# 1. Chương 1 - Giới thiệu

* Về tiểu luận chuyên ngành của học kỳ này, nhóm sẽ tiến hành xây dựng một website cho phép người sử dụng tạo lập, quản lý và tự động đánh giá concept map (bản đồ khái niệm).
* Mục tiêu của việc xây dựng website này là nhằm tạo ra một sản phẩm phục vụ cho việc dạy và học hiệu quả thông qua việc tạo lập và sử dụng các bản đồ khái niệm.
* Bên cạnh đó, giúp sinh viên rèn luyện và trau dồi các kỹ năng lập trình, xây dựng phần mềm hướng đối tượng, lập trình web sử dụng công nghệ ASP.NET MVC 4, entity framework,… của microsoft . Tìm hiểu và sử dụng được một hoặc một vài hệ quản trị cơ sở dữ liệu phổ biến như: SQL Server, MySQL…

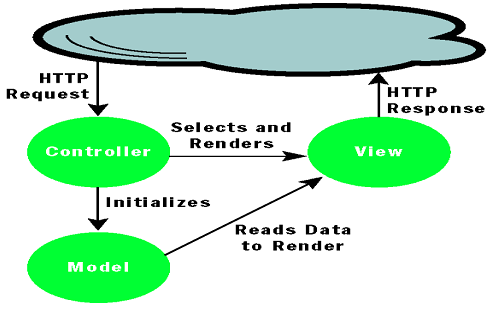
# 2. Chương 2 - Giới thiệu về công nghệ được dùng trong tiểu luận

1. **Các khái niệm chính của mô hình MVC trong ASP.NET.**
   * **ASP.NET là gì?**

ASP.NET là viết tắt của Active Server Pages .NET (.NET ở đây là .NET framework). ASP.NET là một công nghệ dùng để phát triển các ứng dụng về mạng hiện nay cũng như trong tương lai.

* + **MVC là gì?**

MVC viết tắt của các chữ cái đầu của Models, Views, Controllers. MVC chia giao diện UI thành 3 phần tương ứng: đầu vào của controller là các điều khiển thông qua HTTP request, model chứa các miền logic, view là những thứ được sinh ra trả về cho trình duyệt.



*Hình 1. Mô hình MVC cơ bản*

* + **Controller là gì?**

Controller chịu trách nhiệm điều khiển các tương tác của người dùng bên trong ứng dụng MVC. Controller sẽ quyết định cái gì được trả về cho người dùng khi người dùng tạo ra một request trên browser.

* + **View là gì?**

Một view chứa các thẻ HTML và được trả về cho browser.

* + **Model là gì?**

Một model trong ứng dụng ASP.NET MVC chứa tất cả các nghiệp vụ logic mà không có trong controllers và views. Models chứa tất cả các tầng truy xuất dữ liệu logic và tầng nghiệp vụ logic.

* + **URL Routing trong MVC.**

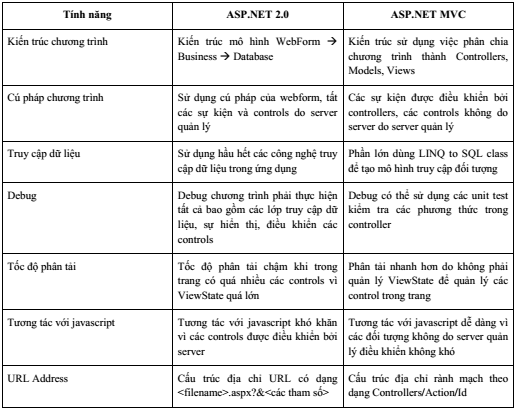
ASP.NET MVC Framework có một hệ thống định tuyến URL (URL Routing System) linh hoạt cho phép xác định các quy tắc ánh xạ địa chỉ URL bên trong ứng dụng. Một hệ thống định tuyến có 2 mục đích:

* Xây dựng một tập hợp các URL đi vào ứng dụng và định tuyến chúng tới các Controller và thực thi các phương thức Action để xử lý.
* Xây dựng các URL gửi đi mà có thể gọi ngược trở lại Controllers/Actions (ví dụ: form posts, liên kết anchor hay các lời gọi AJAX).

Sử dụng các quy tắc ánh xạ URL để điều khiển URL đi vào và đi ra nhằm tăng tính mềm dẻo cho việc lập trình ứng dụng, nghĩa là nếu muốn thay đổi cấu trúc URL (ví dụ /Catalog thành /Products) có thể thay đổi một tập hợp các quy tắc ánh xạ mức ứng dụng mà không cần phải viết lại mã lập trình bên trong Controllers và Views.

1. **Một vài đặc điểm của ASP.NET MVC.**
   * Tách rõ ràng các mối liên quan, mở khả năng test TDD (test driven developer). Có thể test unit trong ứng dụng mà không cần phải chạy Controllers cùng với tiến trình của ASP.NET và có thể dùng bất kỳ một unit testing framework nào như NUnit, MBUnit, MS Test, v.v…
   * Có khả năng mở rộng, mọi thứ trong MVC được thiết kế cho phép dễ dàng thay thế/tùy biến ( ví dụ: có thể lựa chọn sử dụng engine view riêng, routing policy, parameter serialization, v.v…).
   * Bao gồm một ánh xạ URL mạnh mẽ cho phép xây dựng ứng dụng với những URL sạch, các URL không cần cs phần mở rộng ( ví dụ: có thể ánh xạ địa chỉ /Products/Edit/4 để thực hiện hành động “Edit” của lớp điều khiển ProductControllers hoặc ánh xạ địa chỉ /Blog/SomeTopic để thực hiện hành động “Display Topic” của lớp điều khiển BlogEngineController).
   * ASP.NET MVC Framework cũng hỗ trợ những file ASP.NET như .ASPX, .ASCX và .Master, đánh dấu các tập tin này như một “view template” ( có thể dễ dàng dùng các tính năng của ASP.NET như lồng các trang Master, <%= %> snippets, mô tả server controls, template, data-binding, localization, v.v… ). Tuy nhiên sẽ không còn postback và interactive back server và thay vào đó là interactive end-user tới một Controller class (không còn viewstate, page lifecycle)
   * ASP.NET MVC Framework hỗ trợ đầy đủ các tính năng bảo mật của ASP.NET như forms/windows authenticate, URL authorization, membership/roles, output và data caching, session/profile state, configuration system, provider architecture v.v.
2. **Sự khác biệt với webform.**

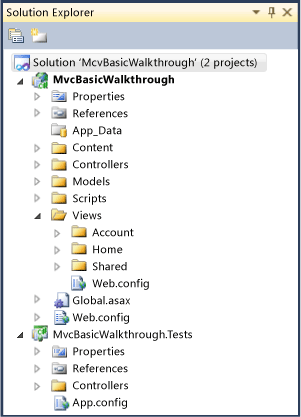
* ASP.NET WebForm sử dụng ViewState để quản lý, các trang ASP.NET đều có lifecycle, postback và dùng các web controls, các events để thực hiện các hành động cho UI khi có sự tương tác với người dùng nên hầu hết ASP.NET WebForm xử lý chậm.
* ASP.NET MVC Framework chia ra thành 3 phần: Models, Views, Controllers. Mọi tương tác của người dùng với Views sẽ được thực hiện hành động trong Controllers, không còn postback, không còn lifecycle không còn events.
* Việc kiểm tra ( test ), gỡ lỗi (debug) với ASP.NET đều phải chạy tất cả các tiến trình của ASP.NET và mọi sự thay đổi ID của bất kỳ controls nào cũng ảnh hưởng đến ứng dụng. Đối với ASP.NET MVC Framework thì việc có thể sử dụng các unit test có thể thẩm định rất dễ dàng các Controller thực hiện như thế nào.



*Hình 2: So sánh ASP.NET và ASP.NET MVC*

1. **Chuẩn bị môi trường phát triển ứng dụng ASP.NET MVC 4.**
   * Để phát triển một ứng dụng ASP.NET MVC 4, máy tính cần cài đặt công cụ microsoft visual studio có tích hợp công nghệ MVC của microsoft. Với các phiên visual studio trước 2012 (2010, 2008 …) cộng nghệ MVC 4 chưa được tích hợp sẵn. Do đó, cần tải và cài đặt MVC 4 tại địa chỉ: <http://www.asp.net/mvc/mvc4>.
   * Nếu đang cài đặt visual studio phiên bản từ 2012 trở đi thì MVC 4 đã được tích hợp sẵn, bạn không cần tải và cài đặt thêm.
   * Ngoài ra, bạn cần phải bật dịch vụ IIS (Internet Information Service) trong window để có thể chạy các ứng dụng ASP.NET. Để bật dịch vụ IIS, thực hiện như sau:
     + Start menu\Control Panel.
     + Chọn Program\Programs and Features.
     + Trong menu bên tay trái của hộp thoại Programs and Features chọn “**Turn windows feature on or off**”.
     + Trong hộp thoại Windows features, tích chọn vào dịch vụ Internet Information Service. Sau đó, click OK. Quá trình chạy hoàn tất thì dịch vụ IIS đã được mở trên máy tính và có thể bắt đầu xây dựng ứng dụng ASP.NET.
2. **Cấu trúc của ứng dụng ASP.NET MVC.**

Một ứng dụng ASP.NET MVC 4 trong microsoft visual studio 2012 khi được tạo ra về cơ bản sẽ có cấu trúc giống như hình sau:



*Hình 3: Cấu trúc ứng dụng ASP.NET MVC 4.*

Ứng dụng được tạo mặc định sẽ có các thư mục sau:

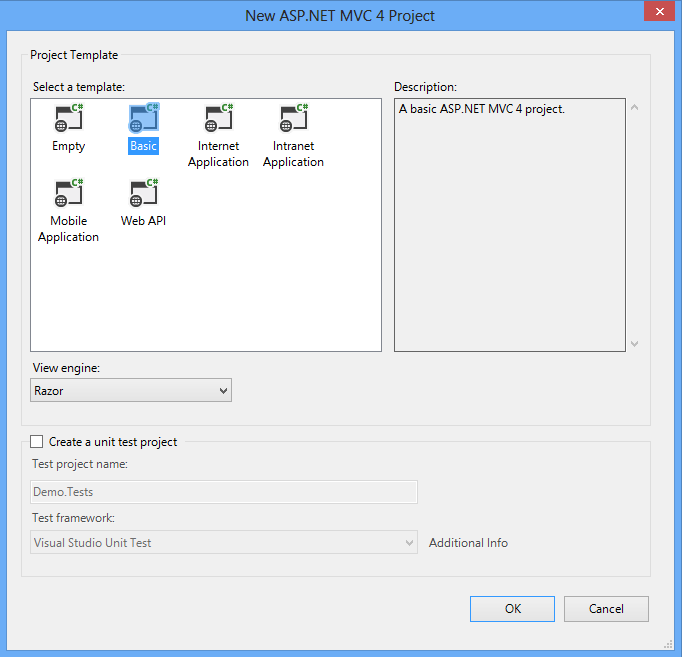
* App\_Data: là nơi chứa cơ sở dữ liệu vật lý của ứng dụng.
* Content: là nơi chứa tài nguyên của website như hình ảnh hay các file css.
* Controllers: nơi chứa các file controller của ứng dụng, các file này được đặt tên với hậu tố “Controller”.
* Models: thông thường chứa các lớp có mã để định nghĩa các đối tượng hay các giao tiếp logic với cơ sở dữ liệu.
* Scripts: là nới chứa các file javascript của website.
* Views : là nới chứa các view của ứng dụng tương ứng với các controller hay các action của controller. Mỗi controller sẽ có một folder tương ứng trong Views, mỗi folder đó lại chứa các file tương ứng là các view của các action trong controller đó.

1. **Hoạt động của một ứng dụng ASP.NET MVC.**

* Trong ASP.NET MVC, url dùng để request tới ứng dụng được ánh xạ theo controller và các action trong controller đó. Không giống với ASP.NET, việc ánh xạ url được thực hiện dựa trên lưu trữ vật lý của các file mã nguồn.
* Do đó, khi browser gửi một request tới server, trong công nghệ mvc, url được request sẽ được định tuyến (routing) bằng tập hợp các luật định tuyến đã được thiết lập sẵn trong ứng dụng. Việc định tuyến sẽ xác định ra các thành phần cần thiết từ url như : controller nào được gọi, action nào của controller đó sẽ được thực hiện hay những tham số cần thiết cho action.
* Khi đã xác định được controller, action, … ứng dụng sẽ thực thi action tương đó. Nếu trong action đó, đơn thuần không có xử lý dữ liên quan tới model (cơ sở dữ liệu) thì kết quả sẽ được trả về ngay cho view. View sẽ nhận các thông tin và trả về cho trình duyệt để hiện thị cho người sử dụng.
* Nếu trong action có thực hiện tương tác với model, cơ sở dữ liệu thì trong action sẽ tiến hành gọi các phương thức xử lý dữ liệu (thêm, xóa, sửa,…) trong các lớp model đã có sẵn để truy xuất dữ liệu. Sau đó, dữ liệu có được cũng sẽ được trả về cho view. View tiếp tục gửi trả dữ liệu này về cho browser để hiển thị cho người sử dụng.

1. **Các bước để tạo một ứng dụng ASP.NET MVC 4 với Visual Studio 2012.**

* Bước 1 : thực hiện đầy đủ các bước chuẩn bị môi trường đã được mô tả ở trên.
* Bước 2 : Trong MS Visual Studio 2012, chọn: **File → New Project → Visual C# → Web → ASP.NET MVC 4 Web Application**. Sau khi đặt tên cho project và định thư mục lưu trữ xong, click OK để tạo mới ứng dụng.
* Bước 3: Xuất hiện hộp thoại New ASP.NET MVC 4 Project. Tiến hành lựa chọn template cho project. (Hình 3)
  + Empty: một ứng dụng hoàn toàn mới.
  + Basic: một ứng dụng mới với các folder rỗng tương ứng với cấu trúc của ứng dụng MVC 4.
  + Internet Application: một ứng dụng MVC 4 mới với một vài controller, model, view mẫu đã được tạo sẵn.
* Bước 4: Tiếp tục tiến hành lựa chọn view engine cho ứng dụng (Hình 3):
  + ASPX: các view được viết theo mã ASP.NET.
  + Razor: các view được viết theo mã Razor.
* Bước 5: Tick chọn Create a unit test project nếu muốn tạo unit test cho ứng dụng. (Hình 3)

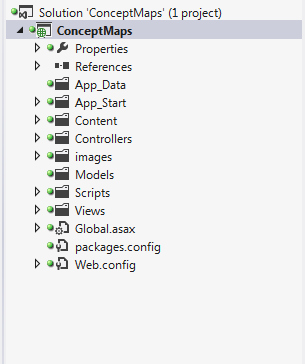
****

*Hình 4: Hộp thoại New ASP.NET MVC 4 Project*

* Bước 6: Click OK để tạo ứng dụng.

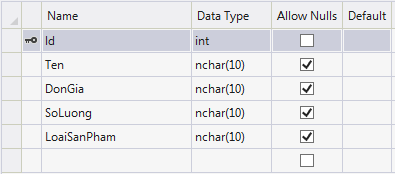
1. **Ví dụ minh họa :**

* Bước 1 : tạo một project theo hướng dẫn ở phần 7. Đặt tên project là Test. Project template chọn Basic. View engine chọn razor view engine. Không chọn tạo unit test cho project. Project mới tạo sẽ có cấu trúc như sau (hình 4):



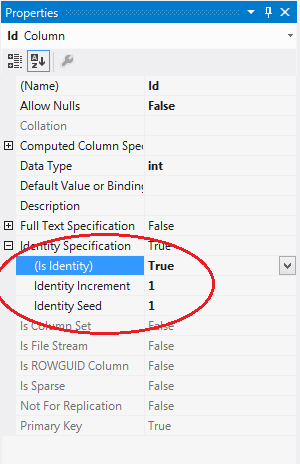
*Hình 5: Cấu trúc website ASP.NET MVC 4.*

* Bước 2: Tạo dữ liệu mẫu:
  + Click phải vào folder App\_Data → Add → New Item.
  + Trong hộp thoại hiện ra chọn Visual C# → Data → SQL Server Database và đặt tên là “Database”.
  + Double click vào file “Database.mdf” mới tạo ra. Trong cửa sổ xuất hiện bên tay trái, double click vào “Database.mdf”, tiếp tục click phải vào Tables và chọn New Query.
  + Copy đoạn mã sql sau vào cửa sổ mới hiện ra, nhấn tổ hợp ctrl + a, click phải và chọn Excute.
  + Hai bảng dữ liệu sẽ được tạo ra như sau:
    - Bảng SanPham:



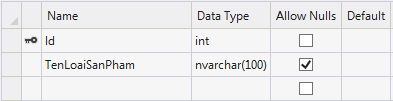
*Hình 6: Bảng SanPham*

Click phải vào trường Id→Properties, trong hộp thoại mới xuất hiện, thiết lập như sau:



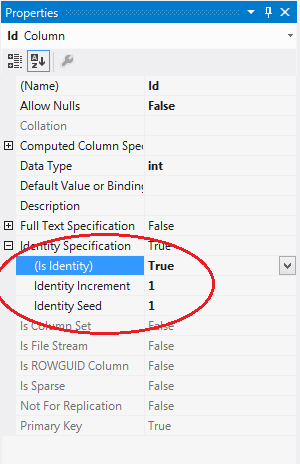
*Hình 7: Thiết lập trường Id của bảng SanPham*

* + - Bảng LoaiSanPham:



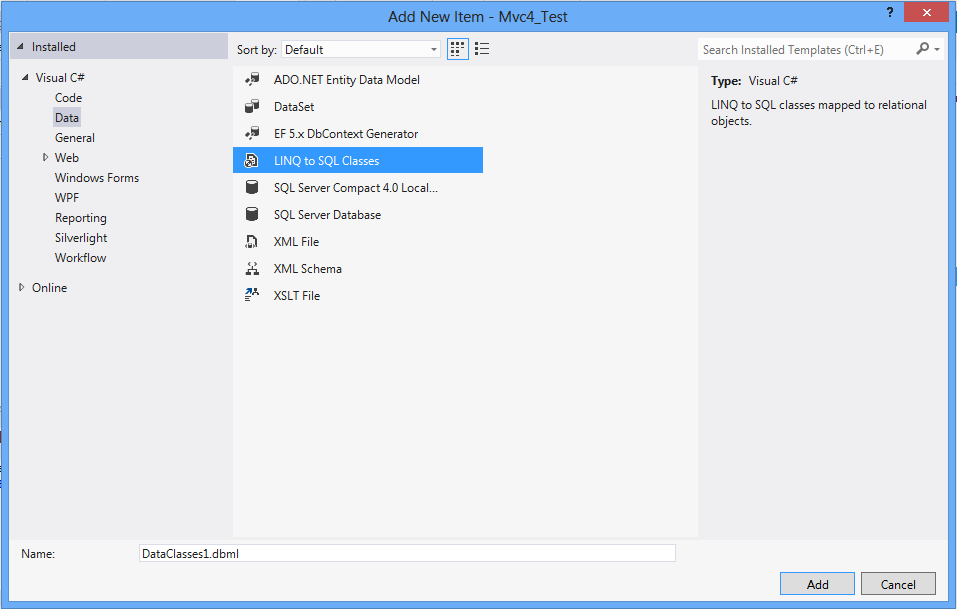
*Hình 8: Bảng LoaiSanPham*

Click phải vào trường Id→Properties, trong hộp thoại mới xuất hiện, thiết lập như sau:



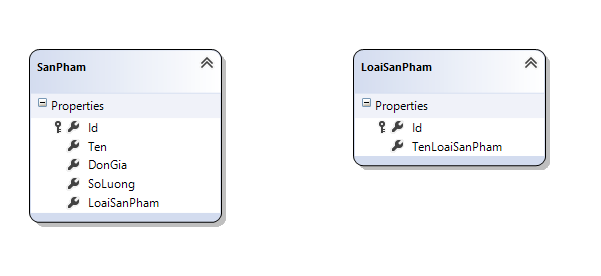
*Hình 9: Thiết lập trường Id của bảng LoaiSanPham*

* Bước 3: Tạo một lớp LINQ to SQL để làm model trong ứng dụng:
  + Trong Solution explorer, click phải vào folder Models → Add → New Item → Visual C# → Data → LINQ to SQL Classes. Đặt tên lớp này là: “DataClasses.dbml”. Chọn Add để thêm mới.



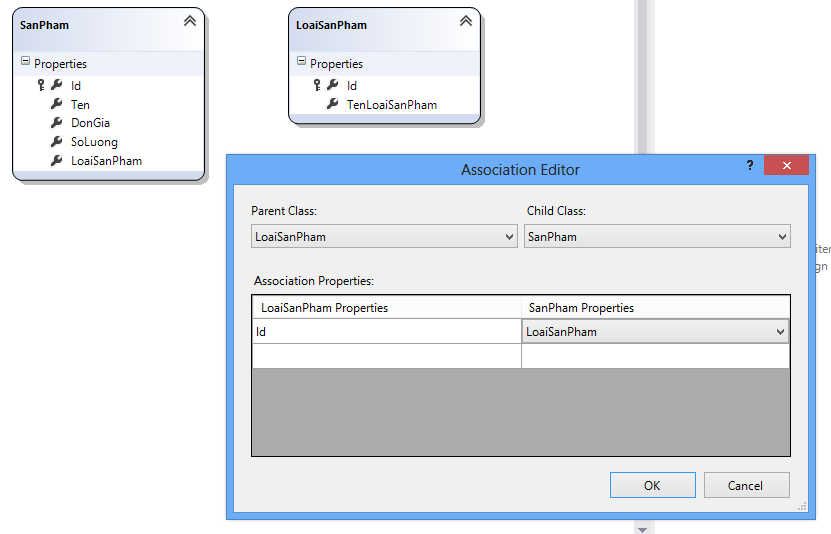
*Hình 10: Tạo LINQ to SQL Class.*

* + Từ Server explorer, kéo thả 2 bảng SanPham và LoaiSanPham vào file DataClass.dbml.



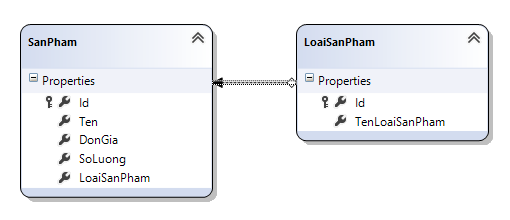
*Hình 11: Kéo thả 2 bảng SanPham và LoaiSanPham*

* + Trong file DataClass.dbml click phải chọn Add Association để tạo liên kết giữa 2 bảng.



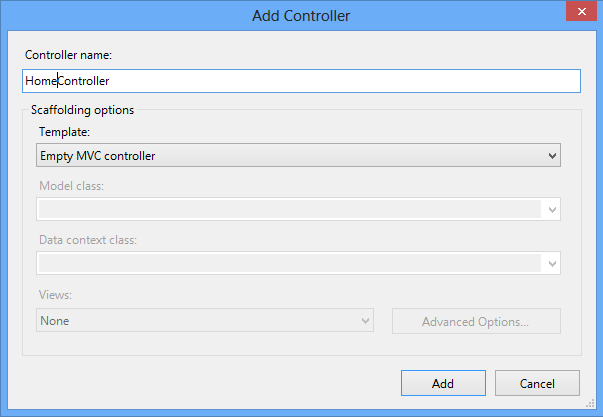
*Hình 12: Thiết lập liên kết giữa 2 bảng*

* + Kết quả được như sau:



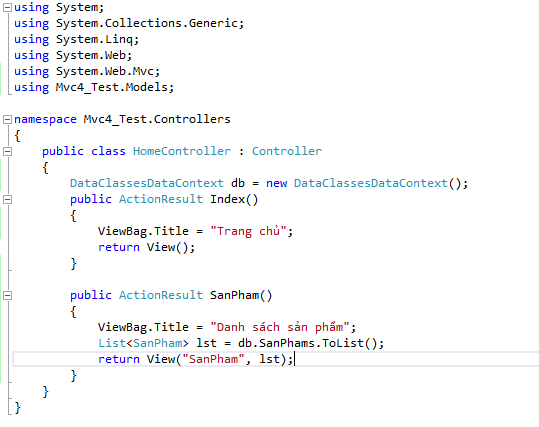
*Hình 13: Kết quả tạo cơ sở dữ liệu mẫu*

* + Thêm dữ liệu mẫu vào các bảng SanPham và LoaiSanPham
* Bước 4: Tạo controller mới
* Click phải vào folder Controllers → Add → Controller. Chọn như hình sau:



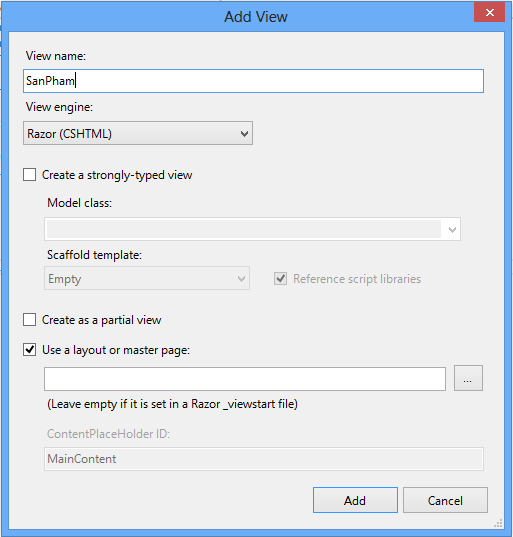
*Hình 14: Thêm controller mới*

* Code của HomeController.cs :



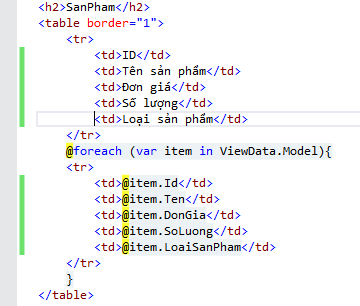
*Hình 15: Code của controller HomeController.cs*

* Bước 5 : Tạo View SanPham để hiện thị thông tin danh sách sản phẩm
  + Click phải vào folder Home nằm trong folder Views → Add → View. Đặt tên view là “SanPham” (tương ứng với action SanPham() trong Home controller). Chọn View engine là: Razor (CSHTML). Như hình sau:



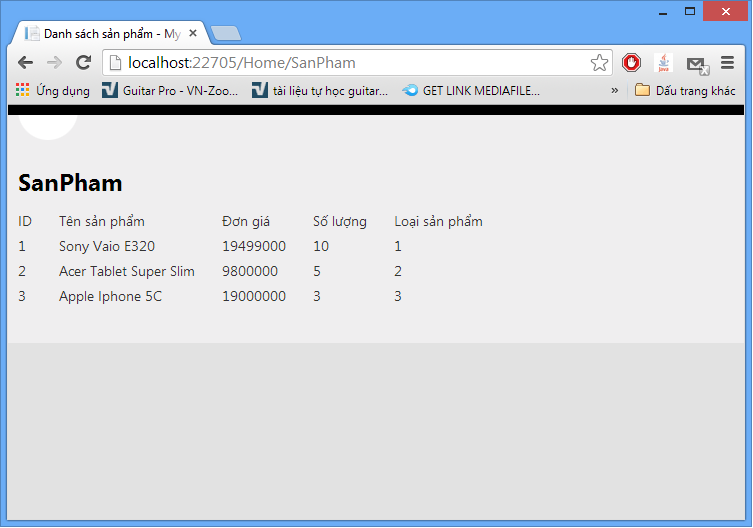
*Hình 16: Thêm View mới*

* + Viết code cho View SanPham trong file SanPham.cshtml vừa mới tạo như sau:



*Hình 17: Code của View SanPham.cshtml*

* Bước 6: Kết quả chạy chương trình:



*Hình 18: Kết quả thực hiện chương trình.*

# 3. Chương 3 - Thiết kế giao diện

# 4. Chương 4 - Thiết kế chương trình và CSDL

# 5. Chương 5 - Cài đặt & Demo

# 6. Chương 6 - Kết luận và hướng phát triển