghiên cứu mô hình kiến trúc lập trình theo mô hình MVC

Trình bày: Nguyễn Thị Sim

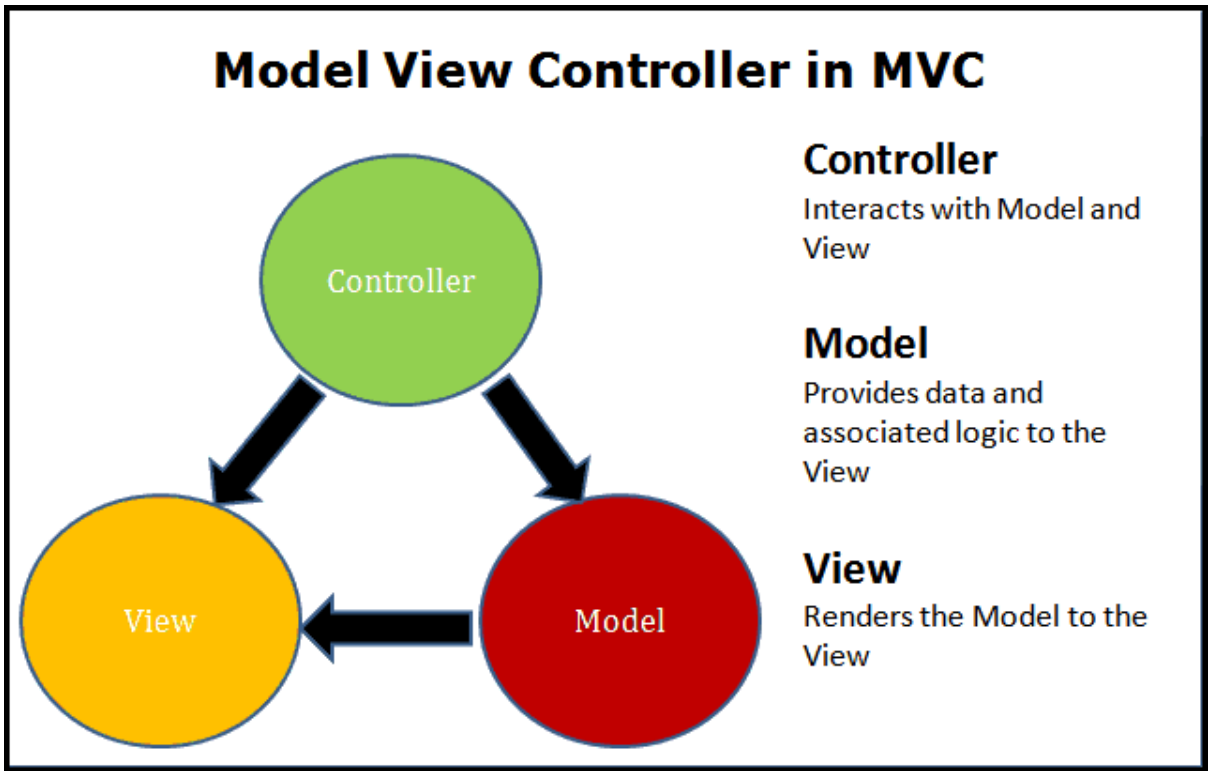
Time: 2019 October

Contents

Giới thiệu về mô hình MVC

MVC được hình thành bởi các nghiên cứu của Trygve Reenskaug vào khoảng năm 1978. Model, view và controller (MVC) là một kiến trúc phát triển ba lớp nổi tiếng được sử dụng cho các phát triển ứng dụng. Tài liệu này trình bày cái nhìn liên quan đến các lớp MVC, cách sử dụng, ưu điểm của nó liên quan đến kiến trúc lập trình trong quá trình phát triển ứng dụng web.

MVC (Model, view và controller) là một mẫu kiến trúc thường được sử dụng trong các ứng dụng dựa trên web. Nó cung cấp ba tầng chính : model, view và controller. Nhiều nhà lập trình sử dụng MVC như một mẫu thiết kế chuẩn.



MVC là một phương pháp khung phân chia việc triển khai ứng dụng thành ba vai trò thành phần: Model, View và Controller

- " Model " trong ứng dụng dựa trên MVC là các thành phần của ứng dụng chịu trách nhiệm duy trì trạng thái. Thường thì trạng thái này được duy trì bên trong cơ sở dữ liệu (ví dụ: chúng ta có thể có một class Product được sử dụng để thể hiện dữ liệu từ bảng Products bên trong SQL).
- " View " trong ứng dụng dựa trên MVC là các thành phần chịu trách nhiệm hiển thị giao diện người dùng của ứng dụng. Thông thường, giao diện người dùng này được tạo ra từ dữ liệu mô hình(model data) (ví dụ: chúng tôi có thể tạo "Chỉnh sửa" view sản phẩm hiển thị các textboxs, dropdowns và checkbox dựa trên trạng thái hiện tại của đối tượng Product).
- " Controller " trong ứng dụng dựa trên MVC là các thành phần chịu trách nhiệm xử lý tương tác của người dùng cuối, thao tác với mô hình và cuối cùng chọn chế độ xem để hiển thị để hiển thị UI. Trong một ứng dụng MVC, view chỉ là về hiển thị thông tin - đó là bộ điều khiển xử lý và đáp ứng đầu vào và tương tác của người dùng.

Mẫu kiến trúc MVC được thiết kế theo ba lớp cơ bản. Nó phân chia các đặc tính của ứng dụng thành những thành phần riêng biệt. Lớp thứ nhất liên quan đến logic đầu vào của người dùng (input), lớp thứ hai có liên quan đến business logic và lớp thứ ba được sử dụng để hiện thị giao diện người dùng. MVC cung cấp sự kết hợp mở giữa ba model-view-controller. Mẫu MVC được sử dụng để xác định vị trí của mỗi logic trong ứng dụng và làm việc độc lập với nhau. Nghĩa là cùng một thời điểm, một developer sẽ làm việc trên logic đầu vào của người dùng(controller logic), developer khác sẽ làm việc trên logic giao diện người dùng(view) và developer thứ ba sẽ làm việc trên business logic(model). Phần thứ hai trong nghiên cứu này sẽ lý giải về việc sử dụng mẫu kiến trúc MVC và lợi ích của việc dùng mẫu kiến trúc MVC. Trong phần ba chúng ta mô tả thành phần của MVC framework.

Trong kiến trúc MVC, mỗi bộ ba Model-View-Controller được thiết kế tương ứng cho các đối tượng mà người dùng có thể tương tác.

Model nắm giữ trạng thái, cấu trúc và các hành vi của dữ liệu được thể hiện và tương tác bởi người dùng. Model không phụ thuộc và tương tác trực tiếp lên các thành phần khác. Thay vì vậy, khi có thay đổi, nó thông báo cho những thành phần như View tương úng thông qua cơ chế là Observer pattem. Model còn cung cấp phương tiện để các thành phần khác tương tác lên nó.

View lấy các thông tin từ Model và trình bày đến người dùng. Trên cùng một model, có thể có nhiều View cùng đăng ký. Khi có một thay đổi từ Model, tất cả các View đều được thông báo thông qua observer mà nó đã đăng ký.

Mỗi View khi được tạo ra sẽ tạo ra một Controller đi kèm theo nó. Trong khi các View đảm nhận kết xuất dữ liệu thì các Controller đảm nhận việc xử lý dữ liệu từ người dùng. Với mỗi sự kiện nhận được, Controller có thể xử lý và tương tác trực tiếp lên thành phần View

và Model tương ứng để đáp trả. View và Controller là hai thành phần cấu thành nên giao diện người dùng của ứng dụng. Chúng lưu giữ liên kết trực tiếp đến Model. Trong khi Controller có thể thay đổi dữ liệu theo yêu cầu của người dùng thì View tương tác để lấy dữ liệu cập nhật vào chính nó từ Model.

Lợi ích của kiến trúc theo mô hình MVC

Khi sử dụng đúng cách, mẫu *MVC* giúp cho người phát triển phần mềm cô lập các nguyên tắc nghiệp vụ và giao diện người dùng một cách rõ ràng hơn. Phần mềm phát triển theo mẫu *MVC* tạo nhiều thuận lợi cho việc bảo trì vì các nguyên tắc nghề nghiệp và giao diện ít liên quan với nhau.

Nền tảng ASP.NET MVC mang lại những lợi ích sau:

- Kiến trúc MVC giúp chúng ta điều khiển độ phức tạp của ứng dụng bằng chia nó thành ba thành phần model, view và controller, việc đó làm cho quá trình phát triển - quản lý - vận hành - bảo trì web diễn ra thuận lợi hơn, tạo ra được các chức năng chuyên biệt hoá đồng thời kiểm soát được luồng xử lý.
- MVC không sử dụng view state hoặc server-based form, bởi vì nó là ý tưởng cho những nhà phát triển muốn kiểm soát hoàn toàn hành vi ứng dụng của họ.
- Hỗ trợ tốt hơn cho mô hình phát triển ứng dụng hướng kiểm thử (TDD)
- MVC sử dụng front controller pattern. Front controller pattern xử lý nhiều request đầu vào sử dụng một giao diện(controller). Front controller cung cấp kiểm soát tập trung. Chúng ta cần để cấu hình duy nhất một controller trong web server thay vì phải cấu hình nhiều.
- Front controller cung cấp nhiều hỗ trợ kết nối bộ định tuyến để thiết kế web ứng dụng.

Lợi ích của ứng dụng được xây dựng trên nền tảng Web Forms

- Nó hỗ trợ cách lập trình hướng sự kiện, quản lý trạng thái trên giao thức HTTP, tiện dụng cho việc phát triển các ứng dụng Web phục vụ kinh doanh. Các ứng dụng trên nền tảng Web Forms cung cấp hàng tá các sự kiện được hỗ trợ bởi hàng trăm các server controls.
- Mô hình này sử dụng view state hoặc server-based form, nhờ đó sẽ giúp cho việc quản lý trạng thái các trang web dễ dàng.
- Nó rất phù hợp với các nhóm lập trình viên quy mô nhỏ và các thiết kế, những người muốn tận dụng các thành phần giúp xây dựng ứng dụng một cách nhanh chóng.
- Nói tóm lại, áp dụng Web Forms giúp giảm bớt sự phức tạp trong xây dựng ứng dụng, bởi vì các thành phần (lớp Page, controls,…) được tích hợp chặc chẽ và thường thì giúp bạn viết ít code hơn là áp dụng theo mô hình MVC.

Khi nào sử dụng kiến trúc mô hình MVC

Kiến trúc MVC pattern cho chúng ta ý tưởng của việc tách biệt mối quan tâm, nó giúp chúng ta thực hiện tách quan hệ giữa các lớp model, view và controller trong các ứng dụng. Phân chia quan hệ giúp dễ dàng để kiểm tra(test) ứng dụng như mối quan hệ giữa các thành phần của ứng dụng rõ ràng hơn và mạch lạc. MVC giúp chúng ta thực hiện một cách tiếp cận với việc xây dựng hướng kiểm thử, thực hiện kiểm thử tự động trước khi viết code. Những test case đơn vị giúp chúng ta xác định trước và xác nhận những yêu cầu của code trước khi viết chúng.

Nếu chúng ta tạo một ứng dụng với kích thích để nghiêm trọng trên client side để từ chối tạo cùng với JavaScript. Nếu chúng ta đang lập trình một ứng dụng mà chủ yếu ở phái máy chủ (server side) và chỉ một ít giao tiếp ở phía máy khách(client side) thì không nên sử dụng kiến trúc MVC pattern, thay vào đó nên sử dụng khởi tạo đơn giản như web-based form model(webform). Sau đây là những đặc điểm nhận dạng giúp chúng ta phân biệt khi nào nên sử dụng kiến trúc MVC và khi nào không:

- Ứng dụng cần giao tiếp không đồng bộ trên backend.
- Ứng dụng có chức năng mà các kết quả không cần tải lại toàn bộ trang ví dụ như nhận xét bài đăng trên Facebook hoặc các trang load scroll vô hạn...
- Thao tác dữ liệu chủ yếu trên máy khách(browser) hơn máy chủ(server side)
- Cùng một loại dữ liệu được chuyển giao theo những cách khác nhau trên cùng một trang(navigation)
- Khi ứng dụng có nhiều kết nối không đáng kể được sử dụng để sửa đổi dữ liệu(button, switches)

Các công cụ và công nghệ được sử dụng với mô hình MVC

Với kiến trúc MVC có rất nhiều tool và công nghệ có thể được sử dụng để lập trình ứng dụng. Tùy thuộc vào nhu cầu của các nhà lập trình, họ có thể sử dụng nhiều công cụ và công nghrrj để lập trình ứng dụng. Đây là những công cụ và công nghệ được sử dụng để lập trình ứng dụng với kiến trúc MVC.

Công cụ

- Visual Studio: không chỉ là một công cụ mà còn là môi trường phát triển hoàn thiện cung cấp các cơ sở để khởi tạo các loại ứng dụng khác nhau. Khi chúng ta muốn xây dựng ứng dụng ASP.NET MVC framework thì visual studio rất hữu dụng.
- MYSQL Server – server quản lý cơ sở dữ liệu quan hệ để duy trì cơ sở dữ liệu
- SQL Server- một database engine để duy trì cơ sở dữ liệu như MYSQL server.
- Net Beans-IDE(integrated development environment) cung cấp môi trường hoàn thiện để xây dựng các ứng dụng khác nhau.
- Glassfish Server: server ứng dụng Java EE

Công nghệ

- HTML, CSS, JQUERY, AJAX cho thiết kế
- Servlet and Java server pages (JSP) used with Net beans
- EJB (Enterprise Java beans) technologies
- JSTL (Java server pages standard tag libraries)
- ASP.NET MVC được sử dụng với Visual studio
- JDBC (Java Database Connectivity) cho phép kết nối các chương trình viết bởi Java với các hệ quản trị cơ sở dữ liệu.

Có rất nhiều công cụ và công nghệ có thể được sử dụng với kiến trúc MVC, nhưng chúng ta chỉ tập trung vào các công cụ và công nghệ mà sẽ sử dụng để xây dựng web ứng dụng sử dụng kiến trúc MVC.

Tìm hiểu ASP.NET MVC Framework

ASP.NET framework cho phép phân tách rõ ràng các mối quan tâm, khả năng kiểm tra và TDD theo mặc định. Tất cả các core contract trong khung MVC đều dựa trên giao diện và có thể dễ dàng giả định (nó bao gồm các dựa trên giao diện IHttpRequest / IhttpResponse intrinsics). Bạn có thể đơn vị kiểm tra ứng dụng mà không phải chạy Controller trong quy trình ASP.NET (giúp kiểm tra đơn vị nhanh). Bạn có thể sử dụng bất kỳ khung kiểm tra đơn vị nào bạn muốn - thực hiện kiểm tra này (bao gồm NUnit, MBUnit, MS Test, v.v.).

Framework này rất mở rộng và có khả năng plug. Mọi thứ trong khung MVC được thiết kế sao cho có thể dễ dàng thay thế / tùy chỉnh (ví dụ: bạn có thể tùy ý plug-in view engine của riêng bạn, routing policy, parameter, v.v.). Nó cũng hỗ trợ sử dụng dependency injection và IOC container models (Windsor, Spring.Net, NHibernate, v.v.).

Nó bao gồm một thành phần ánh xạ URL rất mạnh cho phép bạn xây dựng các ứng dụng với các URL rõ ràng. Các URL không cần phải có các tiện ích mở rộng bên trong chúng và được thiết kế để dễ dàng hỗ trợ các mẫu đặt tên thân thiện với SEO và REST. Ví dụ: tôi có thể dễ dàng ánh xạ URL : /products/edit/4 để thực hiện action "Edit" của class ProductsController trong dự

MVC framework hỗ trợ sử dụng các tệp đánh dấu ASP.NET .ASPX, .ASCX và .Master làm "xem mẫu" (có nghĩa là bạn có thể dễ dàng sử dụng các tính năng ASP.NET hiện có như các trang chính lồng nhau, <% =%> đoạn trích, khai báo điều khiển máy chủ, mẫu, liên kết dữ liệu, bản địa hóa, v.v.). Tuy nhiên, nó không sử dụng mô hình hậu kỳ hiện có để tương tác trở lại máy chủ. Thay vào đó, bạn sẽ định tuyến tất cả các tương tác của người dùng cuối đến một class Controller thay vào đó - điều này giúp đảm bảo phân tách rõ ràng các mối quan tâm và khả năng kiểm tra (điều này cũng có nghĩa là không có vòng đời trang hoặc chế độ xem với các chế độ xem dựa trên MVC).

ASP.NET MVC framework hỗ trợ đầy đủ các tính năng ASP.NET hiện có như forms/windows authentication, URL authorization, membership/roles, output and data caching, session/profile state management, health monitoring, configuration system, the provider architecture, etc.

Nếu bạn đang tìm cách xây dựng các ứng dụng web của mình bằng cách sử dụng phương pháp MVC, tôi nghĩ chọn ASP.NET MVC framework rất rõ ràng và dễ sử dụng.

Tổng kết

Trong bài nghiên cứu này, chúng ta đã thảo luận về mô hình kiến trúc MVC. Chúng ta đã tìm hiểu về mô hình MVC, thảo luận về những lợi ích khác nhau của kiến trúc MVC cũng như đã thảo luận về một số mô hình MVC và đã tìm hiểu về ASP.NET MVC framework và các công cụ cũng như công nghệ. Kiến trúc MVC có ảnh hưởng lớn hơn trong thế giới của ứng dụng dựa trên web. Nó rất hữu ích cho các nhà phát triển. Các nhà phát triển có thể dễ dàng tạo các ứng dụng web của họ bằng kiến trúc MVC. Nó làm giảm độ phức tạp của ứng dụng và chia ứng dụng thành ba thành phần chính là model, view và controller. Nó cung cấp toàn quyền kiểm soát ứng dụng web cho các nhà phát triển. Sử dụng thử nghiệm kiến trúc MVC của các thành phần khác nhau của ứng dụng trở nên rất dễ dàng. Kiến trúc MVC cung cấp mức độ trừu tượng cao hơn cho các ứng dụng web.