**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**----🙣🕮🙡----**



**BÁO CÁO TIỂU LUẬN CHUYÊN NGÀNH**

**ĐỀ TÀI: Tìm hiểu Microsoft ASP.NET Core 2, Entity Framework core và xây dựng website quản lý website bán mĩ phẩm**

**GVHD: TS. Trần Lê Minh Sang**

**SVTH : Nguyễn Đức Thịnh 15110132**

**Vương Văn Hậu 15110044**

**Tp. Hồ Chí Minh, Tháng 12 năm 2018**

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

GIẢNG VIÊN HƯỚNG DẪN

Ký tên

# NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

GIẢNG VIÊN PHẢN BIỆN

Ký tên

# LỜI CẢM ƠN

Thực tế luôn cho thấy, sự thành công nào cũng đều gắn liền với những sự hỗ trợ, giúp đỡ của những người xung quanh dù cho sự giúp đỡ đó là ít hay nhiều, trực tiếp hay gián tiếp. Trong suốt thời gian từ khi bắt đầu làm tiểu luận đến nay, nhóm chúng em đã nhận được sự quan tâm, chỉ bảo, giúp đỡ của thầy cô và bạn bè xung quanh. Với tấm lòng biết ơn vô cùng sâu sắc, em xin gửi lời cảm ơn chân thành nhất từ đáy lòng đến quý Thầy Cô của trường đã cùng dùng những tri thức và tâm huyết của mình để có thể truyền đạt cho chúng em trong vốn kiến thức quý báu suốt thời gian học tập tại trường. Đặc biệt, em xin chân thành cảm ơn cô Trần Lê Minh Sang đã tận tâm chỉ bảo hướng dẫn em qua từng buổi học, từng buổi nói chuyện, thảo luận về đề tài nghiên cứu. Nhờ có những lời hướng dẫn, dạy bảo đó, bài luận văn này của em đã hoàn thành một cách xuất sắc nhất. Một lần nữa, em xin gửi lời cảm ơn chân thành đến cô. Bài tiểu luận được thực hiện trong 3 tháng. Ban đầu em còn bỡ ngỡ vì vốn kiến thức của em còn hạn. Do vậy, không tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của quý Thầy Cô và các bạn học cùng lớp bài luận được hoàn thiện hơn. Nhóm chúng em xin chân thành cảm ơn sự giúp đỡ của quý thầy cô trong suốt thời gian qua!

TP.Hồ Chí Minh, ngày 12 tháng 12 năm 2018

# LỜI NÓI ĐẦU

Khi nói tới [Search Engine](https://tamnguyen.com.vn/Tim-hieu-ve-he-thong-may-tim-kiem-Search-Engine.html) , ta thường nghĩ ngay đến các dịch vụ nổi tiếng như Google Search, Yahoo! Search hay MSN Search, v.v…Tuy nhiên, bộ phận tìm kiếm trong một website cụ thể cũng được coi là Search Engine.

Internet chứa hầu như tất cả những thông tin liên quan tới mọi lĩnh vực, mọi ngõ ngách trong cuộc sống. Nhưng nó rất rộng, rộng đến mức gần như không ai có thể kiểm soát được. Diện mạo của Internet lại thay đổi quá nhanh chóng và mạnh mẽ. Hạt nhân của Internet là Word Wide Web, với số lượng lên tới hàng chục tỉ trang, được lưu trữ trong hàng triệu máy chủ đặt khắp nơi trên toàn thế giới.

Có thể ví Internet như một biển dữ liệu khổng lồ, với muôn vàn những viên ngọc quí nằm giữa các hạt sạn. Trong đời sống hàng ngày, nhu cầu tìm kiếm thông tin đóng vai trò vô cùng to lớn, và một trong những vấn đề bức thiết nhất của công nghệ hiện nay là làm sao "đãi cát tìm vàng", khai thác nguồn tài nguyên này một cách hợp lí, đem lại lợi ích tốt nhất cho con người.

Tìm kiếm thông tin trên mạng Internet quả thật là một thách thức lớn lao. Nó không giống như việc bới các hạt đỗ đen nằm lẫn lộn trong thùng gạo, bởi dữ liệu trên mạng Internet do con người đưa vào, chúng cũng có cấu trúc và tổ chức xác định (mặc dù thiếu tính nhất quán), trong khi đó thì các hạt đỗ đen lại nằm rải rác và lộn xộn, không có một vị trí hay qui luật nào. Tuy nhiên, bài toán tìm kiếm khó hơn bài toán nhặt đỗ đen rất nhiều. Muốn tìm tất cả các hạt đỗ đen, bạn đơn giản chỉ cần thiết kế một cái sàng hình cầu đủ lớn để có thể đổ cả thùng gạo vào đó, với những chiếc lỗ có kích thước phù hợp sao cho hạt gạo chui lọt còn hạt đỗ đen thì không, và quay đủ số vòng để tất cả các hạt gạo đều có cơ hội bay ra ngoài. Việc tìm kiếm thông tin trên Internet lại hoàn toàn khác.

Có tới hàng chục tỉ trang Web tràn ngập trên mạng Internet (gấp nhiều lần số hạt gạo trong thùng), và vấn đề là làm sao đưa ra những gì ta muốn thu thập sao cho đồng thời thỏa mãn hai tiêu chí: chính xác và nhanh chóng. Hơn thế nữa, người dùng cũng không đủ kiên nhẫn để ngồi duyệt qua tất cả các trang web chứa thông tin cần tìm (anh ta cũng không nhất thiết phải đếm từng hạt đỗ đen, tuy nhiên nếu xét trên tiêu chí dinh dưỡng thì đa phần những hạt đỗ đen đều giống nhau, do đó hạt nào cho vào nồi trước cũng không quan trọng). Trên thực tế, người dùng hiếm khi vào quá mười trang web kết quả, và vì thế, một yêu cầu khó khăn nữa cần giải quyết, đó là: những gì phù hợp nhất phải được đặt lên hàng đầu.

Trước đây, người ta thường chia dữ liệu cần lưu trữ làm nhiều mục, đến lượt các mục con này lại được chia nhỏ hơn. Người dùng tìm kiếm thông tin thông qua việc duyệt qua liên kết giữa các mục. Tuy nhiên, những chủ đề được nêu trong Internet đã rộng lớn đến nỗi sự phân chia này trở nên cực kì cồng kềnh và bất tiện. Ngày nay, hầu hết mọi người đều sử dụng Search Engine để tìm kiếm thông tin trên mạng Internet.

Đối với mỗi Search Engine (Google, Yahoo, MSN, v.v…), người dùng truy vấn tìm kiếm (hay nói đơn giản hơn là nhập vào một số từ khóa liên quan đến chủ đề cần tìm), và nhận được một danh sách các trang kết quả (thông thường là những trang web chứa các từ khóa cần tìm kiếm), được sắp xếp theo một tiêu chí nào đó. Những tiêu chí này đều nhằm mục đích "đưa ra kết quả phù hợp nhất với yêu cầu tìm kiếm".

# DANH MỤC HÌNH

[Hình I-1: Màn hình cài đặt [1] 17](#_Toc532726388)

[Hình I-2: Màn hình chạy cài đặt [2] 18](#_Toc532726389)

[Hình I-3: Màn hình chạy cài đặt xong [3] 18](#_Toc532726390)

[Hình I-4: Màn hình chờ chạy [4] 19](#_Toc532726391)

[Hình I-5: Màn hình tạo newproject [5] 20](#_Toc532726392)

[Hình I-6: Màn hình tạo app [6] 21](#_Toc532726393)

[Hình I-7: Màn hình chờ app hiện [7] 21](#_Toc532726394)

[Hình I-8: Tạo phương thức Get() [8] 23](#_Toc532726395)

[Hình I-9:Trả về giá trị id cho Get() [9] 23](#_Toc532726396)

[Hình I-10: Hàm GetProduct() [10] 24](#_Toc532726397)

[Hình I-11:Hàm Post() [11] 24](#_Toc532726398)

[Hình I-12: So sánh component [12] 26](#_Toc532726399)

[Hình I-13: So sánh directive [13] 28](#_Toc532726400)

[Hình I-14: So sánh Input Model [14] 30](#_Toc532726401)

[Hình I-15: So sánh Filter [15] 31](#_Toc532726402)

[Hình I-16: So sánh Providers/Services/Factories/Values/Constants [16] 32](#_Toc532726403)

[Hình I-17: Trang tải Nodejs[17] 32](#_Toc532726404)

[Hình I-18:Trang tải Angular [18] 33](#_Toc532726405)

[Hình I-19: Cài đặt Angular [19] 34](#_Toc532726406)

[Hình II-1: Google Search[20] 36](#_Toc532726407)

[Hình II-2: Duck Duck Go Search[21] 37](#_Toc532726408)

[Hình II-3: Bing Search[22] 38](#_Toc532726409)

[Hình II-4: Dogpile Search[23] 39](#_Toc532726410)

[Hình II-5:Mô hình đánh chỉ mục của Lucene [24] 41](#_Toc532726411)

[Hình II-6:Cấu trúc của chỉ mục nghịch đảo[25] 44](#_Toc532726412)

[Hình II-7:Cài đặt của Lucene[26] 45](#_Toc532726413)

[Hình II-8:Cài đặt của Lucene hoàn thành[27] 46](#_Toc532726414)

[Hình II-9:Lớp khởi tạo dữ liệu mẫu[28] 46](#_Toc532726415)

[Hình II-10:Lớp Sampledata[29] 47](#_Toc532726416)

[Hình II-11:Lớp LuceneSearch [30] 47](#_Toc532726417)

[Hình II-12:Hàm \_luceneDir thêm vào LuceneSearch [31] 48](#_Toc532726418)

[Hình II-13:Hàm \_addToLucene[32] 49](#_Toc532726419)

[Hình II-14:Hàm AddUpdateLuceneIndex[33] 49](#_Toc532726420)

[Hình II-15:Phương thức khác sẽ được thêm[34] 50](#_Toc532726421)

[Hình II-16:Sử dụng 2 phương thức[35] 50](#_Toc532726422)

[Hình II-17:Cung cấp duy nhất 1 bản ghi[36] 50](#_Toc532726423)

[Hình II-18:Lớp ClearLuceneIndexRecord[37] 51](#_Toc532726424)

[Hình II-19: Lớp ClearLuceneIndex [38] 51](#_Toc532726425)

[Hình II-20: Hàm tối ưu trong 1 thời gian [39] 52](#_Toc532726426)

[Hình II-21: Ánh xạ SampleData [40] 53](#_Toc532726427)

[Hình II-22: Hàm \_mapLuceneDocument [41] 53](#_Toc532726428)

[Hình II-23: Hàm định dạng truy vấn [42] 54](#_Toc532726429)

[Hình II-24: Hàm tìm kiếm chính [43] 55](#_Toc532726430)

[Hình II-25: Thêm trường mới vào chỉ mục [44] 55](#_Toc532726431)

[Hình II-26: Hàm search [45] 56](#_Toc532726432)

[Hình II-27: Hàm SearchDefault [46] 57](#_Toc532726433)

[Hình II-28: Hàm trả về toàn bộ chỉ mục tiềm kiếm [47] 57](#_Toc532726434)

[Hình III-1: Cấu trúc thư mục 58](#_Toc532726435)

[Hình III-2: Thư mục Controller 59](#_Toc532726436)

[Hình III-3: Thư mục Models 59](#_Toc532726437)

[Hình III-4: Thư mục Views 59](#_Toc532726438)

[Hình III-5: Thư mục Lucene 59](#_Toc532726439)

[Hình III-6: Màn hình sau khi chạy demo 60](#_Toc532726440)

[Hình III-7: Màn hình tìm kiếm sản phẩm 60](#_Toc532726441)

[Hình III-8: Sử dụng thanh search Lucene 61](#_Toc532726442)

[Hình III-9: Kết quả tìm kiếm. 61](#_Toc532726443)

# MỤC LỤC

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN 1](#_Toc532726512)

[NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN 2](#_Toc532726513)

[LỜI CẢM ƠN 3](#_Toc532726514)

[LỜI NÓI ĐẦU 4](#_Toc532726515)

[DANH MỤC HÌNH 6](#_Toc532726516)

[MỤC LỤC 9](#_Toc532726517)

[CHƯƠNG I: GIỚI THIỆU 11](#_Toc532726518)

[1. Mục tiêu 11](#_Toc532726519)

[2. Lý do 12](#_Toc532726520)

[3. Phạm vi 12](#_Toc532726521)

[CHƯƠNG II: NỘI DUNG 13](#_Toc532726522)

[PHẦN I. CƠ SỞ LÝ THUYẾT VỀ ASP.NET VÀ ANGULAR 4 13](#_Toc532726523)

[1. ASP.NET 13](#_Toc532726524)

[2. ANGULAR 4 25](#_Toc532726525)

[PHẦN II: TÌM HIỀU VỀ SEARCH ENGINE VÀ CÔNG CỤ TÌM KIẾM LUCENE VÀ LUCENE.NET 35](#_Toc532726526)

[1. Lịch sử của SEARCH ENGINE 35](#_Toc532726527)

[*2.* Khái niệm về SEARCH ENGINE. 35](#_Toc532726528)

[3. Một số SEARCH ENGINE hiện nay 36](#_Toc532726529)

[4. Tìm hiểu về Lucene 40](#_Toc532726530)

[CHƯƠNG III: ỨNG DỤNG KIẾN THỨC VÀO WEBSITE MỸ PHẨM 58](#_Toc532726531)

[1. Cấu trúc thư mục 58](#_Toc532726532)

[2. Dùng mô hình MVC 59](#_Toc532726533)

[3. Sản Phẩm sau khi được chạy 60](#_Toc532726535)

[4. Sử dụng công cụ tìm kiếm Lucene.net 61](#_Toc532726536)

[CHƯƠNG IV: KẾT LUẬN 62](#_Toc532726537)

[TÀI LIỆU THAM KHẢO 63](#_Toc532726538)

# CHƯƠNG I: GIỚI THIỆU

## Mục tiêu

Xây dụng trang web bán mỹ phẩm có back-end sẽ sử dụng ASP.NET và front-end sẽ áp dụng Angular 4.

Trong bài này cũng sẽ nghiên cứu về Search Engine. Mục tiêu chính nhằm nhằm:

- Giới thiệu tổng quan của một Search Engine.

- Nghiên cứu chi tiết các thành phần của Search Engine (Lucene và Luncene.net).

- Giúp có những kiến thức cơ bản để hiểu được cấu trúc phức tạp và rộng của Search Engine và đi chi tiết vào Lucene.

- Chuẩn bị nền tảng cho cách xây dựng một trang web có sử dụng Search Engine (Lucene).

Những kiến thức liên quan tới Search Engine rất rộng và tổng hợp, bao gồm thuật toán, cấu trúc dữ liệu, cơ sở dữ liệu, các hệ thống phân tán, tính toán song song, tổ chức file, data mining,v.v… cũng như những vấn đề có liên quan tới toán học. Do đó, việc tìm hiểu Search Engine sẽ hứa hẹn rất nhiều điều thú vị, nếu chỉ đề cập sơ sài, ta sẽ bỏ qua một số chủ đề hấp dẫn. Đây cũng là dịp để chúng ta cùng nhau kiểm tra cũng như cập nhật vốn kiến thức của mình.

Tự xây dựng một Search Engine là một "thách thức" không nhỏ và rất đáng để xem xét. Tất nhiên sản phẩm của những sinh viên không có ý nghĩa gì khi so sánh với Google Search hay Yahoo! Search, và tất nhiên cũng mang rất ít giá trị về mặt thương mại cũng như thực tiễn (phải nói là không có thì đúng hơn). Nhưng đối với bản thân mỗi người học thì giá trị học hỏi và kiến thức là rất to lớn, những mảng đề tài liên quan đến Search Engine là rất nhiều. Nghiên cứu này như nền tảng để có thể hiểu cặn kẽ cấu tạo của Internet Search Engine. Việc hiểu cấu trúc của Search Engine cũng giúp ích trong việc lập trình xây dựng các trang Web (mặc dù công việc đối với trang Web đơn giản hơn nhiều).

Trong giáo trình công nghệ thông tin cũng như khoa học máy tính ở các trường đại học Việt Nam hiện nay không có môn học về Information Retrieval. Việc tìm hiểu Search Engine cũng cho ta một số kiến thức về lĩnh vực này. Hơn nữa, đây là cơ hội rất tốt để thực hành các tri thức ta đã thu thập được trong những năm học đại học.

## Lý do

Trước khi có sự thống trị về lĩnh vực tìm kiếm trên internet của Google, những người dùng trước thời mà ngành công nghệ thông tin bùng nổ mạnh mẽ chắc hẳn vẫn còn nhớ tới Altavista, Lycos, Yahoo,…đó chính là những nhà cung cấp dịch vụ tìm kiếm thông tin trên mạng internet. Cùng với chính đội ngũ chuyên gia phân loại, tóm tắt và đánh giá nội dung hùng hậu (đánh giá chất lượng thông tin thủ công). Và việc đánh giá theo cách đó tất nhiên là sẽ tốt hơn nhưng với sự phát triển mạnh mẽ về hạ tầng phần cứng và viễn thông, cũng như xu hướng đang phát triển mạnh về thông tin trên internet với lượng tài nguyên thông tin khổng lồ vào những năm đầu 2000 thì các cỗ máy tìm kiếm bán tự động đã không thể đáp ứng được nhu cầu tìm kiếm của người sử dụng, đó chính là lý do tại sao nhóm muốn tạo ra 1 trang web bán hàng mỹ phẩm mà back-end sẽ là ASP.NET và front-end sẽ là Angular 4 sẽ được dùng để tạo nên trang. Thêm vào đó là áp dụng cả công cụ tìm kiếm để phát triển mạnh hơn trong tương lai gần.

## Phạm vi

Áp dụng được những kiến thức đã tìm hiểu và nghiên cứu và tạo ra 1 trang web bán mỹ phẩm. Back-end sẽ là ASP.NET và front-end sẽ là Angular 4 sẽ được dùng để tạo nên trang web bán hàng. Ngoài ra, trang còn sẽ ứng dụng thêm phần tìm kiếm sản phẩm bằng công cụ Lucene.net mả nhóm đã tìm hiểu và nghiên cứu thêm để có thể tối ưu hóa việc tìm kiếm sản phẩm theo nhu cầu của khách hàng.

# CHƯƠNG II: NỘI DUNG

## PHẦN I. CƠ SỞ LÝ THUYẾT VỀ ASP.NET VÀ ANGULAR 4

### ASP.NET

#### Giới thiệu.

ASP.NET là một nền tảng phát triển web, cung cấp một mô hình lập trình, một cơ sở hạ tầng phần mềm toàn diện và các dịch vụ khác nhau cần thiết để xây dựng các ứng dụng web mạnh mẽ cho PC, cũng như các thiết bị di động.

ASP.NET hoạt động trên đầu trang của giao thức HTTP và sử dụng các lệnh và chính sách HTTP để thiết lập giao tiếp và hợp tác song phương giữa các trình duyệt với máy chủ.

ASP.NET là một phần của nền tảng Microsoft .Net. Các ứng dụng ASP.NET được biên dịch mã, được viết bằng cách sử dụng các thành phần mở rộng và có thể tái sử dụng hoặc các đối tượng có trong khung .Net. Các mã này có thể sử dụng toàn bộ hệ thống phân cấp của các lớp trong .Net framework.

Mã ứng dụng ASP.NET có thể được viết bằng bất kỳ ngôn ngữ nào sau đây:

* C #
* Visual Basic.Net
* Jscript
* J #

ASP.NET được sử dụng để tạo ra các ứng dụng web theo hướng dữ liệu, tương tác qua internet. Nó bao gồm một số lượng lớn các điều khiển như hộp văn bản, nút và nhãn để lắp ráp, định cấu hình và thao tác mã để tạo các trang HTML.

#### Tổng quan công nghệ .NET CORE

##### Giới thiệu ASP.NET Core

ASP.NET core là một mã nguồn mở và là nền tảng mới cho xây dựng cloud trên internet kết nối các ứng dụng web, loT và mobile backends. ASP.NET Core có thể chạy trên .NET Core hoặc chạy đầy đủ trên .NET Framework. ASP.NET Core được kiến trúc để cung cấp một nền tảng phát triển tối ưu cho các ứng dụng được triển khai tới cloud hoặc chạy on-premises. ASP.NET Core bao gồm các thành phần mô-đun cần thiết tối thiểu, do đó bạn giữ lại được tính linh hoạt trong khi xây dựng các solution của bạn. Bạn có thể phát triển và chạy ASP.NET Core trên Windows, MAC và LINUX. ASP.NET Core là mã nguồn mở tại GitHub

##### Tại sao chọn ASP.NET CORE

* Web UI và Web API được hợp nhất
* Tích hợp các frameworks hiện đại cho khách hàng và nhà phát triển
* A cloud-ready environment-based configuration system
* Built-in dependency injection
* New light-weight and modular HTTP request pipeline
* Khả năng lưu trữ trên IIS hoặc tự chủ trong tiến trình riêng của bạn
* Built on .NET Core, which supports true side-by-side app versioning
* Ships entirely as NuGet packages
* Công cụ vừa mới và đơn giản để phát triển các web hiện đại
* Xây dựng và chạy nền tảng ứng dụng ASP.NET không chỉ trên Windows mà còn trên MAC và Linux nữa
* Mã nguồn mở và có một cộng đồng lớn

##### Đặc điểm của ASP.NET CORE

###### Hỗ trợ đa nền tảng

Ngày nay bạn có thể phát triển và chạy ASP.NET trên cả Windows, Mac, và Linux. Và nếu trên Windows bạn có thể thể sử dụng công cụ tốt nhất Visual Studio 2015 để tạo, quản lý và gỡ lỗi các ứng dụng ASP.NET Core của bạn, thì nay trên bất kỳ nền tảng nào bạn có thể sử dụng Visual Studio Code. Visual Studio Code là một trình soạn thoải với các plugin có hỗ trợ để chỉnh sửa các ứng dụng ASP.NET Core của bạn.

###### Nguồn mở

Ngày nay mã nguồn và tài liệu ASP.NET đã được Microsoft mở tất cả. Các mã nguồn giờ đã có sẵn trong Github bạn giờ có thể tải hay thay đổi bất kỳ mã nào mà bạn thích. Nếu bạn có bất kỳ một góp ý cải tiến gì đó, bạn có thể gửi một yêu cầu đến cho microsoft để xem xét và kết hợp.

Tương tự như vậy, tất cả các tài liệu cũng là mã nguồn mở và có sẵn trong bài viết docs.asp.net. Mỗi trang trên đó điều có chức năng “chỉnh sửa trang này” ở phía trên và bạn có thể chỉnh sửa các tài liệu từ Microsoft

* + - * 1. Hỗ trợ đầy đủ cho framework

Một thông tin hữu ích là asp.net core dẫu trở thành mã nguồn mở nhưng Microsoft vẫn sẽ cung cấp hỗ trợ trong 3 năm cho mỗi bản phát hành lớn nhỏ của họ.

* + - * 1. Hỗ trợ xây dựng bằng Dependency Injection

Nếu mà giải thích thì chỉ vài dòng thì sẽ hơi khó hiểu  
ASP.NET Core đã được xây dựng trong Dependency Injection.Dependency Injection là một mẫu thiết kế cho phép các phụ thuộc của một class được injected như các đối tượng được yêu cầu trong ứng dụng của bạn. Với ASP.NET Core, Microsoft đã cung cấp một Dependency Injection mà bạn có thể sử dụng để xác định sự phụ thuộc được đưa vào Controller, View của bạn, hoặc bất kỳ lớp học khác mà framework sẽ tạo ra cho bạn. Bạn có thể bắt đầu việc cấu hình thông qua phương thức ConfigureServices trong tập tin Startup.cs.

* + - * 1. Một Framework duy nhất

Trong ASP.NET Core, Microsoft đã đưa tất cả các framework vào một framework duy nhất vừa nhẹ hơn và vừa có những tính năng của MVC và WebAPI. Với việc sáp nhập này của MVC và Web API, mọi thứ đơn giản hơn nhiều khi bạn không cần phải cân nhắc định tuyến khác nhau, an toàn, hoặc các bộ lọc cho một ApiController so với MVC Controller. Tất cả các lớp Controller hiện giờ có thể xử lý các yêu cầu sử dụng API Web hoặc cách tiếp cận MVC.

* + - * 1. MVC Helpers Tag

Với ASP.NET Core, Microsoft đã giới thiệu tag helpers để tạo ra mã phía client từ .NET và làm cho nó dễ dàng hơn để tái sử dụng trong Razor markup. Nó được tham chiếu trong đánh dấu phía máy chủ của bạn như thể họ là một tag HTML mà bạn được sử dụng. Công cụ Razor sẽ nhận ra thẻ và thực thi các mã .NET có liên quan tương ứng với nó

#### ASP.NET Web Forms Model

Web Forms ASP.NET mở rộng mô hình sự kiện tương tác với các ứng dụng web. Trình duyệt gửi biểu mẫu web tới máy chủ web và máy chủ trả về trang đánh dấu đầy đủ hoặc trang HTML để phản hồi.

Tất cả các hoạt động của người dùng phía máy khách được chuyển tiếp đến máy chủ để xử lý trạng thái. Máy chủ xử lý đầu ra của các hành động của máy khách và kích hoạt các phản ứng.

Bây giờ, HTTP là một giao thức không trạng thái. ASP.NET framework giúp lưu trữ thông tin về trạng thái của ứng dụng, bao gồm: Trạng thái trang Trạng thái phiên

Trạng thái trang là trạng thái của máy khách, tức là nội dung của các trường nhập khác nhau trong biểu mẫu web.

Trạng thái phiên là thông tin tập thể thu được từ các trang khác nhau mà người dùng đã truy cập và làm việc, tức là trạng thái phiên tổng thể.

Để xóa khái niệm, chúng ta hãy lấy một ví dụ về một giỏ mua hàng. Người dùng thêm các mục vào giỏ hàng. Các mục được chọn từ một trang, cho biết trang mục và tổng số mục và giá được thu thập được hiển thị trên một trang khác, cho biết trang giỏ hàng. Chỉ HTTP không thể theo dõi tất cả thông tin đến từ các trang khác nhau. ASP.NET phiên nhà nước và cơ sở hạ tầng phía máy chủ theo dõi các thông tin thu thập được trên toàn cầu trong một phiên.

Thời gian chạy ASP.NET mang trạng thái trang đến và đi từ máy chủ qua các yêu cầu trang trong khi tạo các mã thời gian chạy ASP.NET và kết hợp trạng thái của các thành phần phía máy chủ trong các trường ẩn. Bằng cách này, máy chủ sẽ nhận biết trạng thái ứng dụng tổng thể và hoạt động theo cách kết nối hai tầng.

#### ASP.NET Component Model

Mô hình thành phần ASP.NET cung cấp các khối xây dựng khác nhau của các trang ASP.NET. Về cơ bản nó là một mô hình đối tượng, mô tả:

Các đối tác phía máy chủ của hầu hết tất cả các phần tử hoặc thẻ HTML, chẳng hạn như <form> và <input>.

Điều khiển máy chủ, giúp phát triển giao diện người dùng phức tạp. Ví dụ, điều khiển Lịch hoặc điều khiển GridView.

ASP.NET là một công nghệ, hoạt động trên khung .Net chứa tất cả các chức năng liên quan đến web. Khuôn khổ .Net được tạo thành từ một hệ thống phân cấp hướng đối tượng. Một ứng dụng web ASP.NET được tạo thành từ các trang. Khi một người dùng yêu cầu một trang ASP.NET, IIS ủy nhiệm việc xử lý trang đó với hệ thống chạy ASP.NET.

Thời gian chạy ASP.NET biến đổi trang .aspx thành một thể hiện của một lớp, được kế thừa từ trang lớp cơ sở của khung .Net. Do đó, mỗi trang ASP.NET là một đối tượng và tất cả các thành phần của nó, tức là các điều khiển phía máy chủ cũng là các đối tượng.

#### ASP.NET – cài đặt môi trường

ASP.NET cung cấp một lớp trừu tượng trên đầu trang của HTTP mà trên đó các ứng dụng web được xây dựng. Nó cung cấp các thực thể mức cao như các lớp và các thành phần trong một mô hình hướng đối tượng.

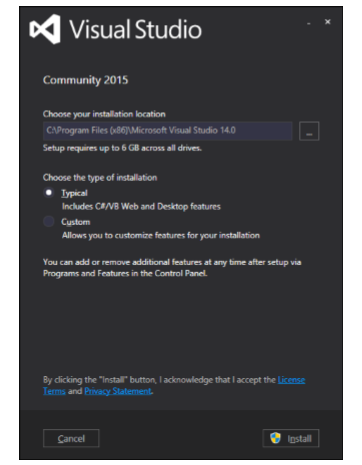
Công cụ phát triển quan trọng để xây dựng các ứng dụng ASP.NET và giao diện người dùng là Visual Studio.

Visual Studio là một môi trường phát triển tích hợp để viết, biên dịch và gỡ lỗi mã. Nó cung cấp một bộ đầy đủ các công cụ phát triển để xây dựng các ứng dụng web ASP.NET, các dịch vụ web, các ứng dụng máy tính để bàn và các ứng dụng di động.

**Cài đặt**

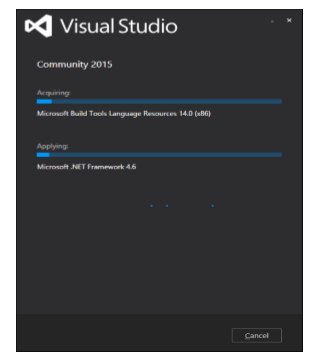
Microsoft cung cấp một phiên bản miễn phí của visual studio cũng chứa SQL Server và nó có thể được tải xuống từ [www.visualstudio.com](http://www.visualstudio.com).

**Bước 1** - Sau khi tải xong, chạy trình cài đặt. Hộp thoại sau sẽ được hiển thị.



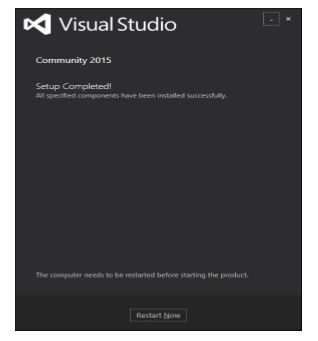
Hình I-1: Màn hình cài đặt [1]

**Bước 2** - Nhấn vào nút Install và nó sẽ bắt đầu quá trình cài đặt



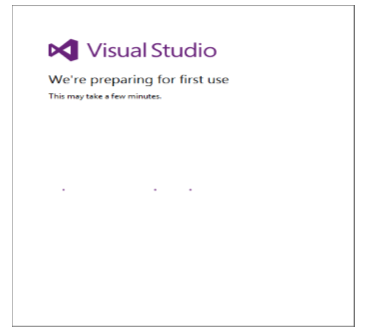
Hình I-2: Màn hình chạy cài đặt [2]

**Bước 3** - Khi quá trình cài đặt hoàn tất thành công, bạn sẽ thấy hộp thoại sau. Đóng hộp thoại này và khởi động lại máy tính của bạn nếu cần.



Hình I-3: Màn hình chạy cài đặt xong [3]

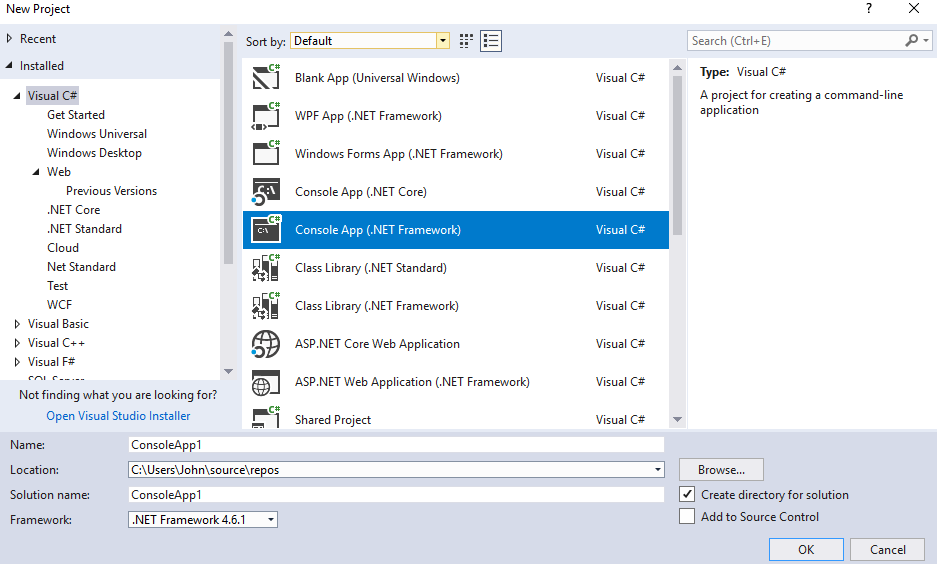
**Bước 4** - Mở Visual Studio từ Start Menu sẽ mở hộp thoại sau. Nó sẽ là một thời gian cho lần đầu tiên chuẩn bị.

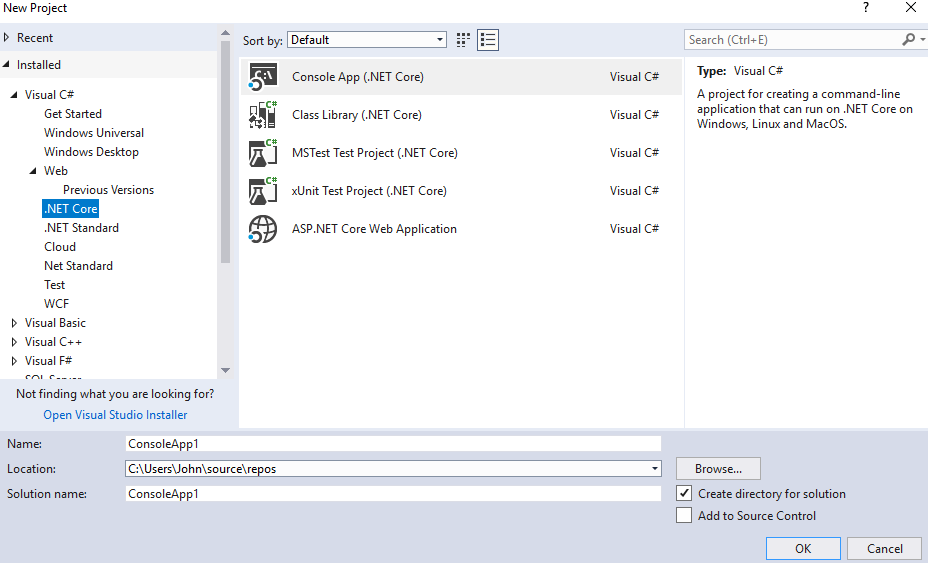


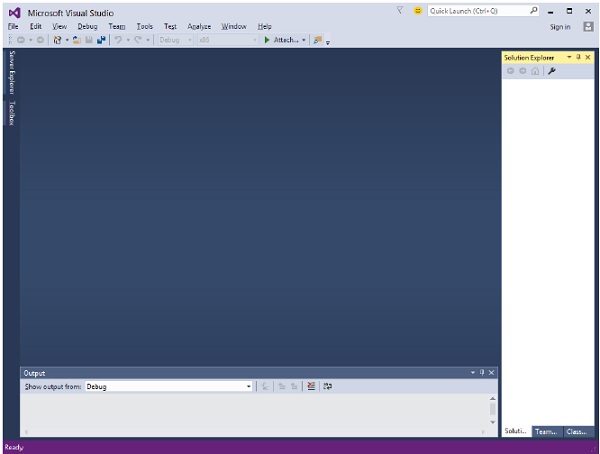
Hình I-4: Màn hình chờ chạy [4]

**Bước 5** - Khi tất cả được thực hiện, bạn sẽ thấy cửa sổ chính của Visual studio. [30]

Tạo 1 project mới từ File → New → Project

Hình I-5: Màn hình tạo newproject [5]



Hình I-6: Màn hình tạo app [6]

Hình I-7: Màn hình chờ app hiện [7]

#### .Net Web API là gì?

##### Khái niệm

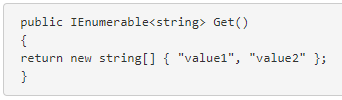
API Web của Net là một cách dễ dàng để triển khai một dịch vụ web RESTful bằng cách sử dụng tất cả các tính ưu việt mà khung .net cung cấp. Một khi bạn hiểu các nguyên tắc cơ bản của REST, thì một Web API .net sẽ rất dễ thực hiện.

Web API được xây dựng trên mô hình đường ống mô-đun, có thể cắm được của net. Điều này có nghĩa là khi một máy chủ lưu trữ một API web nhận được một yêu cầu, nó sẽ đi qua đường dẫn yêu cầu. Điều này cho phép bạn dễ dàng thêm các mô-đun của riêng bạn nếu bạn thấy rằng các khả năng mặc định không đủ cho nhu cầu của bạn. Với các thông báo gần đây về ASP.net vNext, điều này cũng có nghĩa là bạn có khả năng lưu trữ Web API của mình bên ngoài Windows Server, mở ra một loạt các trường hợp sử dụng. Xem http://www.asp.net/vnext để biết chi tiết.

Web API sử dụng các khái niệm điều khiển và hành động từ MVC vì vậy nếu bạn đã hiểu .net MVC bạn đang ở một nơi tốt. Nếu không, thì Web API là một cách tuyệt vời để học MVC.

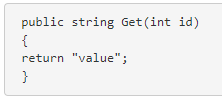
Tài nguyên được ánh xạ trực tiếp tới bộ điều khiển; bạn thường sẽ có một bộ điều khiển khác nhau cho mỗi thực thể dữ liệu chính của bạn (Sản phẩm, Người, Đặt hàng, v.v.). Web API sử dụng công cụ định tuyến .net để ánh xạ URL tới các bộ điều khiển. Thông thường, các API được giữ trong một tuyến đường ‘/ api /’ giúp phân biệt các bộ điều khiển API từ các API không khác trong cùng một trang web.

Các hành động được sử dụng để ánh xạ tới các động từ HTTP cụ thể, ví dụ bạn thường có một hành động GET trả về tất cả các thực thể. Hành động này sẽ phản hồi / api / Sản phẩm (trong đó ‘sản phẩm’ là bộ điều khiển của bạn) và trông giống như sau:



Hình I-8: Tạo phương thức Get() [8]

Bạn cũng có thể có hành động GET chấp nhận một ID cụ thể và trả về một thực thể cụ thể. Nó sẽ trả lời / api / Products / 81 và trông giống như sau:



Hình I-9:Trả về giá trị id cho Get() [9]

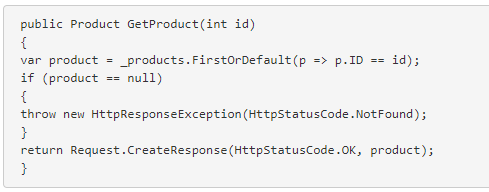
##### Web API là một phần của ‘Một ASP.net’

Web API là một phần của gia đình ‘Một ASP.net’ có nghĩa là nó hỗ trợ tất cả các tính năng chia sẻ tuyệt vời mà bạn hiện có thể sử dụng với MVC hoặc biểu mẫu web, bao gồm (đây chỉ là một vài ví dụ):

* Entity Framework
* Authorisation and identity
* Scaffolding
* Routing

##### Serialization và mô hình ràng buộc

Web API được thiết lập theo mặc định để cung cấp các phản hồi bằng XML hoặc JSON (JSON là mặc định). Tuy nhiên, với tư cách là nhà phát triển, bạn không cần thực hiện bất kỳ chuyển đổi hoặc phân tích cú pháp nào - bạn chỉ cần trả lại đối tượng được nhập mạnh và API Web sẽ chuyển đổi thành XML hoặc JSON và trả lại cho khách hàng gọi điện, đây là quá trình được gọi là Đàm phán nội dung. Đây là một ví dụ về hành động GET trả về một đối tượng Product được gõ mạnh.



Hình I-10: Hàm GetProduct() [10]

Điều này cũng hoạt động cho các yêu cầu đến bằng cách sử dụng tính năng được gọi là Xác thực mô hình. Với Xác thực mẫu, Web API có thể xác thực và phân tích cú pháp dữ liệu nội dung phản hồi đến một đối tượng được nhập mạnh để bạn làm việc với mã của mình. Đây là một ví dụ về mô hình ràng buộc:



Hình I-11:Hàm Post() [11]

##### Trang trợ giúp API động

Nếu bạn chọn Web API làm loại dự án của mình, khi bạn tạo dự án lần đầu tiên, bạn sẽ nhận được tài liệu tự động với API của mình. Đây là một dự án MVC thông thường nằm bên cạnh API của bạn. Dự án trang trợ giúp xem xét mã API, mô hình, thuộc tính của bạn và tự động xây dựng tài liệu, bao gồm các ví dụ, định nghĩa mô hình và hơn thế nữa.

Tài liệu này làm cho một nguồn tài nguyên tuyệt vời để giúp các nhà phát triển của khách hàng gọi điện hiểu cấu trúc của API của bạn. Trang trợ giúp được chứa trong một tuyến đường có tên là ‘/ help /’.

### ANGULAR 4

#### Giới thiệu.

Angular 4 là một khung công tác JavaScript để xây dựng các ứng dụng và ứng dụng web bằng JavaScript, html và TypeScript, là một phần siêu nhỏ của JavaScript. Angular cung cấp các tính năng tích hợp cho hoạt ảnh, dịch vụ http và các tài liệu mà lần lượt có các tính năng như tự động hoàn thành, điều hướng, thanh công cụ, menu, v.v. Mã được viết bằng TypeScript, biên dịch thành JavaScript và hiển thị giống nhau trong trình duyệt.

#### Component trong angular 4.

Component trong angular 4 không tương ứng với controller trong angular 1,vì đây là 2 mô hình quản lí frontend này khác nhau. Controller trong angular 1 dùng để quản lí một phần view, còn component trong angular 4 thì có tác dụng lớn hơn thế, không chỉ quản lí một phần view nó còn phân tách ra view ra.

// Angular 1

const module = angular.module('myModule', []);

module.component('myComponent', {

template: '<div>Hello World!</div>'

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Angular 4

import {Component} from '@angular/core';

@Component({

selector: 'my-component',

template: '<div>Hello World</div>'

})

class MyComponent {}

Hình I-12: So sánh component [12]

#### Directive trong angular 4

Directive trong angular 1 và angular 4 hoàn toàn mapping với nhau. Nhưng vẫn có khác biệt đó là trong angular 1 directive sẽ thay đổi attribute của dom, còn structure directive sẽ thay đổi cấu trúc dom, nói chung là giống hệt với angular 1 chỉ có điều là nó thêm 1 defaul directive trong core angular 4 đó là **directive ng-switch**

// Angular 1

const module = angular.module('myModule', []);

module.component('myComponent', {

template: `

<div

ng-if="$ctrl.isShown"

ng-click="$ctrl.onClick($event)"

ng-class="{blue: $ctrl.isBlue}">

Hello World!

</div>

<ul>

<li ng-repeat="name in $ctrl.names">{{name}}</li>

</ul>

`

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Angular 4

import {Component} from '@angular/core';

@Component({

selector: 'my-component',

template: `

<div

\*ngIf="isShown"

(click)="onClick($event)"

[ngClass]="{blue: isBlue}">

Hello World!

</div>

<ul>

<li \*ngFor="let name of names">{{name}}</li>

</ul>

`

})

class MyComponent {}

Hình I-13: So sánh directive [13]

#### Input Model trong angular 4

Lưu ý rằng trong angular 2 trở về sau để tăng hiệu năng sử dụng tính năng two way binding đã bị ngắt khỏi default. Vì vây để sử dụng nó phải sử dụng cú pháp:

// one way binding

<input type="text" [ngModel]="foo.bar" />

// two binding

<input id="name" name="name" [(ngModel)]="name" />

và cả 2 đều có những attribute to valid như : ng-valid, ng-invalid, ng-dirty, ng-pristine...... như angular 1

// Angular 1

const module = angular.module('myModule', []);

module.component('myComponent', {

template: `

<form name="myForm">

<label for="name">Your Name</label>

<input id="name" name="name" ng-model="$ctrl.name" required />

<div ng-if="myForm.name.$error">Name is Required</div>

</form>

`

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Angular 4

import {Component} from '@angular/core';

@Component({

selector: 'my-component',

template: `

<form>

<label for="name">Your Name</label>

<input id="name" name="name" [(ngModel)]="name" required #nameField="ngModel" />

<div \*ngIf="!nameField.valid">Name is Required</div>

</form>

})

class MyComponent {

name: string;

}

Hình I-14: So sánh Input Model [14]

#### Filter trong angular 4

Trong angular 1 khái niệm này được gọi là filter còn trong angular 2 trở về sau được gọi là pipe.

// Angular 1

const module = angular.module('myModule', []);

module.filter('kebabCase', () => (input) => input.toLowerCase().replace(' ', '-'));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Angular 4

import {Pipe, PipeTransform} from '@angular/core';

@Pipe({name: 'kebabCase'})

class KebabCasePipe implements PipeTransform {

transform(input: string): string {

return input.toLowerCase().replace(' ', '-') }

}

// Usage (Angular 1 + Angular 2)

// {{'Hello World' | kebabCase}}

Hình I-15: So sánh Filter [15]

#### Providers/Services/Factories/Values/Constants trong angular 4

Các khái niệm này trong angular 1 đều đã được gộp lại thành một khái niệm suy nhất trong các phiên bản angular 2 trở về sau đó là các module injectable, được sử dụng như những depndence injection.

// Angular 1

const module = angular.module('myModule', []);

module.service('UserService', ['$http', function ($http) {

this.getUsers = () => {

// Code to retrieve users here

}

}]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Angular 4

import { Injectable } from '@angular/core';

import { Http } from '@angular/http';

@Injectable()

class UserService {

constructor(private http: Http) {}

getUsers() {

// Code to retrieve users here

}

}

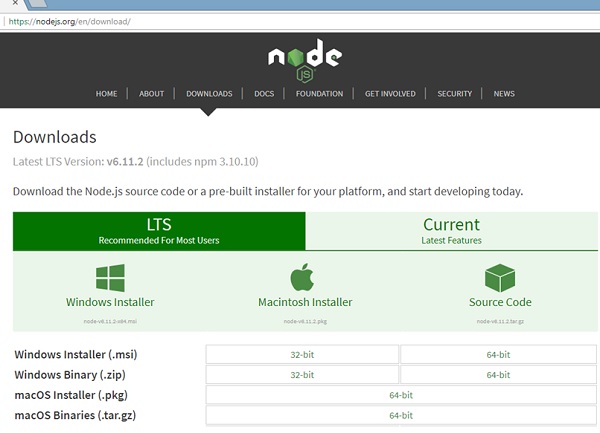
Hình I-16: So sánh Providers/Services/Factories/Values/Constants [16]

#### Cài đặt môi trường

Chúng ta sẽ thảo luận về thiết lập môi trường cần thiết cho Angular 4. Để cài đặt Angular 4, chúng ta yêu cầu những điều sau đây:

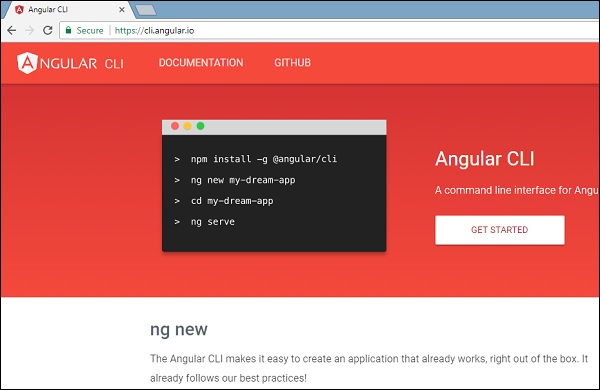
1. Nodejs
2. Npm
3. Angular CLI
4. IDE for writing your code

Để cài đặt nodejs, hãy vào trang chủ https://nodejs.org/en/download/ của nodejs và cài đặt gói dựa trên hệ điều hành của bạn.



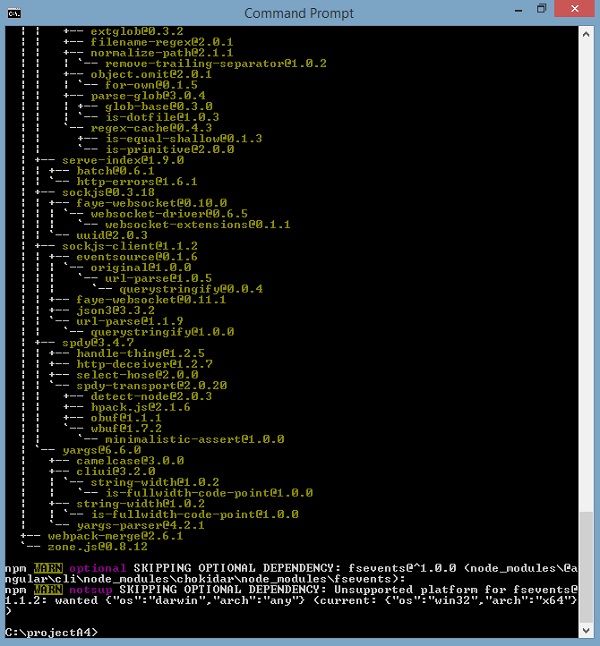
Hình I-17: Trang tải Nodejs[17]

Tiếp theo chúng ta sẽ cài đặt angular 4 bằng cách truy cập vào trang chủ https://cli.angular.io/ của Angular để lấy tham chiếu của lệnh.



Hình I-18:Trang tải Angular [18]

Nhập npm install –g @ angular / cli, để cài đặt cli trên hệ thống của bạn.



Hình I-19: Cài đặt Angular [19]

Một khi Angular CLI được cài đặt. Bạn có thể sử dụng bất kỳ IDE nào do bạn chọn, tức là WebStorm, Atom, Visual Studio Code, v.v..

## PHẦN II: TÌM HIỀU VỀ SEARCH ENGINE VÀ CÔNG CỤ TÌM KIẾM LUCENE VÀ LUCENE.NET

### Lịch sử của SEARCH ENGINE

Ra đời vào năm 1993, cho đến hiện tại Search Engine đã trở thành một phần không thể thiếu của cuộc sống nói chung và cộng đồng ngành công nghệ thông tin (IT) nói riêng, không ngừng được cải thiện và tiến bộ về số lượng cũng như chất lượng các kết quả tìm kiếm đã mang đến cho người dùng những thông tin quan trọng thiết thực và cần thiết. Sự phát triển của Search Engine được gắn liền với sự phát triển của công nghệ thông tin trên toàn thế giới, và đặc biệt hơn hết là nó gắn liền với sự thay đổi của ngành dịch vụ SEO. Vậy thì, Search Engine (SE) là gì ?

### Khái niệm về SEARCH ENGINE.

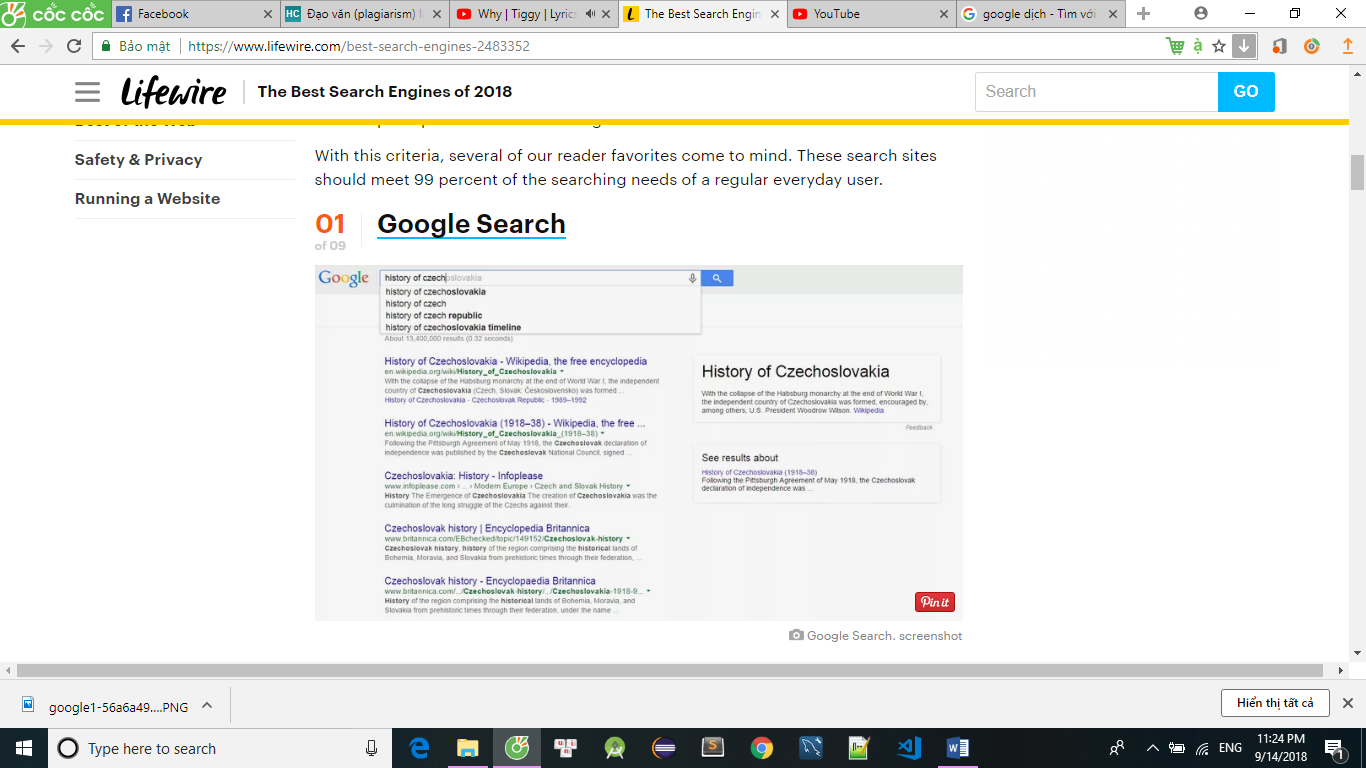
Năm 1998, Đã có hai đồng sáng lập Google đưa ra một thuật toán được gọi là [PageRank](https://www.jpwebseo.com/chi-pa-da-la-gi-trong-seo) nhằm giúp đánh giá chất lượng nội dung các website một cách tự động và tốt hơn đồng thời cũng đã triển khai thử nghiệm trên internet.Và đó cho đến 2013-2014, với sự hoàn thiện và phát triển về các tính năng cũng như có các cộng sự xuất sắc về kỹ thuật máy tính và hạ tầng đủ mạnh, Google chính là cái tên mà được cả thế giới tin dùng với thống kê cho thấy đã chiếm tới 85% thị phần.Và đến ngày nay, hễ nhắc đến khái niệm máy chủ hay công cụ tìm kiếm thì ai cũng nghĩ tới Google.

Công cụ tìm kiếm là một dịch vụ cho phép người dùng Internet tìm kiếm nội dung thông qua World Wide Web (WWW). Người dùng nhập từ khóa hoặc cụm từ khóa vào công cụ tìm kiếm và nhận danh sách kết quả nội dung web dưới dạng trang web, hình ảnh, video hoặc dữ liệu trực tuyến khác. Danh sách nội dung được trả về thông qua công cụ tìm kiếm cho người dùng được gọi là trang kết quả của công cụ tìm kiếm (SERP).

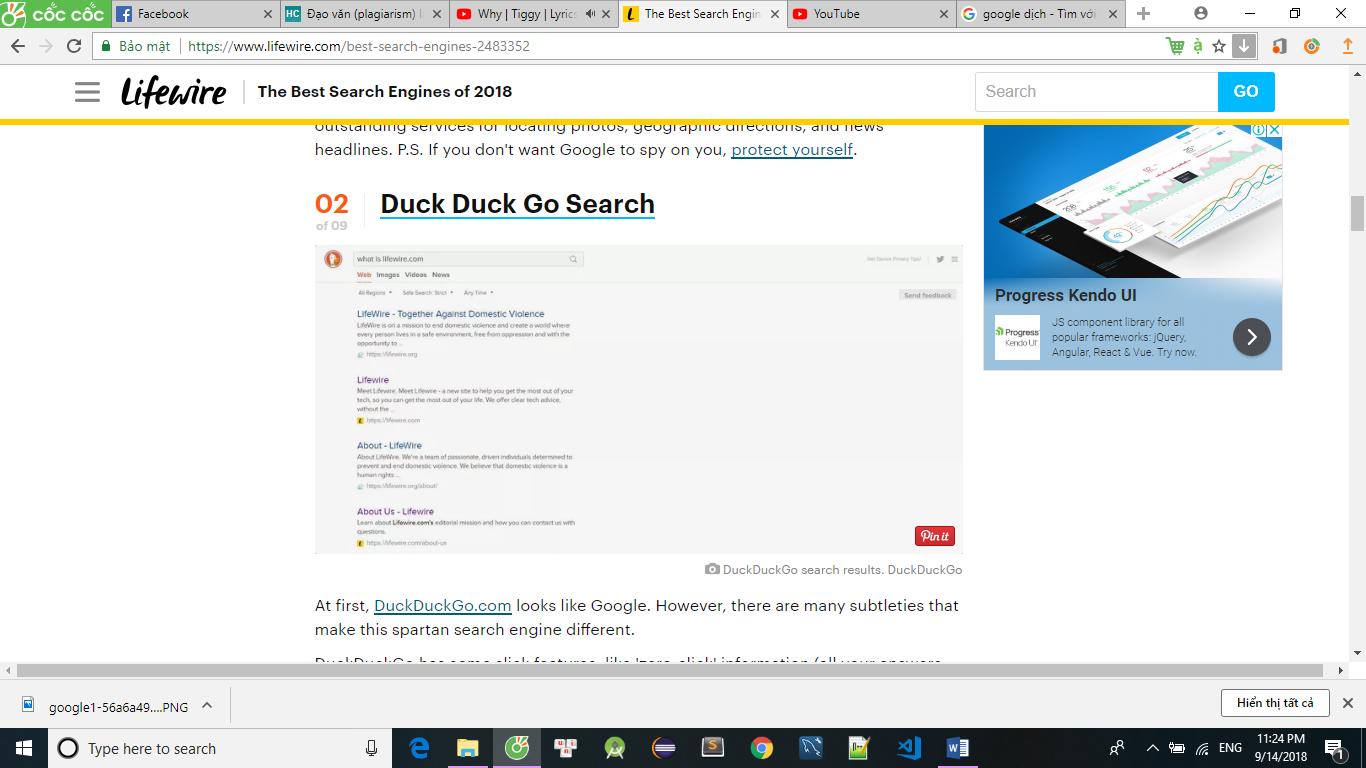
Để đơn giản hóa, hãy nghĩ về công cụ tìm kiếm dưới dạng hai thành phần. Đầu tiên, trình thu thập dữ liệu spider/web crawler trolls cho nội dung được thêm vào chỉ mục của công cụ tìm kiếm (index). Sau đó, khi người dùng truy vấn công cụ tìm kiếm, kết quả có liên quan được trả về dựa trên thuật toán của công cụ tìm kiếm. Các công cụ tìm kiếm sớm dựa chủ yếu vào nội dung trang, nhưng khi các trang web học cách chơi hệ thống, các thuật toán trở nên phức tạp hơn và kết quả tìm kiếm được trả về có thể dựa trên hàng trăm biến.

Đã từng có một số lượng đáng kể các công cụ tìm kiếm có thị phần đáng kể. Hiện tại, Google và Bing của Microsoft kiểm soát phần lớn thị trường. (Trong khi Yahoo tạo ra nhiều truy vấn, công nghệ tìm kiếm back-end của họ được thuê ngoài cho Microsoft.)

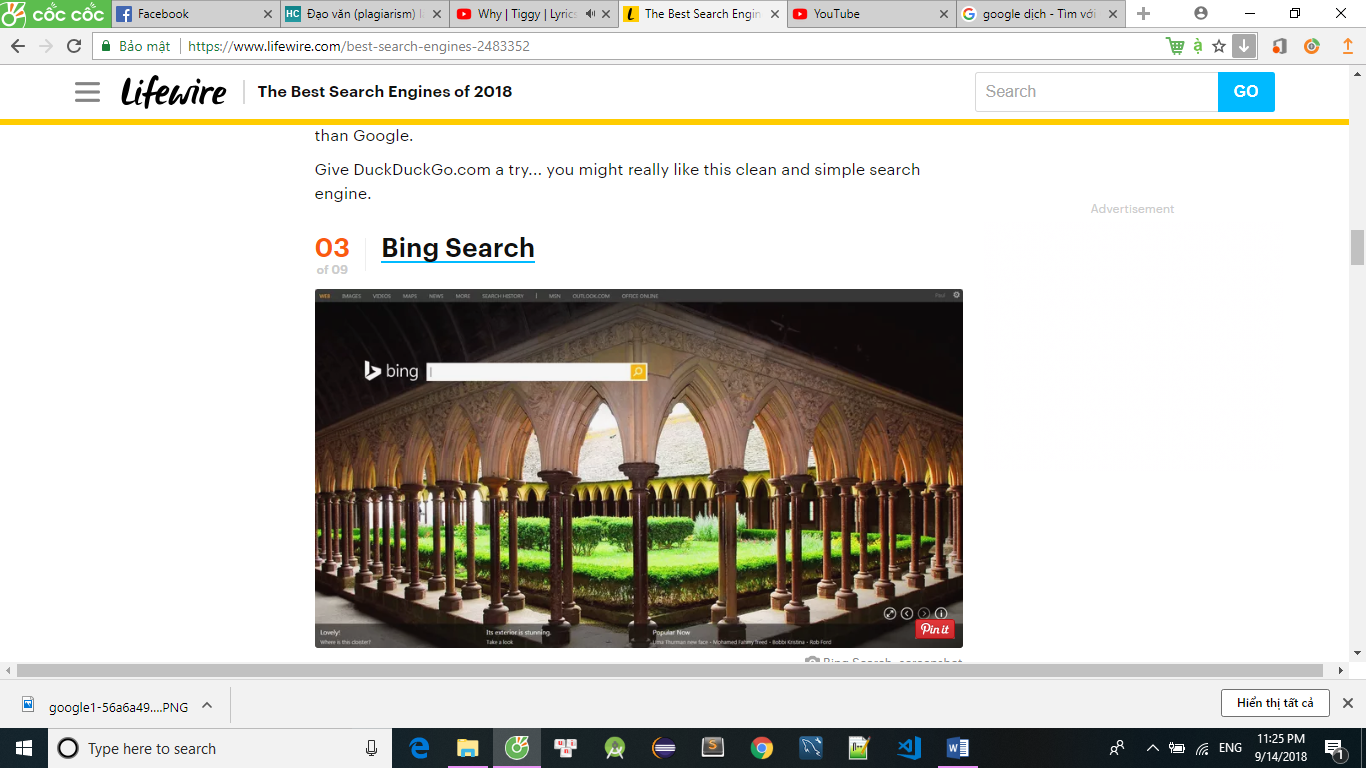
### Một số SEARCH ENGINE hiện nay

****

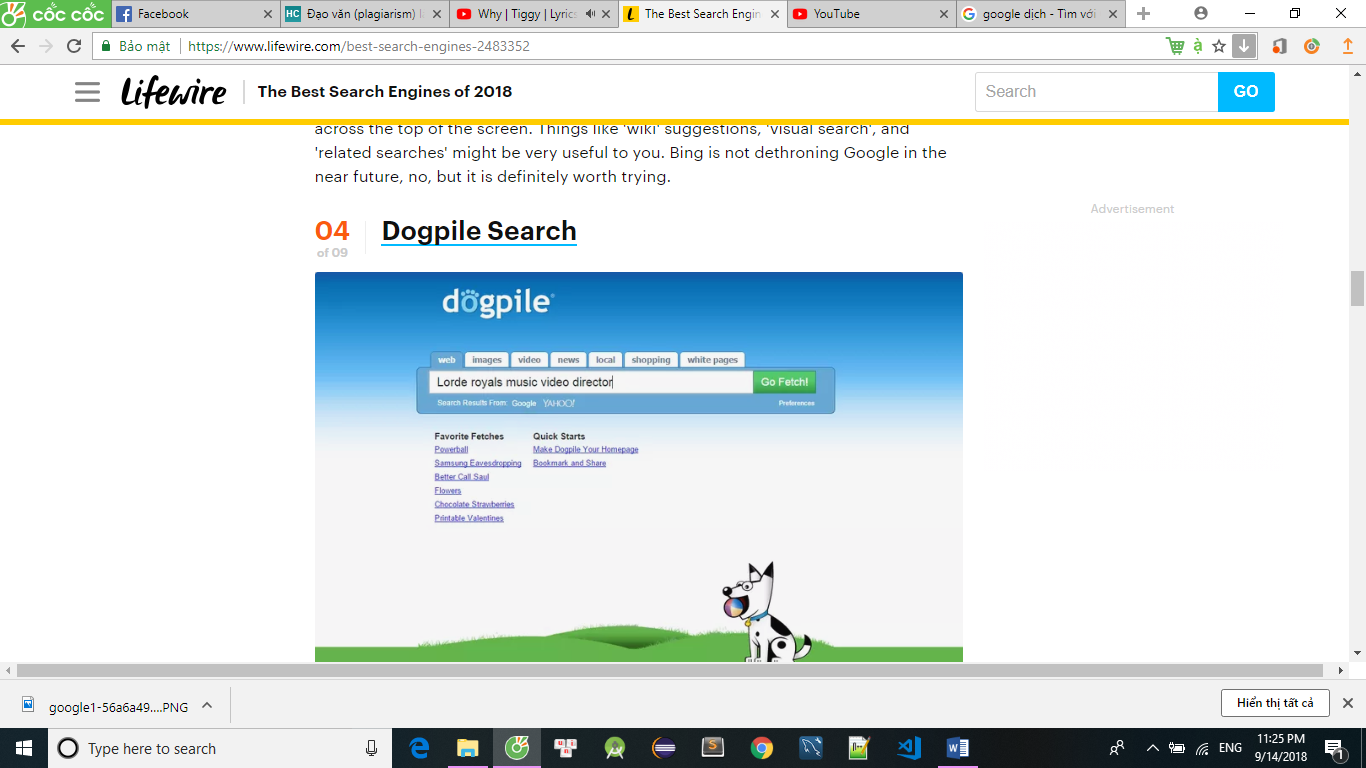
Hình II-1: Google Search[20]

****

Hình II-2: Duck Duck Go Search[21]

****

Hình II-3: Bing Search[22]

****

Hình II-4: Dogpile Search[23]

### Tìm hiểu về Lucene

#### Tìm kiếm với dữ liệu cấu trúc và phi cấu trúc

Với việc dữ liệu do con người tạo ra ngày càng phong phú, nhu cầu tìm kiếm thông tin ngày càng bức thiết và đa dạng, dẫn đến sự ra đời của nhiều công cụ hỗ trợ tìm kiếm. Trong lĩnh vực tìm kiếm, người ta tạm chia dữ liệu thành hai loại chính, dữ liệu có cấu trúc (structured data) và dữ liệu phi cấu trúc (unstructured data).

Dữ liệu có cấu trúc thường dùng để chỉ dữ liệu lưu trữ trong các hệ quản trị cơ sở dữ liệu quan hệ như MS SQL server hay MySQL, trong đó các thực thể và các thuộc tính được định nghĩa sẵn. Ví dụ, dữ liệu của một thí sinh dự thi đại học có thể bao gồm các thông in như Họ tên, Năm sinh, Trường dự thi, Điểm thi các môn.

Dữ liệu phi cấu trúc thường dùng để chỉ dữ liệu ở dạng tự do (free type) và không cần có cấu trúc định nghĩa sẵn. Các trang web, video, ảnh, âm thanh là các ví dụ của dữ liệu phi cấu trúc (Thông thường, nếu ko nói chi tiết, dữ liệu phi cấu trúc được hiểu là dữ liệu phi cấu trúc dạng văn bản).

Nếu dùng các hệ quản trị cơ sở dữ liệu quan hệ như MS SQL Server hay MySQL để lưu trữ dữ liệu phi cấu trúc, việc truy tìm dữ liệu theo danh sách từ khóa sẽ gặp các hạn chế sau:

- Bị giới hạn ở cú pháp của ngôn ngữ SQL. Ví dụ muốn tìm các trang web nói về Cần Thơ, phải chỉ rõ Cần Thơ trong câu truy vấn, nếu dùng "Can Tho" (vì nghĩ có thể sử dụng tiếng Việt không dấu), kết quả sẽ hoàn toàn khác.

- Tốc độ chậm khi tìm kiếm gần đúng (dùng LIKE) trong cơ sở dữ liệu lớn. Các hệ quản trị CSDL quan hệ không được thiết kế để làm việc hiệu quả cho trường hợp người dùng muốn thực hiện các câu truy vấn như: Tìm các tài liệu về các ngành đào tạo của Đại học Cần Thơ, trừ ngành nông nghiệp và ngành công nghệ thông tin (\*).

Điều này đã thúc đẩy cho sự ra đời của các hệ thống hỗ trợ tìm kiếm theo danh sách từ khóa cho dữ liệu phi cấu trúc (full text search - tạm dịch là tìm kiếm toàn văn). Điển hình nhất cho các hệ thống này là các máy tìm kiếm như Google và Yahoo. Gần đây các hệ quản trị CSDL quan hệ như MS SQL Server (phiên bản MS SQL Server 2000 trở đi) và MySQL (phiên bản 3.23.23 trở đi) đều có thêm chức năng hỗ trợ tìm kiếm toàn vẹn cho dữ liệu văn bản mà chúng lưu trữ.

#### Lucene - thư viện mã nguồn mở hỗ trợ tìm kiếm toàn văn

Lucene là một thư viện mã nguồn mở cho tìm kiếm toàn văn, được phát triển bởi Dough Cutting (hiện đang làm việc cho Yahoo). Thư viện này cung cấp các hàm cơ bản hỗ trợ cho việc đánh chỉ mục và tìm kiếm. Để có thể sử dụng Lucene, bạn cần phải có sẵn dữ liệu. Dữ liệu có thể là tập hợp các tập tin dạng PDF, Word hay là các trang web HTML; hoặc là dữ liệu lưu trong các hệ quản trị CSDL như MS SQL Server hay MySQL. Dùng Lucene, bạn có thể tiến hành đánh chỉ mục trên dữ liệu hiện có để sau này có thể thực hiện thao tác tìm kiếm toàn văn trên dữ liệu đó.

|  |
| --- |
|  |
| [mo-hinh-danh-chi-muc-Lucene.jpg](javascript:;)  Hình II-5:Mô hình đánh chỉ mục của Lucene [24] |

Các bước sử dụng Lucene bao gồm: (\*)

*Mô tả đối tượng cần đánh chỉ mục*

Lucene coi mỗi đối tượng cần đánh chỉ mục là một Document. Mỗi Document có thể có nhiều Field, mỗi Field tương ứng một thuộc tính của đối tượng cần đánh chỉ mục. Ví dụ, bạn muốn tìm kiếm các trang web ở dạng HTML. Như vậy đối tượng cần đánh chỉ mục là trang HTML; các thuộc tính có thể là nơi lưu trữ (host), đường dẫn, tiêu đề, metadata và nội dung của chính trang web.

|  |
| --- |
|  |
| *Minh họa đối tượng Document dùng trong Lucene* |

Với mỗi Field, bạn có thể chọn giữa đánh chỉ mục hay không đánh chỉ mục. Nếu chọn đánh chỉ mục, bạn có thể tìm kiếm trên Field đó. Các Field không đánh chỉ mục thường là các Field không quan trọng trong quá trình tìm kiếm và phục vụ chủ yếu cho nhu cầu trình bày kết quả trả về.

*Đánh chỉ mục*

Bạn cần phải tự xây dựng các hàm công cụ để chuyển đổi dữ liệu ban đầu thành dữ liệu mô tả trong Document. Ví dụ, nếu dữ liệu ban đầu của bạn là tập tin PDF hay Word, bạn phải có các hàm để đọc hiểu các định dạng này và chuyển về dạng chuỗi văn bản tương ứng.

Thao tác đánh chỉ mục khá phức tạp. Trước hết dữ liệu văn bản sẽ được phân tích thành các từ khóa, đồng thời loại bỏ các từ không dùng đến (stop words, trong tiếng Anh các từ như a, an, the là các stop words), sau đó các từ khóa sẽ được dùng để tạo inverted index (chỉ mục nghịch đảo) (2) và lưu thành các phân đoạn (segments) dạng thuận tiện cho việc tìm kiếm sau này. Chỉ mục nghịch đảo dùng để chỉ cách lưu trữ danh sách các tài liệu mà có chứa từ cho trước. Gọi là nghịch đảo bởi vì thông thường, với mỗi tài liệu cho trước, người ta lưu trữ danh sách các từ có trong tài liệu đó. Ví dụ, với từ khóa Lucene, ta sẽ lưu trữ danh sách các trang web A, B, C có chứa từ khóa này. Sau này khi người dùng gõ vào từ khóa Lucene, danh sách này có thể giúp định vị nhanh chóng các trang web có chứa nó. Nếu dùng chỉ mục thông thường, phải quét qua hết các trang web có trong cơ sở dữ liệu mới tìm ra. Cách này rất tốn thời gian khi số lượng dữ liệu lớn.

*Tìm kiếm*

Sau khi dữ liệu đã được đánh chỉ mục, bạn có thể thực hiện tìm kiếm trên chúng. Tìm kiếm toàn vẹn cho phép bạn có thể tìm kiếm theo danh sách các từ khóa cùng với các toán tử luận lí (và, hoặc, phủ định). Ví dụ, bạn có thể chỉ định: "Đại học Cần Thơ" NOT ("nông nghiệp" AND "công nghệ thông tin") cho câu truy vấn (\*) ở trên.

Ngoài ra, điểm nổi bật khác của tìm kiếm toàn văn là cho phép xếp hạng các kết quả trả về tùy vào độ tương đồng (relevance) giữa câu truy vấn và kết quả tìm kiếm.

#### Dùng Lucene khác gì với tìm kiếm toàn văn của MySQL và MS SQL server

Có thể thấy hai điểm khác biệt:

- Thứ nhất, dùng Lucene, bạn có thể cung cấp dịch vụ tìm kiếm toàn văn cho bất kì dữ liệu của hệ quản trị CSDL nào. Trong khi đó chức năng tìm kiếm toàn văn của MySQL hay MS SQL Server chỉ dùng cho chính dữ liệu lưu trong chúng mà thôi.

- Thứ hai, dùng Lucene, bạn có thể mở rộng các thành phần hiện có. Ví dụ, bạn có thể bổ sung bộ phân tích từ khóa cho tiếng Việt (để có thể tách cụm từ: 'học sinh học sinh học' thành 3 từ là học sinh, học và sinh học), hoặc bạn có thể thay đổi cách thức tính độ tương đồng cũng như cách xếp hạng cho kết quả trả về. Những điều này không thể làm được với MySQL hay MS SQL Server.

Có một số thư viện sau

- Egothor (http://www.egothor.org/)

- Xapian (http://www.xapian.org/)

- MG4J (http://mg4j.dsi.unimi.it/)

- Sphinx (http://www.sphinxsearch.com/)

Lucene không phải là máy tìm kiếm . Lucene chỉ là tập hợp các hàm thư viện hỗ trợ cho việc đánh chỉ mục và tìm kiếm mà thôi. Để có được ứng dụng như máy tìm kiếm, bạn phải viết chương trình sử dụng các hàm thư viện này. Một lưu ý khác, Lucene không có thư viện hỗ trợ thu thập trang web (web crawler). Nếu bạn quan tâm đến một ứng dụng máy tìm kiếm cụ thể dùng Lucene, hãy tham khảo Nutch, được phát triển bởi cùng tác giả của Lucene.

*Máy tìm kiếm dùng Lucene và so sánh với Google*

Lucene chỉ là một mô hình hỗ trợ cho tìm kiếm toàn vẹn dữ liệu phi cấu trúc. Máy tìm kiếm Google phát triển phức tạp hơn nhiều trong quá trình phân tích dữ liệu, đánh chỉ mục và xếp hạng kết quả tìm kiếm. Gần đây, nhà sáng lập Wikipedia phát triển Wikia Search cũng dựa trên thư viện Lucene (được tích hợp trong Nutch).

|  |
| --- |
|  |
| [cau-truc.jpg](javascript:;)  Hình II-6:Cấu trúc của chỉ mục nghịch đảo[25] |

#### Tìm hiểu về Mã Nguồn mở Lucene cho .NET

##### Lucene.Net là gì ?

Lucene.Net là một line-by-line port phổ biến của Apache Lucene, một thư viện công cụ tìm kiếm văn bản đầy đủ tính năng, hiệu suất cao được viết hoàn toàn bằng Java. Nó là một công nghệ phù hợp cho hầu như bất kỳ ứng dụng nào yêu cầu tìm kiếm toàn văn bản. Đặc biệt, một ứng dụng mà bạn muốn đạt được một cái gì đó gần với kết quả tìm kiếm của Google và không chỉ kết quả tìm kiếm, nhưng kết quả tìm kiếm rất nhanh, hoặc có thể chỉ là kết quả tìm kiếm cực nhanh.

Vì vậy, trong khi về mặt kỹ thuật có thể, mặc dù hơi khó khăn, ta có thể tích hợp Apache Lucene gốc vào ứng dụng .NET , và nó sẽ cung cấp cho bạn tìm kiếm cực nhanh. Nhưng nó sẽ mất khá nhiều thời gian, do đó làm cho trang web của bạn quá phức tạp và dễ bị lỗi. Mục đích chính của Lucene.Net là dễ dàng tích hợp vào bất kỳ ứng dụng .NET nào và cung cấp hầu hết tốc độ và tính linh hoạt của thư viện gốc Java.

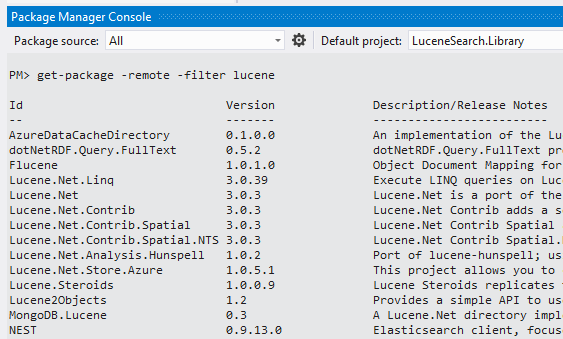
##### Cài đặt Lucene.Net

Để cài đặt Lucene.Net ta sẽ cài bằng NuGet

Mở Package Manager Console trong Visual Studio bằng cách nhấp vào:

**View > Other Windows > Package Manager Console**

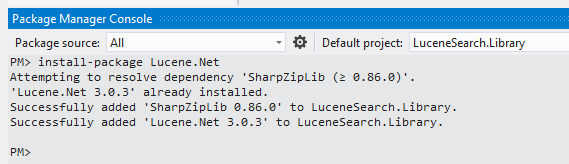
Và một khi bảng điều khiển sẽ bật ở dưới cùng, hãy tìm kiếm 'lucene' đầu tiên bằng cách nhập 'get-package -remote -filter lucene' trong prompt:



Hình II-7:Cài đặt của Lucene[26]

Từ kết quả tìm kiếm, chúng ta có thể thấy rằng có một số gói khác nhau có sẵn cho chúng ta. Đối với hầu hết các phần, các gói này chỉ mở rộng chức năng mặc định của Lucene.Net.Nếu sau này có sử dụng đến thì ta sẽ dùng sau. Gói duy nhất mà chúng ta cần là một Lucene.Net barebone.

Vì vậy, để cài đặt nó, hãy gõ 'install-package Lucene.Net':



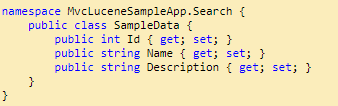
Hình II-8:Cài đặt của Lucene hoàn thành[27]

1. Triển Khai SEARCH dùng Lucene.Net .

Các bước bạn cần thực hiện để xây dựng tìm kiếm cơ bản Lucene.Net

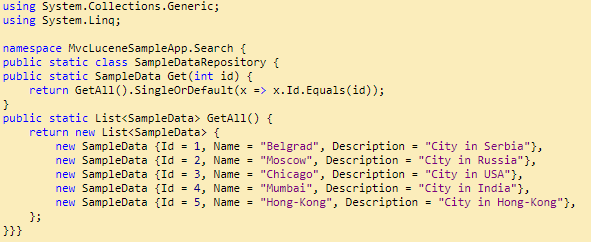
***Bước 1*** *- Tạo nguồn dữ liệu mẫu*

Để sử dụng Lucene.Net, trước hết chúng ta cần tạo chỉ mục tìm kiếm Lucene từ một số tập dữ liệu (có thể là từ cơ sở dữ liệu). Lucene chỉ mục tìm kiếm chỉ là một tập hợp các tập tin văn bản mà Lucene.Net tạo ra, và ta sẽ tạo ra nó sau này. Vì vậy, hãy tạo một tệp SampleData.cs trống và thêm một số đối tượng dữ liệu chung vào nó:



Hình II-9:Lớp khởi tạo dữ liệu mẫu[28]

Lớp này có thể đại diện cho bất kỳ dữ liệu nào bạn muốn, và tất nhiên bạn có thể tạo lớp của riêng bạn hoặc sử dụng lớp hiện có. Ngoài ra, bất cứ khi nào nó đến để thử nghiệm Lucene, bạn có thể muốn tạo một kho lưu trữ nguồn dữ liệu tĩnh đơn giản dựa trên một lớp SampleData ở trên, và bạn có thể gắn mã đó vào cùng một tệp SampleData.cs, ​​bạn vừa tạo:

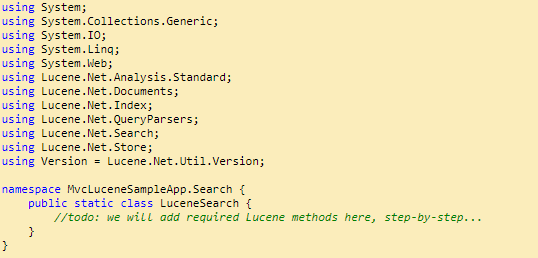
****

Hình II-10:Lớp Sampledata[29]

Chúng ta đã tạo nguồn dữ liệu mẫu với lớp dữ liệu chung SampleData và kho mẫu SampleDataRepository để truy xuất dữ liệu tĩnh .

***Bước 2*** *- Tạo lớp tìm kiếm Lucene rỗng*

Bây giờ hãy tạo một tệp LuceneSearch.cs rỗng. Hãy thêm những đoạn code này vào file.

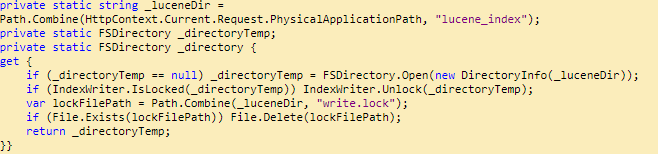
****

Hình II-11:Lớp LuceneSearch [30]

Chúng ta sẽ thêm tất cả các phương thức cần thiết để thực hiện tất cả các công việc liên quan đến tìm kiếm đối với lớp đó. Tại thời điểm đó, bạn có thể đảm bảo tất cả các tham chiếu của mình đều chính xác bằng cách tạo trang web của bạn và nếu việc xây dựng không thành công, hãy sửa các tham chiếu của bạn.

***Bước 3*** *- Thêm trình xử lý thư mục chỉ mục tìm kiếm Lucene*

Chúng ta hãy xem những gì chúng ta có bây giờ - lớp SampleData, đại diện cho một số dữ liệu, và lớp LuceneSearch được cho là thực hiện tìm kiếm. Nhưng nó trống rỗng. Vì vậy, nó cần một số code để làm công việc. Một điều kiện tiên quyết nhỏ đầu tiên. Lucene.Net cần xây dựng chỉ mục tìm kiếm của mình, như đã đề cập trước đó, về cơ bản là một tập hợp các tệp được tạo bởi Lucene trong một số thư mục cục bộ. Vì vậy, chúng ta cần thêm một thuộc tính đặc biệt vào lớp LuceneSearch đại diện cho một trình xử lý cho một thư mục cục bộ sẽ lưu trữ chỉ mục tìm kiếm:

****

Hình II-12:Hàm \_luceneDir thêm vào LuceneSearch [31]

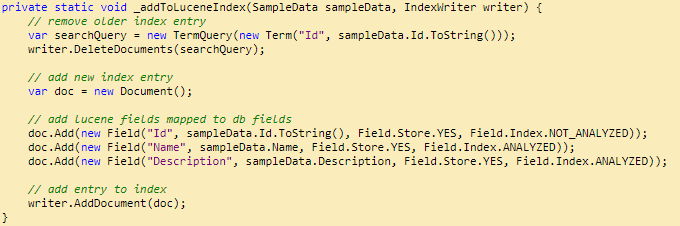
Ở đây \_luceneDir là một đường dẫn vật lý đầy đủ đến thư mục và "lucene\_index" là tên của thư mục đó, nằm trong thư mục gốc của ứng dụng. Vì vậy, tự nhiên bạn cũng cần phải tạo thư mục này bằng tay hoặc viết một số code tạo ra nó tự động. Sau đó, \_directory là một cá thể của lớp FSDirectory của Lucene.Net và sẽ được tất cả các phương thức tìm kiếm sử dụng để truy cập chỉ mục tìm kiếm.

Chúng ta đã thêm trình xử lý thư mục chỉ mục tìm kiếm Lucene để làm cho lớp LuceneSearch của chúng ta sẵn sàng để thêm các phương thức tìm kiếm.

***Bước 4*** *- Thêm các phương thức để Thêm dữ liệu vào chỉ mục tìm kiếm Lucene*

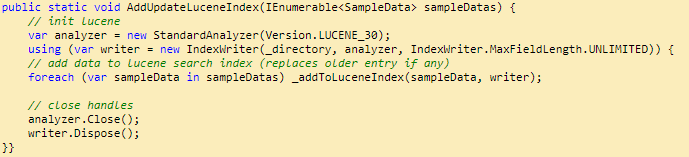
Lucene sẽ tạo chỉ mục tìm kiếm dựa trên một số dữ liệu thực tế, trong trường hợp của chúng ta, nó sẽ là một List <SampleData> với một vài bản ghi, hoặc ghi lại bản ghi SampleData.

Phương pháp đầu tiên chúng ta cần là một phương thức riêng tạo một mục chỉ mục tìm kiếm dựa trên dữ liệu của chúng ta và nó sẽ được tái sử dụng bởi các phương thức công khai mà chúng ta sẽ bổ sung sau:

****

Hình II-13:Hàm \_addToLucene[32]

Về cơ bản, phải mất một bản ghi với một lớp SampleData, ánh xạ nó vào tài liệu lớp Lucene và thêm nó vào chỉ mục tìm kiếm bằng cách sử dụng IndexWriter. Xin lưu ý các trường **<code> Name** và **<code> Description** sử dụng tham số **Field.Index.ANALYZED**, trong khi **Id** sử dụng **Field.Index.NOT\_ANALYZED** một. Về cơ bản, bạn chỉ muốn sử dụng **ANALYZED** trên các thuộc tính văn bản hoặc chuỗi đơn và **NOT\_ANALYZED** trên các giá trị số ít, như các **số nguyên Id**. Bây giờ hãy thêm phương thức công khai sẽ sử dụng \_addToLuceneIndex () để thêm danh sách các bản ghi vào chỉ mục tìm kiếm:

****

Hình II-14:Hàm AddUpdateLuceneIndex[33]

Ngoài ra, hãy thêm phương thức công khai khác sẽ thêm một bản ghi vào chỉ mục tìm kiếm:

****

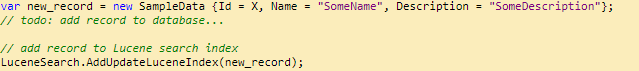
Hình II-15:Phương thức khác sẽ được thêm[34]

Mẫu sử dụng của hai phương thức này rất đơn giản - thêm tất cả các bản ghi cơ sở dữ liệu một lần:

****

Hình II-16:Sử dụng 2 phương thức[35]

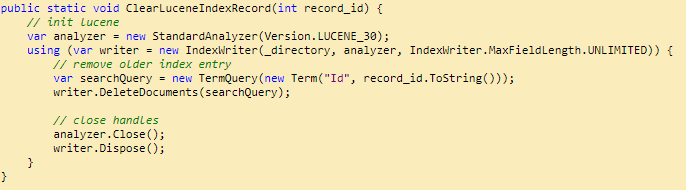
Sau đó, bất cứ khi nào bản ghi mới được thêm vào cơ sở dữ liệu, hãy gọi cùng một phương thức, chỉ cung cấp cho nó bản ghi duy nhất:

****

Hình II-17:Cung cấp duy nhất 1 bản ghi[36]

***Bước 5*** *- Thêm các phương thức xóa và tối ưu hóa dữ liệu từ chỉ mục tìm kiếm Lucene*

Bất cứ khi nào banr ghi cơ sở dữ liệu của chúng ta bị xóa, chúng cũng sẽ bị xóa khỏi chỉ mục tìm kiếm Lucene, nếu không, tìm kiếm của bạn sẽ trả về các bản ghi có thể không tồn tại trong cơ sở dữ liệu. Trước tiên, hãy thêm một phương thức đơn giản để xóa bản ghi đơn khỏi chỉ mục tìm kiếm Lucene theo trường Id của bản ghi:

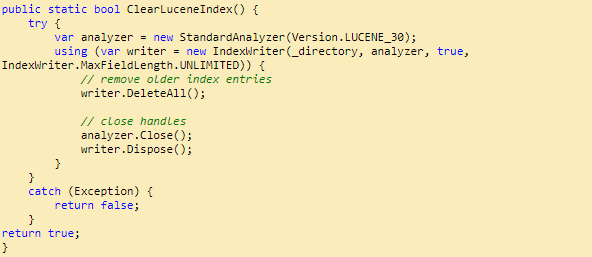
****

Hình II-18:Lớp ClearLuceneIndexRecord[37]

Mã trên cơ bản tìm kiếm bản ghi theo trường Id (và bạn có thể sử dụng bất kỳ trường nào), nhận kết quả tìm kiếm từ chỉ mục và xóa tất cả chúng khỏi chỉ mục và trong trường hợp của chúng ta, nó thường sẽ xóa bản ghi duy nhất, miễn là bạn luôn cung cấp giá trị Id đầy đủ và duy nhất.

Lưu ý: các phương thức tìm kiếm chính của Lucene sẽ được thêm vào sau và dĩ nhiên được giải thích chi tiết hơn.

Thứ hai, bất cứ khi nào thay đổi lược đồ cơ sở dữ liệu, hoặc bạn chỉ muốn xóa toàn bộ chỉ mục một cách nhanh chóng, bạn sẽ cần một phương thức để xóa tất cả chỉ mục, vì vậy hãy thêm quá trình đó:

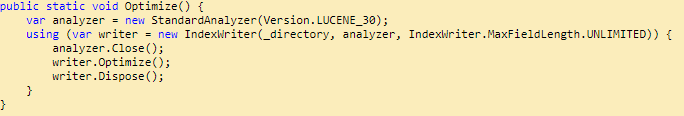
****

Hình II-19: Lớp ClearLuceneIndex [38]

Phương thức này chỉ đơn giản loại bỏ toàn bộ chỉ mục tìm kiếm Lucene thông qua một phương thức được xây dựng trong Lucene IndexWriter. Bây giờ có lẽ là một thời điểm tốt để đề cập đến việc Lucene đặt một "khóa" vào các tệp chỉ mục tìm kiếm, vì vậy khi chúng được cập nhật hoặc tìm kiếm, vì vậy chúng không thể bị thay đổi. Ngoài ra, điều quan trọng là mỗi tệp được "mở khóa" vì nếu một số tệp sẽ bị xóa theo cách thủ công hoặc bằng một số mã và bạn cố gắng tìm kiếm / cập nhật sau đó, nó sẽ khiến bạn thấy nhiều lỗi và chúng ta đều không muốn điều đó.

Vì vậy, chúng ta luôn cần .Close () và .Dispose () bất kỳ trình xử lý Lucene nào như IndexWriter và StandardAnalyzer.

Ngoài ra, sẽ có ích khi chạy tối ưu hóa chỉ mục tìm kiếm Lucene một lần trong một thời gian để tăng tốc tìm kiếm, đặc biệt nếu chỉ mục của bạn ngày càng lớn hơn. Vì vậy, hãy thêm một phương thức nhỏ để làm điều đó:

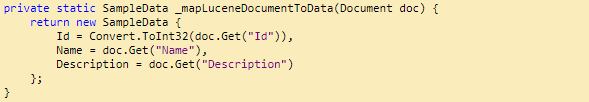
****

Hình II-20: Hàm tối ưu trong 1 thời gian [39]

Chúng ta đã thêm ba phương thức - ClearLuceneIndexRecord () để xóa bản ghi duy nhất khỏi chỉ mục tìm kiếm Lucene, ClearLuceneIndex () để xóa tất cả các bản ghi trong chỉ mục và phương thức Optimize () để tối ưu hóa các chỉ mục lớn để tìm kiếm nhanh hơn.

***Bước 6*** *- Thêm các phương thức để ánh xạ dữ liệu chỉ mục tìm kiếm Lucene đến SampleData*

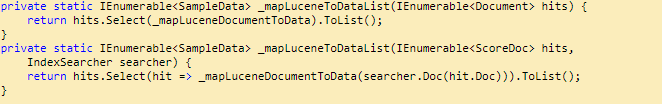
Dừng ở đó, mọi người, bước này là bước cuối cùng, trước khi chúng ta thêm các phương pháp tìm kiếm chính của mình! Bây giờ trước khi chúng ta có thể tìm kiếm, bạn có nhớ phương thức **\_addToLuceneIndex ()** của chúng ta từ **Bước 4**, ánh xạ dữ liệu cơ sở dữ liệu của chúng ta tới chỉ mục tìm kiếm Lucene không? Vâng, để có được kết quả tìm kiếm của chúng ta dưới dạng **ListDampleData** hoặc tương tự, chúng ta cần một hàm sẽ ánh xạ chỉ mục tới lớp **SampleData** của chúng ta, và ở đây nó là:

****

Hình II-21: Ánh xạ SampleData [40]

Vì vậy, phương thức trên sẽ lấy tài liệu Lucene từ chỉ mục (trong đó mỗi trường được biểu diễn dưới dạng một chuỗi) và sẽ ánh xạ nó tới SampleData. Khá đơn giản.

Ngoài ra, chúng ta cần thêm hai phương thức nữa để ánh xạ một danh sách các tài liệu Lucene và một danh sách các Lucene ScoreDocs, mỗi phương thức được trả về bởi phương thức tìm kiếm Lucene khác nhau, và nhiều hơn nữa trong bước 7 và 8. Cả hai phương thức đó sử dụng lại phương thức **\_mapLuceneDocumentToData ()** được xác định ở trên:

****

Hình II-22: Hàm \_mapLuceneDocument [41]

Chúng ta đã thêm các phương thức để ánh xạ kết quả do Lucene trả về cho lớp dữ liệu của chúng ta để được sử dụng lại trên trang web. Phương thức **\_mapLuceneDocumentToData ()** ánh xạ Tài liệu Lucene với các kết quả tìm kiếm từ chỉ mục đến lớp của chúng ta **SampleData** và phương thức **\_mapLuceneToDataList ()** lần lượt ánh xạ một danh sách các tài liệu Lucene hoặc ScoreDocs.

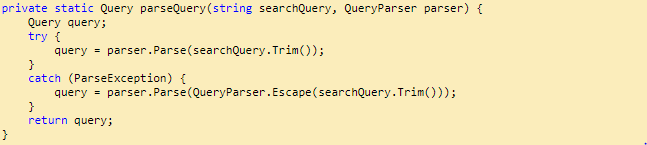
***Bước 7*** *- Thêm phương pháp tìm kiếm chính*

Và cuối cùng, phương pháp tìm kiếm chính. Nó sẽ tìm kiếm chỉ mục tìm kiếm Lucene theo một trường cụ thể (Id, Name hoặc Description) bất cứ khi nào chúng ta cung cấp tên, hoặc, nó sẽ tìm kiếm cả ba trường, đây là cơ sở cho tìm kiếm phổ biến tương tự như công cụ tìm kiếm internet your\_favourite . Bạn có thể nhận thấy rằng nó vẫn là một phương pháp riêng. Lý do cho điều đó là phương thức này là phổ quát cho bất kỳ truy vấn Lucene nào.

Truy vấn Lucene không chỉ là văn bản mà bạn đang tìm kiếm. Ví dụ cơ bản là khi truy vấn trông giống như "Mumbai", Lucene sẽ tìm kiếm đối sánh chính xác cho từ này và nếu truy vấn giống như "Mum \*" thì tất cả các trường có từ bắt đầu bằng "Mẹ" sẽ được trả lại dưới dạng kết quả tìm kiếm. Chắc chắn có nhiều cách để viết các quả Lucene tiên tiến, nhưng điều đó sẽ không được đề cập trong bài viết này.

Trong trường hợp của chúng ta, truy vấn sẽ được cung cấp bởi phương thức công khai, định dạng bạn sẽ yêu cầu tìm kiếm một trường hợp tìm kiếm cụ thể và nó sẽ được thêm vào Bước 8. Kịch bản của chúng ta là phương pháp tìm kiếm riêng dưới đây không thay đổi nhiều, và chỉ một công chúng sẽ được điều chỉnh cho nhu cầu tìm kiếm của chúng ta.

Nhưng một điều kiện tiên quyết nhỏ, hãy thêm phương thức này trước:

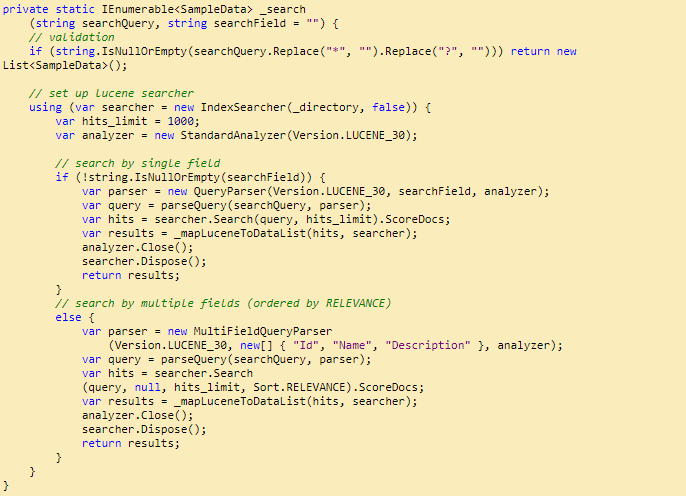
****

Hình II-23: Hàm định dạng truy vấn [42]

Phương pháp này sẽ làm gì, về cơ bản phân tích chuỗi truy vấn tìm kiếm của bạn thành đối tượng Lucene Query và nếu phân tích cú pháp không thành công, nó sẽ sửa bất kỳ ký hiệu nào gây ra lỗi và trả về đối tượng Query cố định.

Lưu ý: Nếu bạn muốn Lucene tìm các từ tương tự, bạn sẽ cần sử dụng **FuzzyQuery** thay vì Truy vấn thông thường, xem ví dụ tại đây: *Http://www.codeproject.com/Messages/4643090/Re-Find- tương tự-Words.aspx*

Và bây giờ hãy thêm phương thức tìm kiếm chính của chúng ta:

****

Hình II-24: Hàm tìm kiếm chính [43]

Tất nhiên phương thức \_search () riêng của chúng ta có nhiều điểm để được điều chỉnh và tối ưu hóa, và nó sẽ tùy thuộc vào bạn để điều chỉnh nó theo nhu cầu cụ thể của bạn.

Bạn có nhận thấy **(var hits\_limit = 1000 ; )** ? Như đã đề cập trong Bước 5, khi Lucene nhận được hơn 1000 kết quả tìm kiếm, nó trở nên ngày càng chậm, vì vậy bạn muốn giới hạn nó thành một số có liên quan trong trường hợp của bạn.

Nếu bạn thêm các trường mới vào chỉ mục của mình, đừng quên thêm tên trường để tìm kiếm vào dòng này trong nhiều trường tìm kiếm như sau:

****

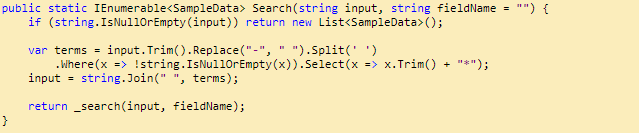
Hình II-25: Thêm trường mới vào chỉ mục [44]

Chỉ cần nhớ rằng các kết quả trả về bởi phương thức **\_search ()** sẽ được sắp xếp theo **Sort.RELEVANCE**, có nghĩa là kết quả chính xác hơn sẽ được trả về trước. Một tùy chọn khác là **Sort.INDEXORDER**, trả về kết quả theo thứ tự mà chúng đã được thêm vào chỉ mục tìm kiếm. Tuy nhiên đối với hầu hết các trường hợp, **Sort.RELEVANCE** sẽ hoạt động tốt.

Chúng ta đã thêm phương thức **private \_search ()** chính sẽ thực hiện tìm kiếm theo trường đơn hoặc nhiều trường trong chỉ mục tìm kiếm Lucene dựa trên truy vấn tìm kiếm được cung cấp.

***Bước 8*** *- Thêm phương thức công khai gọi phương thức tìm kiếm chính*

Công cụ tìm kiếm nhỏ của chúng ta được thực hiện ngay bây giờ. Bây giờ là lúc thêm các phương thức cuối cùng sẽ giao tiếp với trang web hoặc ứng dụng của chúng ta. Việc đầu tiên chỉ đơn giản là định dạng truy vấn tìm kiếm Lucene và gọi phương thức tìm kiếm riêng tư chính **\_search ():**

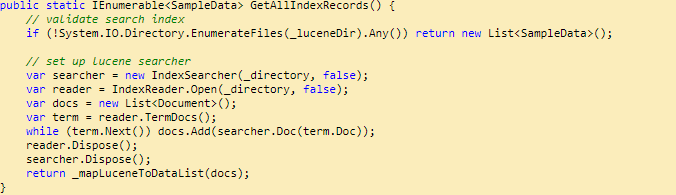
****

Hình II-26: Hàm search [45]

Nó thay thế tất cả dấu gạch ngang "-" trong yêu cầu tìm kiếm của bạn và thêm "\*" (dấu sao) sau mỗi từ để bạn có thể tìm kiếm theo từng từ. Vì vậy, về cơ bản, phương pháp Search () ở trên là khá nhiều sân chơi chính của bạn, nơi bạn tùy chỉnh querries của bạn để có được kết quả tìm kiếm hoàn hảo. Ngoài ra, để thử các hạt tìm kiếm gốc Lucene, chúng ta có thể thêm phương pháp tìm kiếm mặc định **SearchDefault ()**, không định dạng truy vấn của bạn theo bất kỳ cách nào:

 Hình II-27: Hàm SearchDefault [46]

Tìm kiếm trên internet về cách viết các cụm tìm kiếm Lucene nâng cao và chính xác, hãy thử sử dụng hàm **SearchDefault ()** , sau đó bạn có thể sửa đổi phương thức Search () để phù hợp hơn với nhu cầu của bạn. Như đã đề cập trong **Bước 5**, bây giờ chúng ta cũng đang thêm một phương thức để trả về toàn bộ chỉ mục tìm kiếm Lucene:

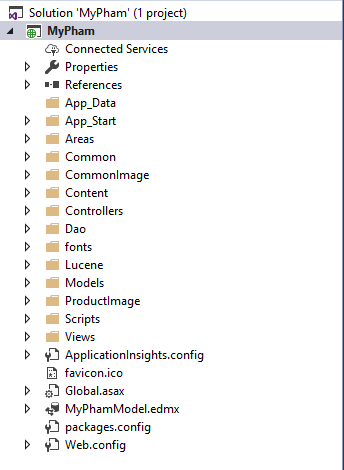
 Hình II-28: Hàm trả về toàn bộ chỉ mục tiềm kiếm [47]

Hãy nhớ, sử dụng nó một cách hợp lý, bởi vì chỉ số càng lớn thì càng mất nhiều thời gian để tải hết.

Cuối cùng thì chúng ta đã thêm hai phương pháp cho đến nay hoàn thành lớp LuceneSearch cơ bản - phương thức đầu tiên là Search (), định dạng truy vấn tìm kiếm Lucene và tìm kiếm theo tất cả các trường hoặc bởi một trường đơn lẻ. Phương thức thứ hai là **GetAllIndexRecords ()** chỉ trả về tất cả các bản ghi trong chỉ mục tìm kiếm.

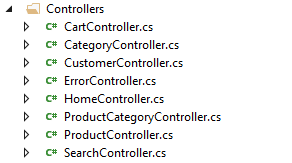
# CHƯƠNG III: ỨNG DỤNG KIẾN THỨC VÀO WEBSITE MỸ PHẨM

## Cấu trúc thư mục



Hình III-1: Cấu trúc thư mục

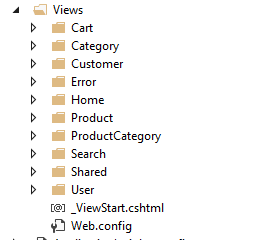
## Dùng mô hình MVC



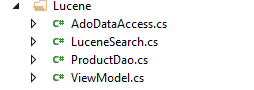
Hình III-2: Thư mục Controller

# 

Hình III-3: Thư mục Models

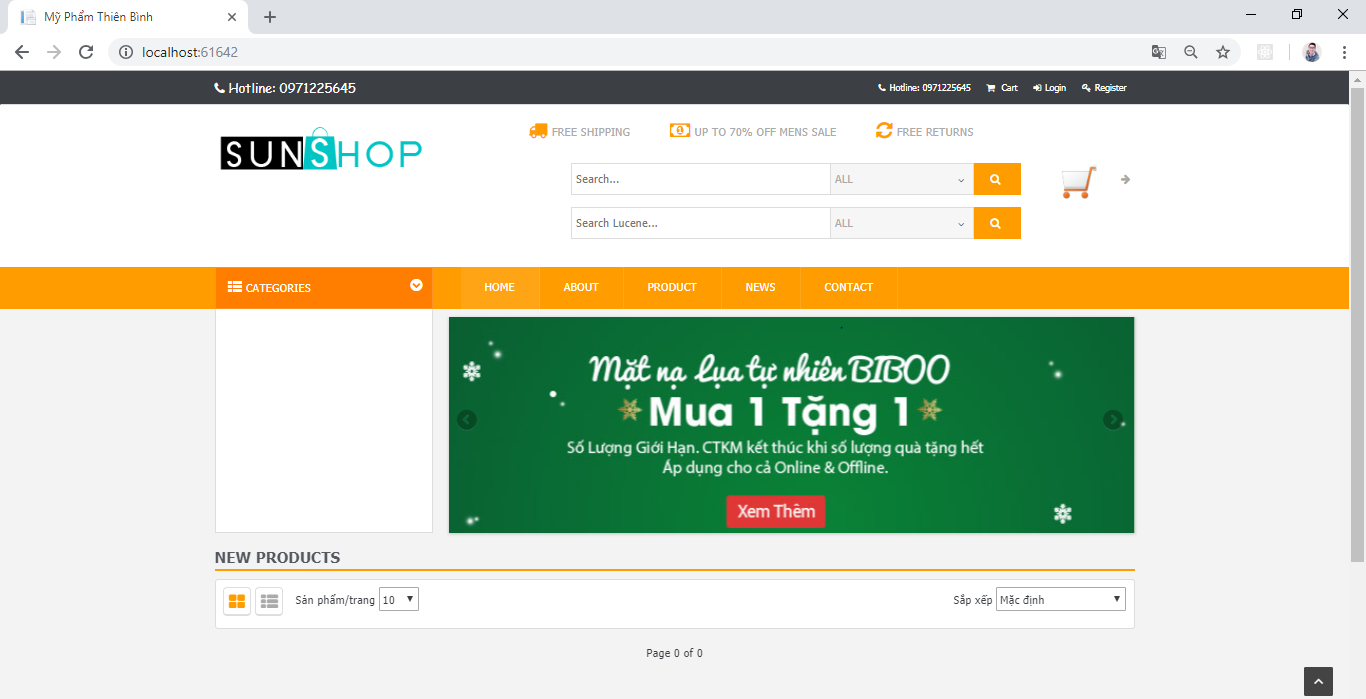


Hình III-4: Thư mục Views

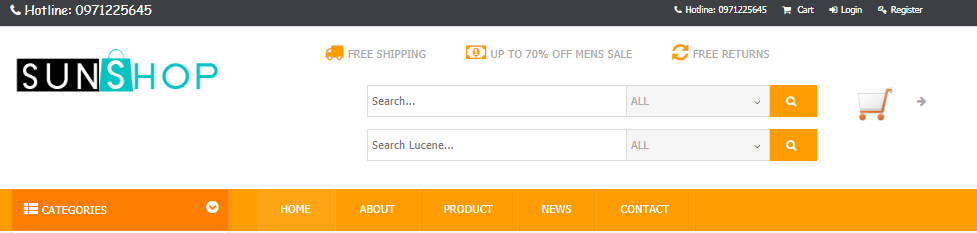


Hình III-5: Thư mục Lucene

## Sản Phẩm sau khi được chạy

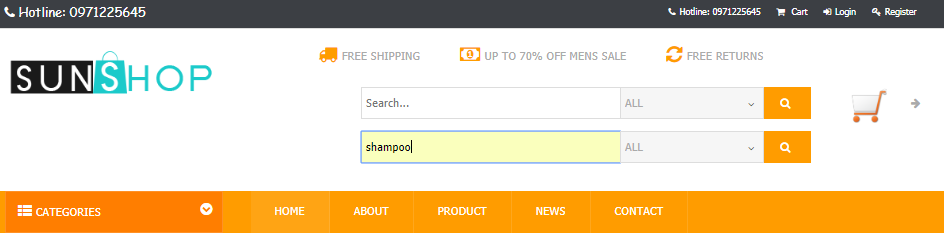


Hình III-6: Màn hình sau khi chạy demo

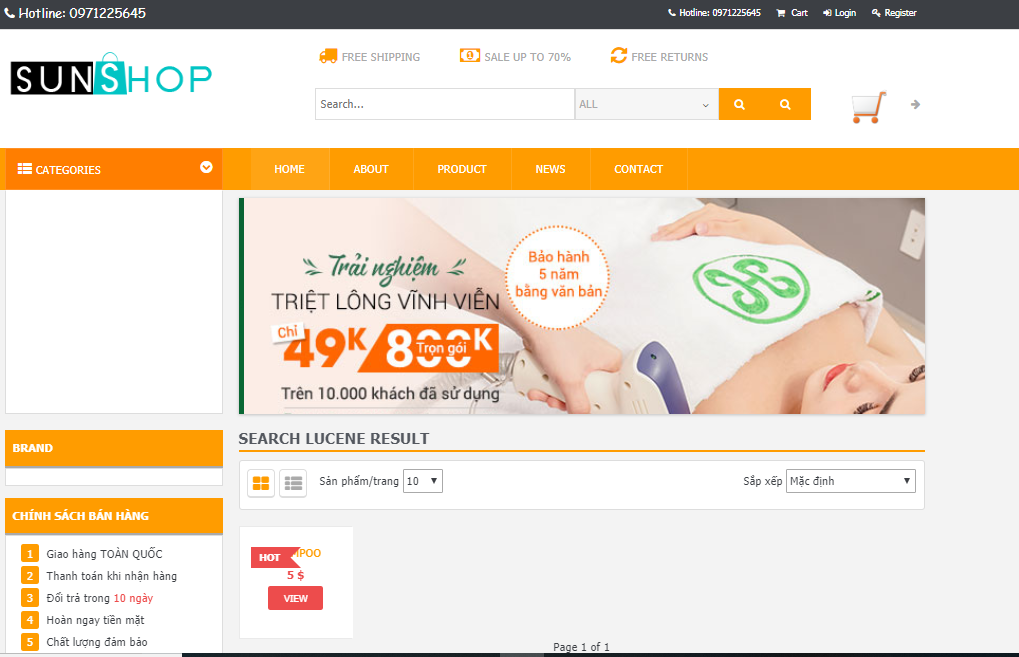


Hình III-7: Màn hình tìm kiếm sản phẩm

## Sử dụng công cụ tìm kiếm Lucene.net



Hình III-8: Sử dụng thanh search Lucene



Hình III-9: Kết quả tìm kiếm.

# CHƯƠNG IV: KẾT LUẬN

Sau khi kết thúc quá trình nghiên cứu thì nhóm đã hiện thực được ứng dụng demo là website bán mĩ phẩm theo công nghệ ASP.NET, Angular 4…. Ngoài ra nhóm cònthực hiện được việc tìm hiểu cơ bản về Search Engine và chi tiết đối với Lucene cũng như Lucene.net. Nhóm đã thực hiện được việc code và áp dụng Lucene vào trong trang web bán mỹ phẩm của nhóm mình để tối ưu hóa việc tìm kiếm cũng như thể hiện được tính ứng dụng của việc nghiên cứu.

Thế nhưng việc coding không hề đơn giản như trong quá trình nghiên cứu vào tìm hiểu bởi nhóm gặp phải khá nhiều khó khan trong việc thực hiện tính ứng dụng của Lucene (hay Lucene.net) trong việc tìm kiếm sản phẩm để xem có tốt hơn những tìm kiếm thông thường của các hệ quản trị MySQL hay SQL servervaf từ đó mới có thể có nền tảng để dựa theo và phát triển ứng dụng trong tương lai sắp tới để hoàn thiện hơn .

Trong tương lai gần , nhóm sẽ dựa theo những gì đã nghiên cứu cộng với việc so sánh các kết quả của các quá trình tìm kiếm để hoàn thiện dần việc tối ưu khi sử dụng Lucene.net .Cùng với sự phát triển hiện nay thì sau khi nhóm hoàn thành việc nghiên cứu thì không chỉ sử dụng trong nghiên cứu nữa mà có thể đem ra ngoài ứng dụng vào trong thực tế giúp cho 1 số trang web hiện nay tốt hơn trong việc tìm kiếm sản phẩm hoặc tìm kiếm theo mong muốn của khác hàng . Đồng thời cũng sẽ hoàn thiện hết các chức năng cần thiết của trang web bán hàng.

# TÀI LIỆU THAM KHẢO

**[1] – [7]**: tutorialspoint, asp.net\_environment\_setup , Nov 2018

(https://www.tutorialspoint.com/asp.net/asp.net\_environment\_setup.htm)

**[8] – [11]:** [Martin Kearn](https://blogs.msdn.microsoft.com/martinkearn), Introduction to REST and .net Web API, Nov 2018

(<https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-web-api/?fbclid=IwAR3Lkgam4ewE35qAEUZsduSLPNt9-Lgd9pdXhvPrBZVKKW4mAB7RxeTBOUo>)

**[12] – [16]:** [Martin Kearn](https://blogs.msdn.microsoft.com/martinkearn), Introduction to REST and .net Web API, Dec 2018

(https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-web-api/?fbclid=IwAR25XPzeXQi0-KcdQTtNYo-4H7-O3U2M1vzoe2OXIkK\_qnZkYMbTrCNVYkY)

**[17] – [19]:** [tuanna2704](https://viblo.asia/u/tuanna2704), Angular 4 dành cho angular 1 developer, Dec 2018

(https://viblo.asia/p/angular-4-danh-cho-angular-1-developer-yMnKM38ml7P)

**[20]-[23]:** [Paul Gil](https://www.lifewire.com/paul-gil-2482912), The Best Search Engines of 2018, Sep 2018

(<https://www.lifewire.com/best-search-engines-2483352>)

**[24]-[25]:** John Smart, Lucene: A Tutorial Introduction to Full-Text Indexing, Oct 2018

(http://www.informit.com/articles/printerfriendly.aspx?p=461633.)

**(\*)** Gospodnetic, Otis (2004). Lucene in Action. Manning Publications. ISBN 978-1-932394-28-3. Oct 2018

**[26] – [47]:** [Mikhail-T](https://www.codeproject.com/script/Membership/View.aspx?mid=8444182) , Lucene.Net ultra fast search for MVC or WebForms site , Oct 2018

(<https://www.codeproject.com/Articles/320219/Lucene-Net-ultra-fast-search-for-MVC-or-WebForms> )