**TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

-----🙞🙜🕮🙞🙜-----

****

**BÁO CÁO MÔN HỌC**

**CHUYÊN ĐỀ J2EE**

**SHOP BÁN GIÀY**

Giảng viên hướng dẫn: **Thạc sĩ Nguyễn Trác Thức**

Sinh viên thực hiện: Nhóm: Group 09

12520127 – Hoàng Ngọc Hạnh

12520122 – Cấn Hoàng Hải

13520036 – Nguyễn Quốc Bảo

**MỤC LỤC**

[1. GIỚI THIỆU HỆ THỐNG 7](#_Toc484238316)

[1.1. Giới thiệu chung 7](#_Toc484238317)

[1.2. Giới thiệu Shoes Shop 7](#_Toc484238318)

[1.3. Đặc tả chức năng 8](#_Toc484238319)

[1.3.1. Quản lý sản phẩm 8](#_Toc484238320)

[Frontend 8](#_Toc484238321)

[Backend 8](#_Toc484238322)

[1.3.2. Quản lý giỏ hàng 8](#_Toc484238323)

[Frontend 8](#_Toc484238324)

[1.3.3. Quản lý đơn đặt hàng (chức năng dành cho admin) 8](#_Toc484238325)

[Backend 8](#_Toc484238326)

[1.3.4. Quản lý tin tức 8](#_Toc484238327)

[Frontend 8](#_Toc484238328)

[Backend 9](#_Toc484238329)

[1.3.5. Quản lý bình luận 9](#_Toc484238330)

[Frontend 9](#_Toc484238331)

[1.3.6. Quản lý người dùng 9](#_Toc484238332)

[Backend 9](#_Toc484238333)

[1.3.7. Liên hệ 9](#_Toc484238334)

[Frontend 9](#_Toc484238335)

[Backend 9](#_Toc484238336)

[1.3.8. Tìm kiếm 10](#_Toc484238337)

[Frontend 10](#_Toc484238338)

[1.3.9. Quản lý khuyến mãi 10](#_Toc484238339)

[Frontend 10](#_Toc484238340)

[Backend 10](#_Toc484238341)

[2. CÔNG NGHỆ SỬ DỤNG 10](#_Toc484238342)

[1.2. Servlets 10](#_Toc484238343)

[1.3. Cấu trúc của Servlets 11](#_Toc484238344)

[1.3.1. Nhiệm vụ của Servlets 11](#_Toc484238345)

[1.3.2. Package trong Servlets 11](#_Toc484238346)

[1.3.3. Cài đặt môi trường cho Servlets 12](#_Toc484238347)

[1.3.4. Vòng đời (Life Cycle) của Servlet 15](#_Toc484238348)

[1.3.5. Phương thức init() 15](#_Toc484238349)

[1.3.6. Phương thức service() 16](#_Toc484238350)

[1.3.7. Phương thức doGet() 16](#_Toc484238351)

[1.3.8. Phương thức doPost() 16](#_Toc484238352)

[1.3.9. Phương thức destroy() 17](#_Toc484238353)

[1.4. JSP 18](#_Toc484238354)

[1.4.1. Lợi thế của JSP 18](#_Toc484238355)

[1.4.2. Tiến trình JSP 19](#_Toc484238356)

[1.4.3. Biên dịch JSP 21](#_Toc484238357)

[1.4.4. Khởi tạo JSP 22](#_Toc484238358)

[1.4.5. Thực thi JSP 22](#_Toc484238359)

[1.4.6. Hủy JSP 22](#_Toc484238360)

[1.5. Spring 23](#_Toc484238361)

[1.5.1. EJB và Spring 23](#_Toc484238362)

[1.5.2. Sự ra đời của Spring 23](#_Toc484238363)

[1.5.3. Spring Core 24](#_Toc484238364)

[1.5.4. Spring Bean 24](#_Toc484238365)

[1.5.5. Dependency Injection (DI) 25](#_Toc484238366)

[1.5.6. Spring Context 25](#_Toc484238367)

[1.5.7. Spring MVC 26](#_Toc484238368)

[1.5.8. Spring Data 26](#_Toc484238369)

[1.5.9. Spring Security 26](#_Toc484238370)

[1.5.10. Spring Boot 26](#_Toc484238371)

[1.5.11. Spring Batch 26](#_Toc484238372)

[1.5.12. Spring Integration 26](#_Toc484238373)

[1.5.13. Spring XD 26](#_Toc484238374)

[1.5.14. Spring Social 26](#_Toc484238375)

[1.6. Hibernate 26](#_Toc484238376)

[1.6.1 Giới thiệu về Hibernate 26](#_Toc484238377)

[1.6.2 Hibernate Framework là gì? 27](#_Toc484238378)

[1.6.3 Hibernate Framework có các lợi ích 27](#_Toc484238379)

[1.6.4 Database được hỗ trợ 28](#_Toc484238380)

[1.6.5 Các công nghệ được hỗ trợ 28](#_Toc484238381)

[1.6.6 Kiến trúc Hibernate 28](#_Toc484238382)

[1.6.7 Các thành phần trong kiến trúc Hibernate 30](#_Toc484238383)

[1.6.8 Đối tượng Configuration 30](#_Toc484238384)

[1.6.9 Đối tượng SessionFactory 31](#_Toc484238385)

[1.6.10 Đối tượng Session 31](#_Toc484238386)

[1.6.11 Đối tượng Transaction 31](#_Toc484238387)

[1.6.12 Đối tượng Query 31](#_Toc484238388)

[1.6.13 Đối tượng Criteria 31](#_Toc484238389)

[3. Thiết kế, CÀI ĐẶT VÀ THỬ NGHIỆM 32](#_Toc484238390)

[3.1 Kiến trúc hệ thống 32](#_Toc484238391)

[3.1.1 Sitemap 33](#_Toc484238392)

[3.1.2 Deploy diagram 34](#_Toc484238393)

[3.2 Sơ đồ Use-case diagram 34](#_Toc484238394)

[3.3 Sơ đồ Sequence diagram 35](#_Toc484238395)

[3.3.1. Xem Sản phẩm 35](#_Toc484238396)

[3.3.2. Xem chi tiết sản phẩm 35](#_Toc484238397)

[3.3.3. Checkout 36](#_Toc484238398)

[3.4 Giao diện ứng dụng 36](#_Toc484238399)

[3.4.1 Danh sách các màn hình 36](#_Toc484238400)

[3.4.2 Trang Chủ 37](#_Toc484238401)

[3.4.3 Giới thiệu 37](#_Toc484238402)

[3.4.4 Sản phẩm 39](#_Toc484238403)

[3.4.5 Khuyến mãi 40](#_Toc484238404)

[3.4.6 Tin Tức 41](#_Toc484238405)

[3.4.7 Liên Hệ 42](#_Toc484238406)

[3.4.8 Chi tiết sản phẩm 43](#_Toc484238407)

[3.4.9 Giỏ hang 44](#_Toc484238408)

[3.4.10 Tìm kiếm 44](#_Toc484238409)

[3.5 Giao diện admin 44](#_Toc484238410)

[3.5.1 Danh sách màn hình 44](#_Toc484238411)

[3.5.2 Danh sách sản phẩm 45](#_Toc484238412)

[3.5.3 Thêm mới sản phẩm 46](#_Toc484238413)

[3.5.4 Danh sách đơn hang 46](#_Toc484238414)

[3.5.5 Danh Sách bài viết 47](#_Toc484238415)

[3.5.6 Thêm mới bài viết 47](#_Toc484238416)

[3.5.7 Danh sách liên hệ 48](#_Toc484238417)

[3.5.8 Người dùng 48](#_Toc484238418)

[3.5.9 Thêm mới người dùng 49](#_Toc484238419)

[3.5.10 Trang login 49](#_Toc484238420)

[4. PHÂN CÔNG CÔNG VIỆC 50](#_Toc484238421)

**NHẬN XÉT**

# GIỚI THIỆU HỆ THỐNG

## Giới thiệu chung

Kinh doanh online là một xu hướng đã phổ biến từ lâu. Bước qua mỗi năm thì những hình thức kinh doanh bán hàng online lại có một phương thức, một chiến lược phát triển khác nhau. Bước sang năm 2017 thì hình thức này quá thịnh hành và đơn giản với các cá nhân, tổ chức.

Chỉ đơn giản bản thân mỗi người cũng có thể tự đăng sản phẩm/dịch vụ nhỏ lẻ lên tường Facebook, chia sẻ rao vặt, diễn đàn, giới thiệu đến bạn bè và người quen là có thể bán được hàng. Nhưng để tăng được doanh số hơn nữa và để tồn tại lâu dài trên Internet thì cần phải có những chiến lược cực kỳ hiệu quả. Và một trong số đó là xây dựng một website bán hàng trực tuyến.

Một giao diện website đẹp, bắt mắt: hình ảnh sản phẩm sắc nét, nội dung được phân bổ đồng đều, những điều này sẽ giúp khách hàng ở lại trang web lâu hơn, bản thân sẽ có một ấn tượng tốt về doanh nghiệp. Vì vậy **nhóm G09** đã tiến hành xây dựng hệ thống bán đồ thể thao trực tuyến **Shoes Shop** đề minh chứng cho điều đó.

## Giới thiệu Shoes Shop

**Shoes Shop** được triển khai với mong muốn cung cấp cho người dùng một kênh mua sắm trực tuyến tiện lợi, nơi khách hàng có thể chọn lựa những trang phục, phụ kiện, dụng cụ thể thao yêu thích. Các mặt hàng kinh doanh chính của **Shoes Shop** bao gồm các loại giày (giày bóng đá, giày quần vợt, giày Nike

**Shoes Shop** được thiết kế khá đơn giản nhưng cũng không kém phần bắt mắt, phục vụ nhu cầu mua hàng trực tuyến của khách hàng. Đến với **Shoes Shop**, khách hàng có thể dễ dàng xem xét các mặt hàng mình cần đã được sắp xếp thành các chuyên mục (áo thể thao, giày thể thao, dụng cụ thể thao). Nếu khách hàng yêu thích một sản phẩm nào đó thì có thể xem chi tiết sản phẩm đó và **Shoes Shop** hỗ trợ tính năng *giỏ hàng* để khách hàng có thể thỏa sức lưu lại những sản phẩm mà mình muốn mua. Trong *giỏ hàng* khách hàng có thể thực hiện *thanh toán* bằng hình thức đặt hàng (điền thông tin cá nhân của mình và gửi cho cửa hàng). Khách hàng không cần phải đến trực tiếp cửa hàng để lấy hàng, dựa trên đơn đặt hàng, cửa hàng sẽ giao hàng đến tận nơi. Nếu có ý kiến góp ý gì cho cửa hàng, khách hàng có thể sử dụng kênh liên hệ trên **Shoes Shop**, các ý kiến của khách hàng sẽ nhanh chóng được cửa hàng tiếp thu và điều chỉnh một cách kịp thời. Nắm bắt được thị hiếu của khách hàng, **Shoes Shop** cũng sẽ thường xuyên cập nhật các thông tin về các mặt hàng giảm giá thông qua kênh *khuyến mãi* và các thông tin, bài viết liên quan đến các sản phẩm của cửa hàng cũng sẽ được đăng tải thường xuyên qua kênh *tin tức*.

## Đặc tả chức năng

### Quản lý sản phẩm

#### Frontend

* Trang chủ cần hiển trị 8 sản phẩm mới nhất, 4 sản phẩm nổi bật (là những sản phẩm được xem nhiều nhất), 4 sản phẩm mua nhiều nhất.
* Trang hiển thị danh sách sản phẩm theo category
* Trang xem chi tiết sản phẩm: có nút share qua Facebook, Google+

#### Backend

* Thêm sản phẩm mới
* Sửa thông tin sản phẩm
* Xóa sản phẩm

### Quản lý giỏ hàng

#### Frontend

* Hiển thị danh sách sản phẩm trong giỏ hàng, bao gồm các thuộc tính: ảnh sản phẩm (cỡ nhỏ), tên sản phẩm, đơn giá, số lượng, tổng tiền
* Thêm sản phẩm vào giỏ hàng
* Xóa sản phẩm khỏi giỏ hàng
* Thanh toán: khi ấn nút thanh toán thì chuyển sang trang cho phép người dùng nhập các thông tin đặt hàng để cửa hàng thực hiện giao hàng. Thông tin đặt hàng bao gồm: họ, tên, tên công ty, email, số điện thoại, địa chỉ, mã bưu điện, tỉnh / thành phố, ghi chú

### Quản lý đơn đặt hàng (chức năng dành cho admin)

#### Backend

* Hiển thị danh sách đơn đặt hàng
* Thêm mới đơn đặt hàng
* Sửa thông tin đơn đặt hàng
* Xóa đơn đặt hàng

### Quản lý tin tức

#### Frontend

* Hiển thị danh sách các tin tức của shop (được phân trang theo từng nhóm 10 tin)

#### Backend

* Hiển thị danh sách tin tức, bao gồm: tiêu đề, trạng thái (published, draft), tác giả, ngày tạo
* Thêm một tin mới: cấu trúc một tin bao gồm: tiều đề, nội dung – cung cấp editor để người dùng dễ soạn thảo
* Sửa thông tin bài viết
* Xóa bài viết

### Quản lý bình luận

#### Frontend

* Sử dụng plugin comment của Facebook để bình luận cho tin tức

### Quản lý người dùng

#### Backend

Hiển thị danh sách người dùng

Các thông tin hiển thị bao gồm: avatar, họ và tên, username, email, địa chỉ, số điện thoại, loại loại người dùng, ghi chú.

Người dùng của hệ thống có 2 loại:

* admin: có toàn quyền trong hệ thống
* subadmin: không có chức năng quản lý người dùng

Các chức năng:

* Thêm mới người dùng
* Sửa thông tin người dùng
* Xóa người dùng khỏi hệ thống
* Xem thông tin chi tiết của người dùng

### Liên hệ

#### Frontend

* Hiển thị trang liên hệ
* Cho phép người dùng nhập các thông tin liên hệ, bao gồm: họ và tên, điện thoại, email, nội dung liên hệ

#### Backend

* Hiển thị danh sách các liên hệ
* Xem chi tiết liên hệ

### Tìm kiếm

#### Frontend

* Tìm kiếm sản phẩm mong muốn: hiển thị sản phẩm theo từ khóa mà người dùng nhập vào

### Quản lý khuyến mãi

#### Frontend

* Hiển thị các sản phẩm khuyến mãi

#### Backend

* Thêm thông tin khuyến mãi
* Sửa thông tin khuyến mãi
* Xóa thông tin khuyến mãi

# CÔNG NGHỆ SỬ DỤNG

## Servlets

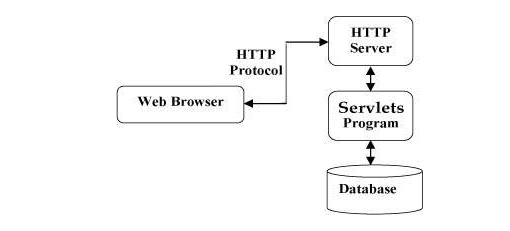
Java Servlets là các chương trình chạy trên một Web server hoặc một Application server và thực hiện như là một tầng trung gian giữa một Yêu cầu từ một trình duyệt web hoặc HTTP client với các Database hoặc các ứng dụng trên HTTP server.

Sử dụng Servlets, bạn có thể thu thập Input từ người sử dụng thông qua các form trên trang web, hiển thị các bản ghi (record) từ một Database hoặc từ nguồn khác, và tạo các trang web động.

Java Servlets thường có chung mục đích: là các chương trình độc lập bởi việc sử dụng Common Gateway Interface (CGI). Nhưng so với CGI, thì Servlets có các lợi thế sau:

* Hiệu năng tốt hơn đáng kể.
* Servlets thực thi bên trong không gian địa chỉ của một Web server, không cần thiết phải tạo một tiến trình riêng biệt để xử lý mỗi yêu cầu từ Client.
* Servlets là độc lập trên nền tảng bởi vì chúng được viết bằng Java.
* Bởi vì viết bằng Java (mà Java cung cấp tình bảo mật cao trên Server) nên Servlets là đáng tin cậy.
* Tính năng đầy đủ của thư viện của các lớp trong Java là luôn luôn có sẵn cho Servlets. Nó có thể giao tiếp với Applet, Database hoặc phần mềm khác thông qua các Socket và kỹ thuật RMI mà bạn đã từng thấy.

## Cấu trúc của Servlets



### Nhiệm vụ của Servlets

Servlets thực hiện các tác vụ chủ yếu sau:

* Đọc dữ liệu hiển thị (explicit) được gửi bởi Client (hoặc trình duyệt) bao gồm một HTML Form trên một trang web hoặc nó cũng có thể từ một Applet hoặc một chương trình Custom từ HTTP Client.
* Đọc dữ liệu yêu cầu HTTP ẩn (implicit) được gửi bởi Client (hoặc trình duyệt) bao gồm cookie, các loại media.
* Xử lý dữ liệu và cho ra kết quả. Tiến trình này có thể yêu cầu Database, đang thực thi một triệu hồi tới RMI hoặc CORBA, triệu hồi một Web Service, hoặc tính toán phản hồi một cách trực tiếp.
* Gửi dữ liệu hiển thị (ví dụ: tài liệu) tới các Client (hoặc trình duyệt). Tài liệu này có thể được gửi theo nhiều định dạng khác nhau, gồm text (HTML hoặc XML), nhị phân (hình ảnh GIF), Excel, .v.v.
* Gửi phản hồi HTTP ẩn tới các Client (hoặc trình duyệt), thông báo cho trình duyệt hoặc Client về kiểu của tài liệu được trả về (ví dụ: HTML), thiết lập các Cookie và Caching các tham số, cùng các tác vụ khác.

### Package trong Servlets

Java Servlets là các lớp trong Java chạy bởi một Web Server mà có một trình thông dịch hỗ trợ Java Servlets.

Servlets có thể được tạo bởi sử dụng các gói javax.servlet và javax.servlet.http là một phiên bản mở rộng của thư viện lớp Java để hỗ trợ các dự án phát triển có phạm vi lớn. Các lớp này triển khai Java Servlet và JSP.

Java Servlet đã được tạo và được biên dịch giống như các lớp khác trong Java. Sau khi bạn cài đặt các gói servlet và thêm chúng vào Classpath trong máy của bạn, bạn có thể biên dịch Servlet với bộ biên dịch JDK hoặc bất kỳ bộ biên dịch nào khác.

Giống như bất kỳ chương trình Java nào khác, bạn cần biên dịch một servlet bởi sử dụng trình biên dịch javac của Java và sau khi biên dịch ứng dụng servlet này, nó sẽ được triển khai trong một môi trường để chạy.

### Cài đặt môi trường cho Servlets

Cài đặt Java Development Kit

Tại bước này, bạn tải sự triển khai của Java Software Development Kit (SDK) và cài đặt biến môi trường PATH một cách thích hợp.

Bạn có thể tải SDK từ: [Java SE Downloads](http://www.oracle.com/technetwork/java/javase/downloads/index.html).

Sau khi bạn tải xong, theo các chỉ dẫn đã cho để cài đặt và định cấu hình cho nó. Cuối cùng, thiết lập các biến môi trường PATH và JAVA\_HOME để tham chiếu tới thư mục mà chứa java và javac, tương ứng là java\_install\_dir/bin và java\_install\_dir.

Với Windows và cài đặt SDK trong C:\jdk1.5.0\_20, bạn sẽ đặt dòng sau trong C:\autoexec.bat file của bạn.



Trên Windows NT/2000/XP, bạn cũng có thể nhấp chuột phải vào My Computer, chọn Properties, sau đó là Advanced, và Environment Variables. Sau đó, bạn cập nhật giá trị PATH và nhấn OK.

Trên Unix (Solaris, Linux, .v.v.), nếu cài đặt SDK trong /usr/local/jdk1.5.0\_20, và bạn sử dụng C Shell, thì bạn đặt dòng sau vào .cshrc file của bạn.



Nếu bạn sử dụng một IDE (viết tắt của Integrated Development Environment) như Borland JBuilder, Eclipse, IntelliJ IDEA, hoặc Sun ONE Studio, thì bạn biên dịch và thực thi một chương trình đơn giản để xác nhận rằng IDE biết nơi bạn đã cài đặt Java.

Cài đặt Web Server: Tomcat

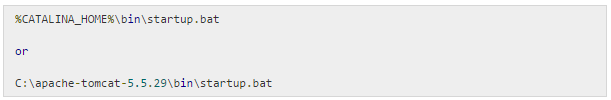
Một số Web server hỗ trợ các Servlet là có sẵn trên thị trường. Tomcat là một trong số các Web server miễn phí.

Apache Tomcat là một triển khai phần mềm Java Servlet và các công nghệ JavaServer Pages mã nguồn mở và có thể hoạt động như một Standalone Server để kiểm tra các servlet và có thể được tích hợp với Apache Web Server. Sau đây là các bước để cài đặt Tomcat trên máy của bạn:

Tải phiên bản mới nhất của Tomcat từ: <http://tomcat.apache.org/>.

Mở và đặt nó vào trong vị trí thích hợp. Ví dụ: trong C:\apache-tomcat-5.5.29 trên Windows, hoặc trong /usr/local/apache-tomcat-5.5.29 trên Linux/Unix và tạo biến môi trường CATALINA\_HOME chỉ tới các vị trí này.

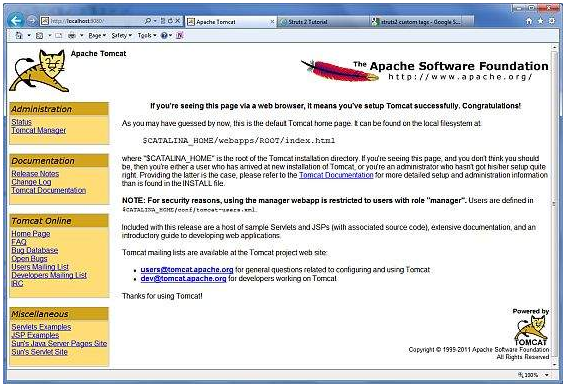
Tomcat có thể được bắt đầu bởi thực thi lệnh sau trên các thiết bị Windows:



Hoặc các lệnh sau trên Unix (Solaris, Linux, .v.v.):



Sau khi cài đặt, các ứng dụng web mặc định bao gồm Tomcat sẽ là có sẵn bởi http://localhost:8080/. Nếu mọi thứ là hoàn hảo, thì nó sẽ hiển thị:



Thông tin bổ sung về định hình và chạy Tomcat có thể được tìm thấy trong: http://tomcat.apache.org

Tomcat có thể bị dừng bởi thực thi các lệnh sau trên thiết bị Windows:



Hoặc lệnh sau trên Unix (Solaris, Linux, .v.v.):



Thiết lập CLASSPATH

Khi các servlet chưa là một phần của Java Platform, Standard Edition, bạn phải đồng nhất các lớp servlet tới bộ biên dịch.

Nếu bạn đang chạy trên Windows, bạn cần đặt các dòng sau vào trong C:\autoexec.bat file.



Trên Windows NT/2000/XP, bạn cũng có thể nhấp chuột phải vào My Computer, chọn Properties, sau đó là Advanced, và Environment Variables. Sau đó, bạn cập nhật giá trị CLASSPATH và nhấn OK.

Trên Unix (Solaris, Linux, .v.v.), nếu bạn đang sử dụng C Shell, bạn đặt các dòng sau vào trong .cshrc file của bạn:



Chú ý: Giả sử thư mục phát triển của bạn là C:\ServletDevel (Windows) hoặc /usr/ServletDevel (Unix) thì bạn cần thêm các thư mục này trong CLASSPATH theo cách tương tự như bạn đã làm ở trên.

### Vòng đời (Life Cycle) của Servlet

Vòng đời của Servlet có thể được định nghĩa như là một tiến trình đầy đủ từ khi được tạo ra đến khi bị hủy. Một Servlet theo các giai đoạn sau:

* Servlet được khởi tạo bởi gọi phương thức init().
* Servlet gọi phương thức service() để xử lý một yêu cầu từ Client.
* Servlet bị hủy bởi triệu hồi phương thức destroy().
* Cuối cùng, servlet trở thành rác và được thu thập bởi Garbage Collector của JVM.

Tiếp theo chúng ta đi vào chi tiết từng phương thức trong vòng đời của Servlet:

### Phương thức init()

Phương thức được thiết kế để chỉ được gọi một lần. Nó được gọi khi Servlet lần đầu được tạo, và không được gọi lại cho mỗi yêu cầu của người dùng. Vì thế, nó được sử dụng cho các khởi tạo one-time, giống như phương thức init của Applet.

Thường thì, Servlet được tạo khi một người dùng lần đầu triệu hồi một URL tương ứng với Servlet đó, nhưng bạn cũng có thể xác định rằng Servlet này được tải khi Server được khởi động lần đầu.

Khi một người sử dụng triệu hồi một Servlet, một đối tượng đơn của Servlet được tạo, với mỗi yêu cầu từ người dùng, là kết quả trong một thread mới, mà được thao tác tới doGet hoặc doPost một cách thích hợp. Phương thức init() tạo hoặc tải một vài dữ liệu mà sẽ được sử dụng qua vòng đời của Servlet đó.

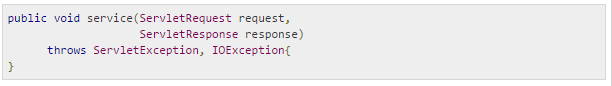
Định nghĩa phương thức init() như sau:



### Phương thức service()

Phương thức service() là phương thức chính để thực hiện tác vụ thực sự. Nơi chứa servlet (ví dụ như Web Server) gọi phương thức service() để xử lý các yêu cầu từ Client (hoặc trình duyệt) và viết phản hồi đã được định dạng trở lại Client đó.

Mỗi khi Server nhận một yêu cầu cho một Servlet, thì Server tạo một Thread mới và triệu hồi service(). Phương thức service() kiểm tra kiểu yêu cầu HTTP (Kiểu GET, POST, PUT, DELET, .v.v.) và gọi các phương thức doGet, doPost, doPut, doDelete .v.v. tương ứng một cách thích hợp.



Phương thức service() được gọi bởi Container và nó triệu hồi các phương thức doGet, doPost, doPut, doDelet, .v.v. tương ứng. Vì thế, bạn không phải làm gì cả với service() nhưng việc bạn ghi đè phương thức hoặc doGet() hoặc doPost tùy thuộc vào kiểu yêu cầu mà bạn nhận từ Client.

Hai phương thức doGet() và doPost() được sử dụng thường xuyên nhất với mỗi service. Sau đây là chi tiết về hai phương thức này:

### Phương thức doGet()

Một yêu cầu GET, là kết quả từ một yêu cầu chuẩn cho URL hoặc từ một HTML form, mà không có PHƯƠNG THỨC nào được xác định và nó nên được xử lý bởi phương thức doGet().



### Phương thức doPost()

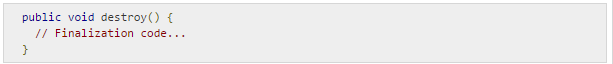
Một yêu cầu POST, là kết quả từ một HTML form mà liệt kê POST như là PHƯƠNG THỨC, và nên được xử lý bởi phương thức doPost():



### Phương thức destroy()

Phương thức destroy() chỉ được gọi một lần ở giai đoạn cuối trong vòng đời Servlet. Phương thức này giúp servlet của bạn một cơ hội để đóng các kết nối tới Database, dừng thread, viết các danh sách cookie hoặc viết tính toán trên đĩa, và thực hiện các hoạt động cleanup khác.

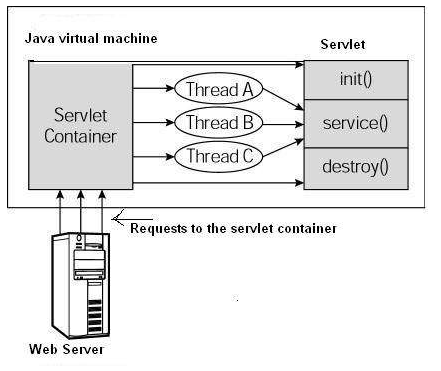
Sau khi phương thức destroy() được gọi, đối tượng servlet này được đánh dấu cho Garbage Collector. Phương thức này trông giống như sau:



Sơ đồ cấu trúc vòng đời của Servlet

Sơ đồ sau miêu tả các giai đoạn trong vòng đời của một Servlet đặc trưng:

* Đầu tiên, các HTTP Request tới Server và được đưa tới Container của Servlet.
* Container Servlet tải các servlet trước khi gọi phương thức service().
* Sau đó Container Servlet xử lý nhiều yêu cầu bởi việc tạo nhiều thread, mỗi thread thực thi phương thức service() cho một đối tượng servlet đơn.



## JSP

JavaServer Pages (JSP) là một công nghệ để phát triển các Webpage mà hỗ trợ nội dung động, giúp các lập trình viên chèn java code vào trong các HTML page bằng việc sử dụng các JSP tag đặc biệt, hầu hết bắt đầu với <% và kết thúc với %>.

Một thành phần JavaServer Pages (JSP) là một loại Java Servlet, được thiết kế để thực hiện vai trò của một giao diện người dùng (User Interface) cho một ứng dụng Java Web. Lập trình viên Web viết JSPs như là các text file mà kết nối HTML code hoặc XHTML code, phần tử XHTML, và các JSP actions và câu lệnh JSP được nhúng.

Sử dụng JSP, bạn có thể thu thập input từ người dùng thông qua các Webpage Form, trình bày các record từ một Database hoặc đăng ký quyền ưu tiên của người dùng, truy cập các thành phần JavaBeans, truyền điều khiển giữa các page và chia sẻ thông tin giữa các Request, page, …

JavaServer Pages (JSP) thường hoạt động với cùng mục đích như các chương trình độc lập bởi sử dụng Common Gateway Interface (CGI). Nhưng JSP thường có một số lợi thế trong khi so sánh với CGI:

* Hiệu năng tốt hơn đáng kể bởi vì JSP cho phép nhúng các phần tử động trong chính các HTML page thay vì phải có một CGI file riêng biệt.
* JSP luôn luôn được biên dịch trước khi nó được xử lý bởi Server, không giống như CGI/Perl mà yêu cầu Server tải một trình thông dịch (Interpreter) và Target Script mỗi khi trang được yêu cầu.
* JavaServer Pages (JSP) được xây dựng ở trên cùng của Java Servlet API, vì thế, giống Servlet, JSP cũng có quyền truy cập tới tất cả Enterprise Java APIs mạnh mẽ, bao gồm JDBC, EJB, JAXP, …
* Các JSP page có thể được sử dụng để kết nối với Servlet mà xử lý Business logic, mô hình được hỗ trợ bởi Java Servlet.
* Cuối cùng, JSP là một phần toàn bộ của Java EE, một Platform đầy đủ cho các ứng dụng lớp Enterprise. Nghĩa là, JSP có thể hoạt động như là một phần trong các ứng dụng đơn giản nhất cũng như các ứng dụng phức tạp.

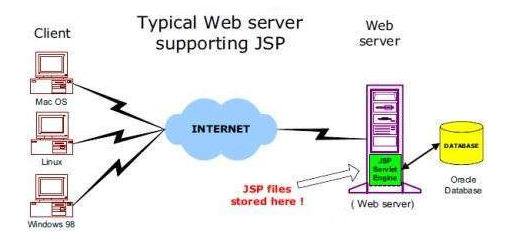
### Lợi thế của JSP

Dưới đây là các lợi thế của việc sử dụng JSP khi so sánh với các công nghệ khác:

* So với Active Server Pages (ASP): Lợi thế của JSP có thể coi là gấp đôi. Đầu tiên, các phần động được viết bằng Java, không phải bằng Visual Basic hoặc ngôn ngữ MS khác, vì thế nó mạnh mẽ hơn và dễ dàng để sử dụng hơn. Thứ hai, nó thích hợp cho các Hệ điều hành khác, không chỉ là Microsoft Web Server.
* So với Pure Servlets: Nó tiện lợi hơn khi viết (và sửa đổi) HTML, vì có nhiều lệnh printIn hơn.
* So với Server-Side Includes (SSI): SSI chỉ dành cho các thể đơn giản, không dành cho các chương trình "thực" mà sử dụng Form Data, tạo kết nối Database.
* So với JavaScript: JavaScript có thể tạo HTML động trên Client nhưng lại tương tác khó khan với Web Server để thực hiện các tác vụ phức tạp như truy cập Database và xử lý hình ảnh, …

Với Static HTML: Tất nhiên, HTML thông thường không thể chứa thông tin động.

* Web Server cần một JSP engine ví dụ: Container) để xử lý các JSP page. JSP container có nhiệm vụ chặn các Request cho các JSP page. Bài hướng dẫn này sử dụng Apache, mà có JSP container có sẵn, để hỗ trợ phát triển các JSP page.
* Một JSP container làm việc với Web Server để cung cấp môi trường runtime và các dịch vụ khác mà một JSP cần đến. Nó biết cách để hiểu các phần tử đặc biệt mà là một phần của JSP.
* Sơ đồ sau minh họa vị trí của JSP container và JSP file trong một Web Application.

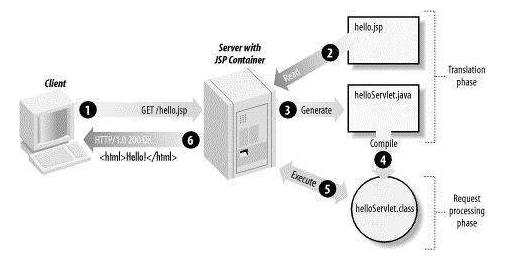


### Tiến trình JSP

Dưới đây là các bước giải thích cách Web Server tạo các Webpage bởi sử dụng JSP:

* Với một trang thông thường, trình duyệt gửi một HTTP request tới Web server.
* Web server nhận ra rằng HTTP request là cho một JSP page và chuyển nó tới một JSP engine. Điều này được thực hiện bởi sử dụng một URL hoặc một JSP page mà kết thúc với .jsp thay vì .html.
* JSP engine tải JSP page từ đĩa và biến đổi nó thành một nội dung Servlet. Sự biến đổi này là rất đơn giản, trong đó tất cả Template Text được biến đổi thành các lệnh printIn() và tất cả phần tử JSP được biến đổi thành Java code mà triển khai các hành vi động tương ứng của trang.
* JSP engine biên dịch Servlet thành một lớp có thể thực thi và chuyển Request ban đầu tới một Servlet engine.
* Một phần của Web server triệu hồi Servlet engine tải lớp Servlet và thực thi nó. Trong quá trình thực thi, Servlet này tạo một output trong định dạng HTML, mà Servlet engine truyền tới Web server bên trong một HTTP response.
* Web server chuyển HTTP Response tới trình duyệt của bạn nội dung HTML tĩnh.
* Cuối cùng, trình duyệt web xử lý trang HTML đã được tạo một cách năng động bên trong HTTP response một cách chính xác như kiểu nó là một trang tĩnh.

Tất cả các bước trên có thể được minh họa trong sơ đồ sau:



Nét đặc trưng là, JSP engine kiểm tra để biết có hay không một Servlet cho một JSP file là đã tồn tại và có hay không ngày sửa đổi JSP là cũ hơn Servlet đó. Nếu JSP là cũ hơn Servlet đã tạo của nó, thì JSP engine giả sử rằng JSP đã không thay đổi và Servlet đã được tạo đó vẫn kết nối với nội dung của JSP. Điều này làm cho tiến trình khác biệt hơn khi so với các ngôn ngữ Scripting khác (ví dụ như PHP) và vì thế nó nhanh hơn.

Trước khi hiển thị ra trình duyệt, tập tin JSP phải được biên dịch thành Servlet, dùng bộ biên dịch JSP (JSP compiler). Bộ biên dịch JSP có thể tạo servlet thành mã nguồn Java trước, rồi biên dịch mã nguồn ra tập tin .class dùng bộ biên dịch Java, hoặc có thể trực tiếp tạo mã byte code cho servlet từ trang JSP. Ngoại trừ giai đoạn phiên dịch, một JSP page được xử lý chính xác giống như một Servlet thông thường.

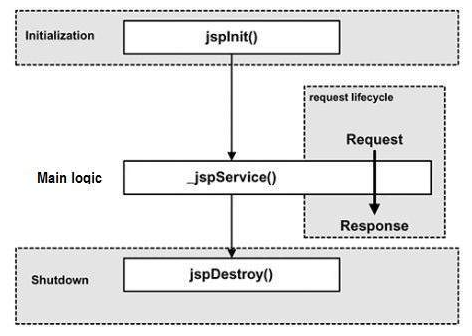
Chìa khóa để tìm hiểu tính năng tầm thấp của JSP là hiểu vòng đời của nó.

Vòng đời của JSP có thể được định nghĩa như là toàn bộ tiến trình từ khi tạo ra đến khi hủy nó, tương tự như vòng đời của một Servlet, nhưng thêm một bước để biên dịch một JSP thành Servlet.

Sau đây là các giai đoạn trong vòng đời của của một JSP:

* Biên dịch
* Khởi tạo
* Thực thi
* Hủy

4 giai đoạn chủ yếu trong vòng đời của JSP là tương tự như vòng đời của Servlet và chúng như sau:



### Biên dịch JSP

Khi môt trình duyệt yêu cầu một JSP, đầu tiên JSP engine kiểm tra để biết có hay không nó cần biên dịch trang này. Nếu trang chưa được biên dịch, hoặc nếu JSP đã được sửa đổi từ lần sửa đổi cuối cùng, thì JSP engine biên dịch trang này.

Tiến trình biên dịch gồm 3 bước:

* Parse một JSP
* Biên dịch JSP thành một Servlet
* Biên dịch Servlet đó.

### Khởi tạo JSP

Khi một container tải một JSP, nó gọi phương thức jspInit() trước khi phục vụ bất kỳ yêu cầu nào. Nếu bạn cần thực hiện sự khởi tạo JSP riêng, ghi đề phương thức jspInit():



Nét đặc trưng là, việc khởi tạo được thực hiện chỉ một lần và với phương thức int của Servlet, nói chung bạn khởi tạo các kết nối Database, mở các file, và tạo các bảng tra cứu trong phương thức jspInit.

### Thực thi JSP

Giai đoạn này trong vòng đời JSP biểu diễn tất cả các tương tác với Request tới khi JSP đó bị hủy.

Bất cứ khi nào một trình duyệt yêu cầu một JSP và trang đã được tải và được khởi tạo, thì JSP engine triệu hồi phương thức \_jspService() trong JSP đó.

Phương thức \_jspService() nhận một HttpServletRequest và một HttpServletResponse như là các tham số của nó.



Phương thức \_jspService() của một JSP được triệu hồi một lần cho mỗi yêu cầu và nó chịu trách nhiệm tạo Response cho Request đó và phương thức này cũng chịu trách nhiệm tạo các phản hồi tới tất cả 7 phương thức của HTTP, ví dụ: GET, POST, DELETE, …

### Hủy JSP

Giai đoạn hủy một JSP trong vòng đời JSP biểu thị khi nào thì một JSP bị gỡ bỏ khỏi một container.

Phương thức jspDestroy() trong JSP là phương thức hủy tương đương với trong Servlet. Ghi đè phương thức jspDestroy khi bạn cần thực hiện bất kỳ quá trình hủy nào, ví dụ như giải phóng kết nối với Database, hoặc đóng các file.

Phương thức jspDestroy() có form sau:



## Spring

### EJB và Spring

Có thể nói rằng Spring Framework ra đời là nhờ sự thành công của EJB. EJB (Enterprise JavaBeans) là một kiến trúc phát triển dành cho các ứng dụng doanh nghiệp đòi hỏi tính mạnh mẽ và khả năng mở rộng cao. EJB là một thành phần quan trọng của nền tảng Java EE. EJB được tạo ra bởi hai gã khổng lồ công nghệ là IBM và Sun vào những năm cuối thập niên 90. Dưới sự phát triển của hai tập đoàn này, EJB đã trở nên nặng nề hơn. Tiêu chuẩn EJB cũng nhanh chóng được thông qua. Sau tất cả, EJB được ca ngợi là giải pháp tốt nhất để xây dựng các ứng dụng phân tán dành cho doanh nghiệp.

Nhưng rồi những chỉ trích về chi tiết kỹ thuật của EJB đã tăng nhanh còn hơn sự nổi tiếng của nó. EJB là một framework lớn với độ phức tạp cao. Các nhà phát triền khi làm việc với EJB đã nhận ra rằng nó không hoạt động như những gì nó đã đề ra. EJB chỉ tuyệt vời trên bàn giấy, nhưng ra thực tế đó lại là một chuyện khác. Trong những phiên bản đầu tiên, các lập trình viên phải cài đặt các lớp implementation cho 2 interface khác nhau, cần qúa nhiều mã nghi thức (ceremonial code) để đáp ứng yêu cầu của EJB …

EJB vẫn được tiếp tục phát triển, dễ sử dụng hơn nhiều so với buổi ban đầu. Nhưng những chỉ trích vẫn còn đó, nỗi thất vọng trong cộng đồng Java ngày càng tăng lên, đòi hỏi cần có một giải pháp thay thế.

### Sự ra đời của Spring

Vào năm 2002, Spring Framework phát hành phiên bản đầu tiên bởi Rod Johnson. Việc xây dựng các ứng dụng doanh nghiệp trở nên đơn giản và dễ dàng hơn. Rod Johnson đã cho xuất bản một cuốn sách được phổ biến rộng rãi có nhan đề “J2EE Development without EJB”. Điều này đã cho Spring Framework rất nhiều động lực để thay thế EJB.

Ngày nay, Spring đã trở thành framework mã nguồn mở phổ biến nhất để xây dựng các ứng dụng doanh nghiệp. Cách tiếp cận thực tế ban đầu của Rod Johnson tiếp tục được phát triển và hướng tới một bộ công cụ hoàn chỉnh dành cho xây dựng các ứng dụng doanh nghiệp. Theo một số nguồn, trên 50% các ứng dụng web Java hiện nay đang sử dụng Spring.

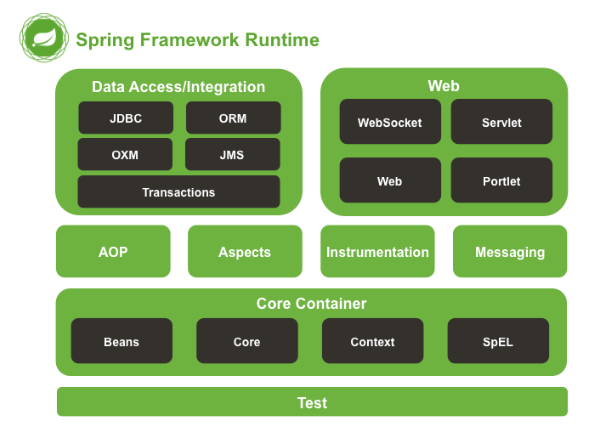
Để ngăn chặn sự phức tạp trong phát triển các ứng dụng, Spring Framework thường dựa trên các quan điểm như sau:

* Đơn giản hóa công việc phát triển thông qua việc sử dụng các đối tượng Java đơn giản hay còn được gọi là POJO (Plain Old Java Object)
* Nới lỏng ràng buộc giữa các thành phần thông qua việc sử dụng Dependency Injection và viết các interface
* Tiếp cận lập trình khai báo bằng cách sử dụng các quy tắc (convention) và các khía cạnh (aspect) chung
* Giảm thiểu các mã nghi thức và soạn sẵn (boilerplate) thông qua việc sử dụng các khuôn mẫu (template) và các khía cạnh

### Spring Core

Spring Core chính là thành phần trung tâm, cốt lõi của Spring Framework. Đây chính là nền tảng để xây dựng nên các thành phần khác.

Lược đồ dưới đây mô tả mối quan hệ giữa Spring Core với các thành phần khác trong Spring Framework:



### Spring Bean

Spring Bean là trung tâm của Spring Core và là trái tim của một ứng dụng Spring. Trái ngược với EJB, Spring Framework được thiết kế từ lõi bằng cách sử dụng các POJO hay các Spring Bean. Spring Bean có thể được hiểu là các đối tượng Java đơn giản. Điều này khá tương đồng với nguyên lý thiết kế “Đơn trách nhiệm” (single responsibility) của Robert C Martin trong lập trình hướng đối tượng.

### Dependency Injection (DI)

Dependency Injection (có thể dịch tiêm các thành phần phụ thuộc) là một sức mạnh nổi bật của Spring Framework. Trong các phiên bản gốc của EJB người ta không thấy sự xuất hiện của Dependency Injection, nhưng có lẽ do nhận thấy sự thành công từ Spring khi sử dụng Dependency Injection, các phiên bản EJB 3.x đã cung cấp các annotation để thực hiện inject các dependency. Dependency Injection là một mẫu thiết kế phần mềm mà các đối tượng phụ thuộc sẽ được inject vào một lớp nào đó. Dependency Injection là một implementation cụ thể của khái niệm Inversion of Control (đảo ngược điều khiển).

Ta sẽ có một ví dụ đơn giản về Dependency Injection như sau: Bạn có một web controller có nhiệm vụ lưu thông tin gửi từ form người dùng. Theo nguyên lý Đơn trách nhiệm, bạn không muốn lớp controller tương tác với cơ sở dữ liệu. Thay vào đó, bạn sẽ sử dụng một lớp service để làm công việc này. Như vậy, controller của bạn sẽ chỉ phải xử lý dữ liệu của form (get form data, validate data, …) rồi gọi một phưong thức của lớp service được inject để lưu dữ liệu. Controller không cần phải quan tâm hay lo lắng về kết nối cơ sở dữ liệu, pooling hay bảng nào sẽ được update. Cũng như service không cần phải biết request có những thông tin gì.

### Spring Context

Spring Context mang mọi thứ lại với nhau. Spring Context kế thừa các tính năng của Spring Bean và bổ sung các hỗ trợ cho internationalization (ví dụ như các resource bundle), event propagation, resource loading … Ngoài ra, Spring Context cũng hỗ trợ các tính năng của Java EE như EJB, JMX và truy cập từ xa cơ bản. Interface ApplicationContext là tiêu điểm của Spring Context.

Spring Expression Language (SpEL)

Spring Expression Language là một ngôn ngữ ngắn gọn giúp cho việc cấu hình Spring Framework trở nên linh hoạt hơn.

Các dự án trong Spring Framework

Trên thực tế Spring Framework là một tập hợp của nhiều dự án con. Spring Core như chúng ta đã tìm hiểu là nền tảng của các dự án trong Spring Framework. Một một dự án sẽ đảm nhận một chức năng riêng trong việc xây dựng các ứng dụng doanh nghiệp. Chúng ta sẽ cùng điểm qua các dự án lớn trong Spring.

### Spring MVC

Spring MVC được thiết kế dành cho việc xây dựng các ứng dụng nền tảng web. Đây là một dự án chúng ta không thể bỏ qua khi xây dựng các ứng dụng Java web.

### Spring Data

Cung cấp một cách tiếp cận đúng đắn để truy cập dữ liệu từ cơ sở dữ liệu quan hệ, phi quan hệ, map-reduce và thậm chí còn hơn thế nữa.

### Spring Security

Dự án này cung cấp các cơ chế xác thực (authentication) và phân quyền (authorization) cho ứng dụng của bạn.

### Spring Boot

Spring Boot là một framework giúp chúng ta phát triển cũng như chạy ứng dụng một cách nhanh chóng.

### Spring Batch

Dự án này giúp chúng ta dễ dàng tạo các lịch trình (scheduling) và tiến trình (processing) cho các công việc xử lý theo mẻ (batch job).

### Spring Integration

Spring Integration là một implementation của Enterprise Integration Patterns (EIP). Dự án này thiết kế một kiến trúc hướng thông điệp hỗ trợ việc tích hợp các hệ thống bên ngoài.

### Spring XD

Bạn đang băn khoăn về Big Data, tại sao không thử tìm hiểu Spring XD – mục tiêu của dự án này là đơn giản hóa công việc phát triển các ứng dụng Big Data.

### Spring Social

Dự án này sẽ kết nối ứng dụng của bạn với các API bên thứ ba của Facebook, Twitter, Linkedin …

## Hibernate

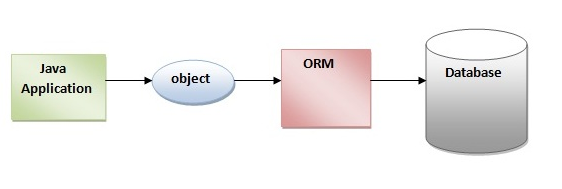
### Giới thiệu về Hibernate

Hibernate ra đời năm 2001 bởi nhà sáng lập Gavin King như một sự thay thế cho EJB2 kiểu thực thể bean. Hiện nay, phiên bản mới nhất của Hibernate là 5.2.10.Final. Seri bài hướng dẫn học Hibernate này cung cấp các khái niệm về Hibernate Framework cùng với các ví dụ đơn giản và nâng cao. Seri bài học này rất hữu ích cho người mới bắt đầu và những người đã có kinh nghiệm.

### Hibernate Framework là gì?

Hibernate framework là một giải pháp ORM (Object Relational Mapping) mã nguồn mở, gọn nhẹ. Hibernate giúp đơn giản hoá sự phát triển của ứng dụng java để tương tác với cơ sở dữ liệu.

Tool ORM giúp đơn giản hoá việc tạo ra dữ liệu, thao tác dữ liệu và truy cập dữ liệu. Đó là một kỹ thuật lập trình để ánh xạ đối tượng vào dữ liệu được lưu trữ trong cơ sở dữ liệu.



Các tool ORM sử dụng JDBC API để tương tác với cơ sở dữ liệu.

Lợi ích của Hibernate Framework

### Hibernate Framework có các lợi ích

* Mã nguồn mở và nhẹ: Hibernate Framework là mã nguồn mở có giấy phép LGPL và nhẹ.
* Hiệu suất nhanh: Hiệu suất của Hibernate Framework là nhanh bởi vì bộ nhớ cache được sử dụng trong nội bộ Hibernate Framework. Có hai loại bộ nhớ cache trong Hibernate Framework, gồm bộ nhớ cache cấp một và bộ nhớ cache cấp hai. Bộ nhớ cache cấp một được bật bằng lệnh mặc định.
* Truy vấn cơ sở dữ liệu độc lập: HQL (Hibernate Query Language) là phiên bản hướng đối tượng của SQL. Nó tạo ra các truy vấn cơ sở dữ liệu độc lập. Vì vậy, bạn không cần phải viết các truy vấn cơ sở dữ liệu cụ thể. Trước Hibernate, nếu dự án có cơ sở dữ liệu bị thay đổi, chúng ta cần phải thay đổi truy vấn SQL dẫn đến sự cố bảo trì.
* Tạo bảng tự động: Hibernate framework cung cấp phương tiện để tạo ra các bảng cơ sở dữ liệu tự động. Vì vậy, không cần phải tạo ra các bảng trong cơ sở dữ liệu bằng tay.
* Đơn giản lệnh join phức tạp: Có thể lấy dữ liệu từ nhiều bảng một cách dễ dàng với Hibernate framework.
* Cung cấp thống kê truy vấn và trạng thái cơ sở dữ liệu: Hibernate hỗ trợ bộ nhớ cache truy vấn và cung cấp số liệu thống kê về truy vấn và trạng thái cơ sở dữ liệu.

### Database được hỗ trợ

Hibernate hỗ trợ hầu hết tất cả RDBMS chính. Dưới đây là danh sách vài cơ sở dữ liệu quan hệ được hỗ trợ bởi Hibernate.

* HSQL Database Engine
* DB2/NT
* MySQL
* PostgreSQL
* FrontBase
* Oracle
* Microsoft SQL Server Database
* Sybase SQL Server
* Informix Dynamic Server

### Các công nghệ được hỗ trợ

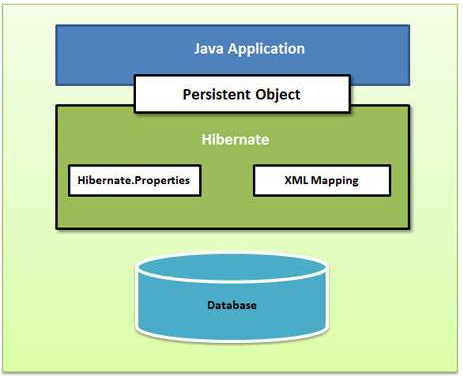
Hibernate hỗ trợ nhiều công nghệ khác, bao gồm:

* XDoclet Spring
* J2EE
* Eclipse plug-ins
* Maven

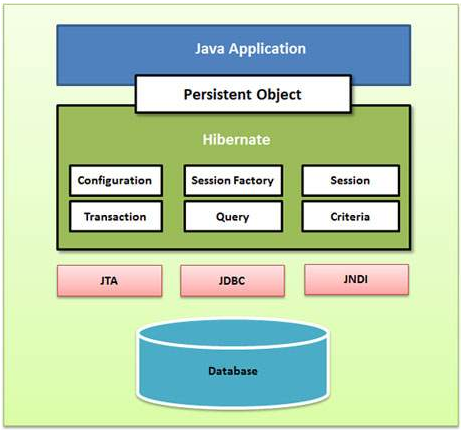
### Kiến trúc Hibernate

Kiến trúc Hibernate bao gồm nhiều đối tượng như đối tượng persistent, session factory, transaction factory, connection factory, session, transaction,…

Có 4 tầng trong kiến trúc hibernate đó là tầng ứng dụng java, tầng hibernate framework, tầng backhand api và tầng database. Dưới đây là sơ đồ kiến trúc hibernate:



Dưới đây là một hình ảnh chi tiết về Kiến trúc ứng dụng Hibernate với vài lớp core quan trọng.



Hibernate sử dụng các API Java hiện có, như JDBC, Java Transaction API (JTA) và Java Naming and Directory Interface (JNDI). JDBC cung cấp một mức độ trừu tượng cơ bản của cơ sở dữ liệu quan hệ, cho phép hầu hết các cơ sở dữ liệu với trình điều khiển JDBC được hỗ trợ bởi Hibernate. JNDI và JTA cho phép Hibernate được tích hợp với các máy chủ ứng dụng J2EE.

### Các thành phần trong kiến trúc Hibernate

Để tạo ứng dụng hibernate đầu tiên, chúng ta phải biết các thành phần của kiến trúc Hibernate. Chúng là như sau:

### Đối tượng Configuration

Đối tượng Configuration là đối tượng Hibernate đầu tiên bạn tạo trong bất kỳ ứng dụng Hibernate nào và chỉ cần tạo một lần trong quá trình khởi tạo ứng dụng. Nó đại diện cho một tập tin cấu hình hoặc thuộc tính yêu cầu của Hibernate. Đối tượng Configuration cung cấp hai thành phần chính:

* Database Connection: Thao tác này được xử lý thông qua một hoặc nhiều tệp cấu hình được Hibernate hỗ trợ. Các tệp này là hibernate.properties và hibernate.cfg.xml.
* Class Mapping Setup: Thành phần này tạo ra kết nối giữa các lớp Java và các bảng cơ sở dữ liệu.

### Đối tượng SessionFactory

Đối tượng Configuration được sử dụng để tạo ra một đối tượng SessionFactory mà cấu hình Hibernate cho ứng dụng bằng cách sử dụng tệp tin cấu hình được cung cấp và cho phép một đối tượng Session được khởi tạo. SessionFactory là một đối tượng luồng an toàn và được sử dụng bởi tất cả các luồng của một ứng dụng.

SessionFactory là đối tượng nặng nên thường nó được tạo ra trong quá trình khởi động ứng dụng và lưu giữ để sử dụng sau này. Bạn sẽ cần một đối tượng SessionFactory cho mỗi cơ sở dữ liệu bằng cách sử dụng một tập tin cấu hình riêng biệt. Vì vậy, nếu bạn đang sử dụng nhiều cơ sở dữ liệu thì bạn sẽ phải tạo nhiều đối tượng SessionFactory.

### Đối tượng Session

Một session được sử dụng để có được một kết nối vật lý với một cơ sở dữ liệu. Đối tượng Session là nhẹ và được thiết kế để được tạo ra thể hiện mỗi khi tương tác với cơ sở dữ liệu. Các đối tượng liên tục được lưu và truy xuất thông qua một đối tượng Session.

Các đối tượng Session không nên được mở trong một thời gian dài bởi vì chúng thường không phải là luồng an toàn và chúng cần được tạo ra và được đóng khi cần thiết.

### Đối tượng Transaction

Một Transaction đại diện cho một đơn vị làm việc với cơ sở dữ liệu và hầu hết các RDBMS hỗ trợ chức năng transaction. Các transaction trong Hibernate được xử lý bởi trình quản lý transaction và transaction (từ JDBC hoặc JTA).

Đây là một đối tượng tùy chọn và các ứng dụng Hibernate có thể chọn không sử dụng interface này, thay vào đó quản lý transaction trong code ứng dụng riêng.

### Đối tượng Query

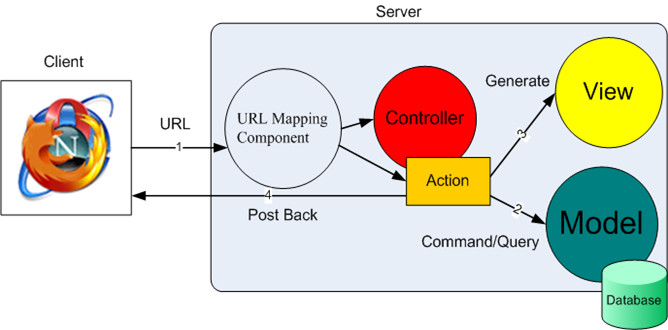
Các đối tượng Query sử dụng chuỗi truy vấn SQL hoặc Hibernate Query Language (HQL) để lấy dữ liệu từ cơ sở dữ liệu và tạo các đối tượng. Ví dụ truy vấn được sử dụng để ràng buộc các tham số truy vấn, giới hạn số lượng kết quả được trả về bởi truy vấn và cuối cùng thực hiện truy vấn.

### Đối tượng Criteria

Đối tượng Criteria được sử dụng để tạo và thực hiện truy vấn các tiêu chí định hướng đối tượng để lấy các đối tượng.

# Thiết kế, CÀI ĐẶT VÀ THỬ NGHIỆM

## Kiến trúc hệ thống



* **Models**: Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong CSDL. Ví dụ như, một đối tượng SANPHAM (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng SANPHAM ở SQL Server.
* **Views**: Views là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Trong đồ án này, nhóm đã sử dụng jsp để thực hiện lớp này.
* **Controllers**: Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng. Ví dụ, controller sẽ quản lý các dữ liệu người dùng gởi lên (query-string values) và gởi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này.

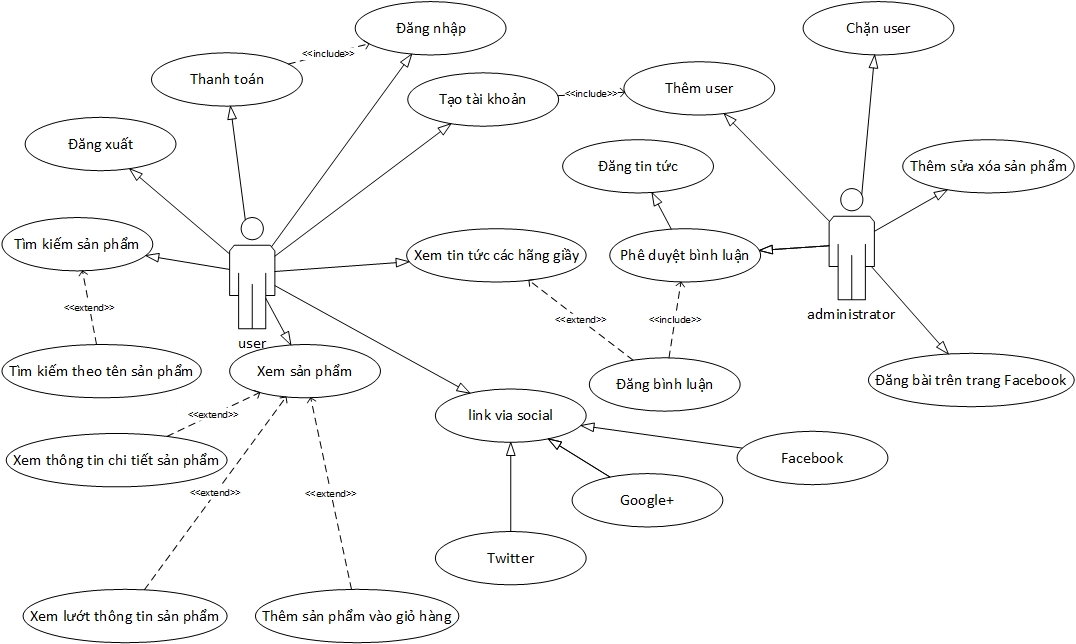
### Sitemap



### Deploy diagram

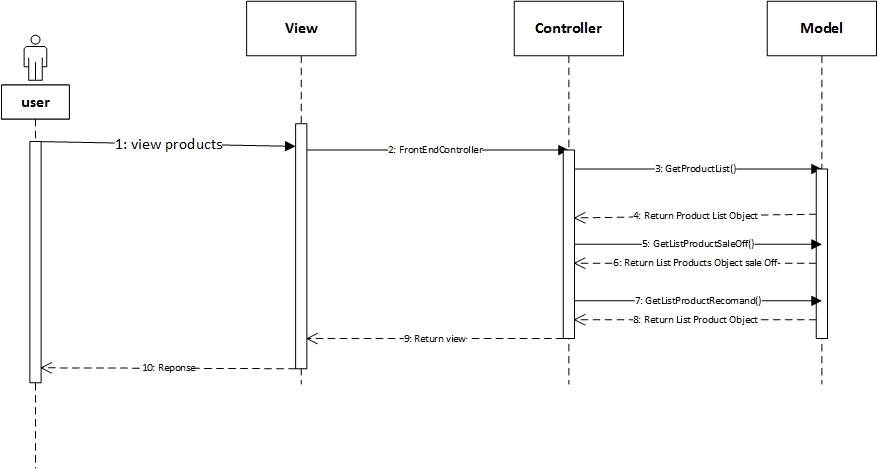


## Sơ đồ Use-case diagram

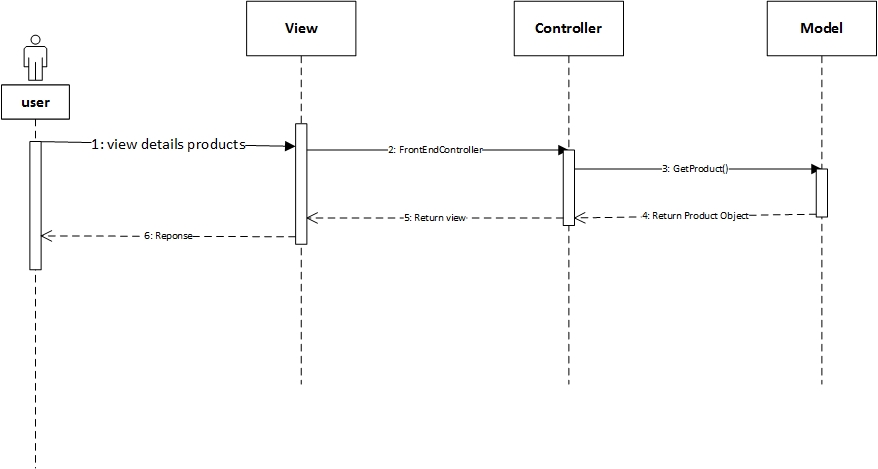


## Sơ đồ Sequence diagram

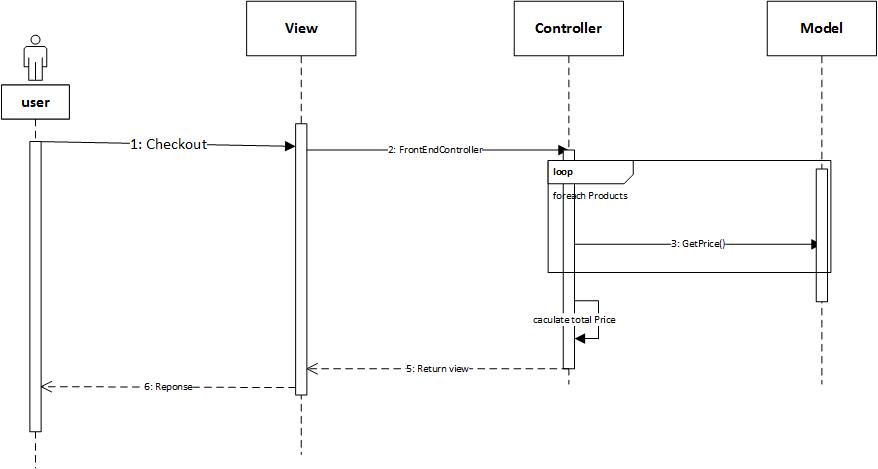
### Xem Sản phẩm



### Xem chi tiết sản phẩm



### Checkout

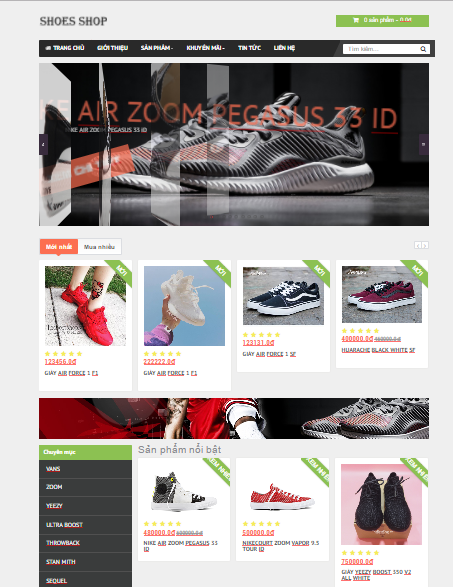


## Giao diện ứng dụng

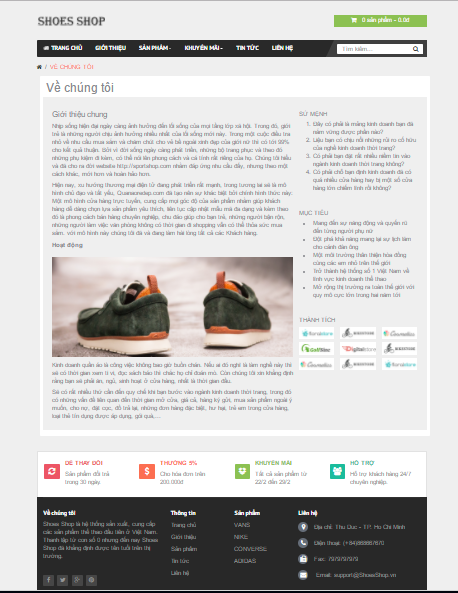
### Danh sách các màn hình

|  |  |  |
| --- | --- | --- |
| STT | Tên Màn Hình | Mô Tả |
| 1 | Trang Chủ | Hiển thị tổng quan trang web |
| 2 | Giới Thiệu | Giới thiệu thông tin và trang web |
| 3 | Sản Phẩn | Phân loại sản phẩm theo category |
| 4 | Khuyến Mãi | Hiển thị sản phẩm khuyến mãi |
| 5 | Tin Tức | Hiển thị tin tức mới nhất của trang web |
| 6 | Liên Hệ | Là site để người dùng contact với website khi cần |
| 7 | Chi Tiết Sản Phẩm | Hiển thị thông tin chi tiết sản phẩm, và các thông tin liên quan về sản phẩm đang xem |
| 8 | Giỏ hang | Hiển thị các sản phẩm được thêm vào giỏ hang |
| 9 | Tìm Kiếm | Hiển thị thông tin tìm kiếm |

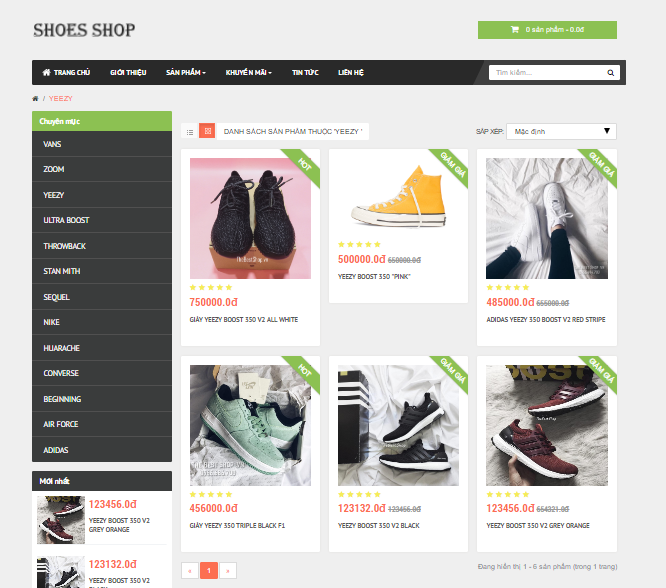
### Trang Chủ



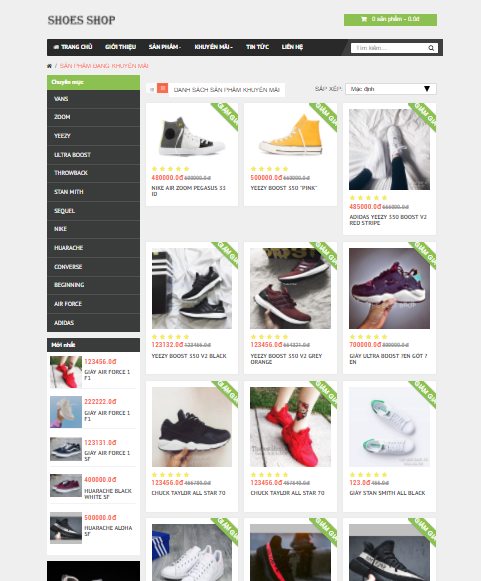
### Giới thiệu



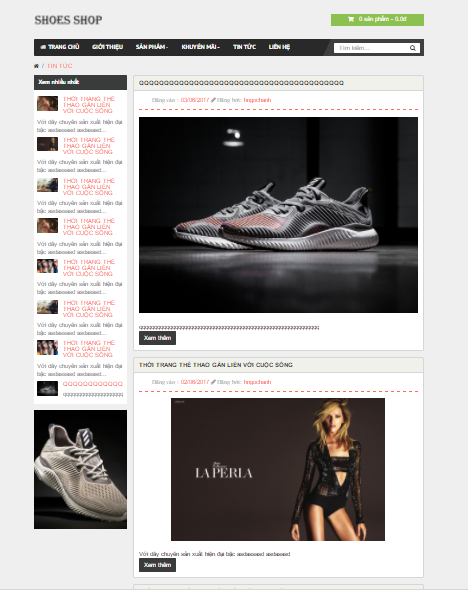
### Sản phẩm



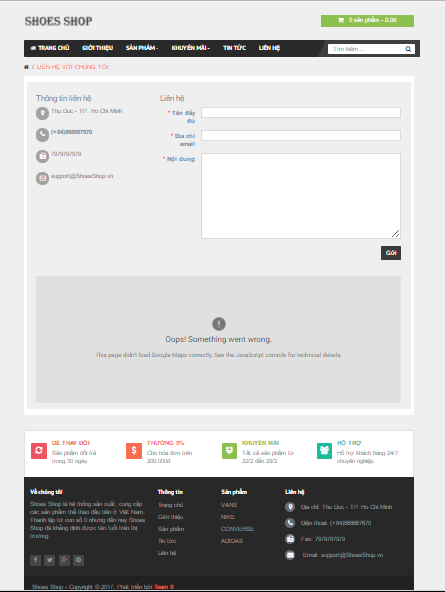
### Khuyến mãi



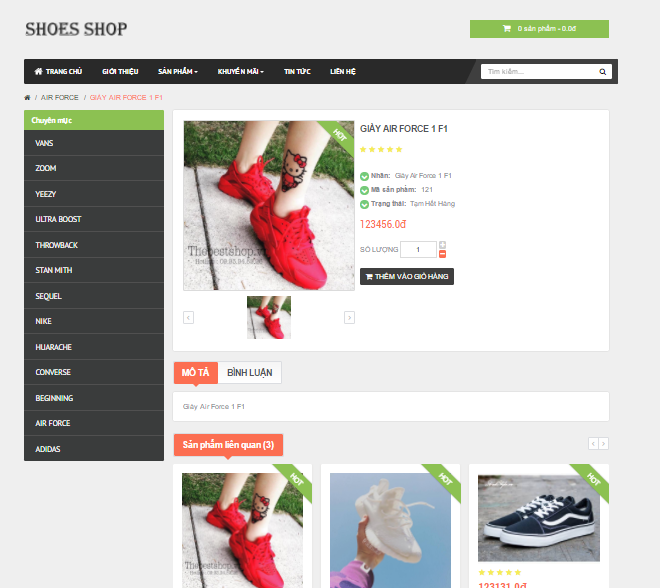
### Tin Tức



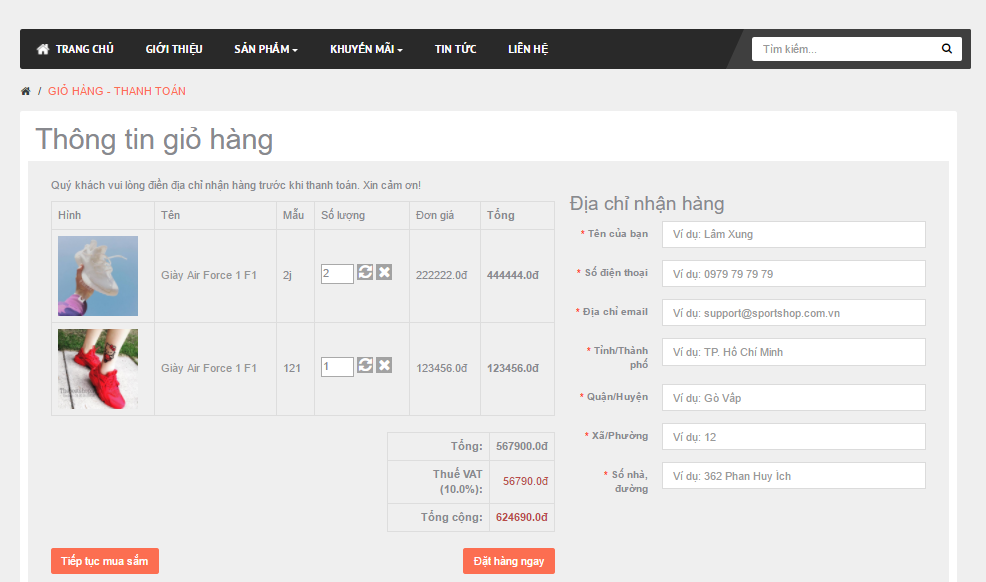
### Liên Hệ



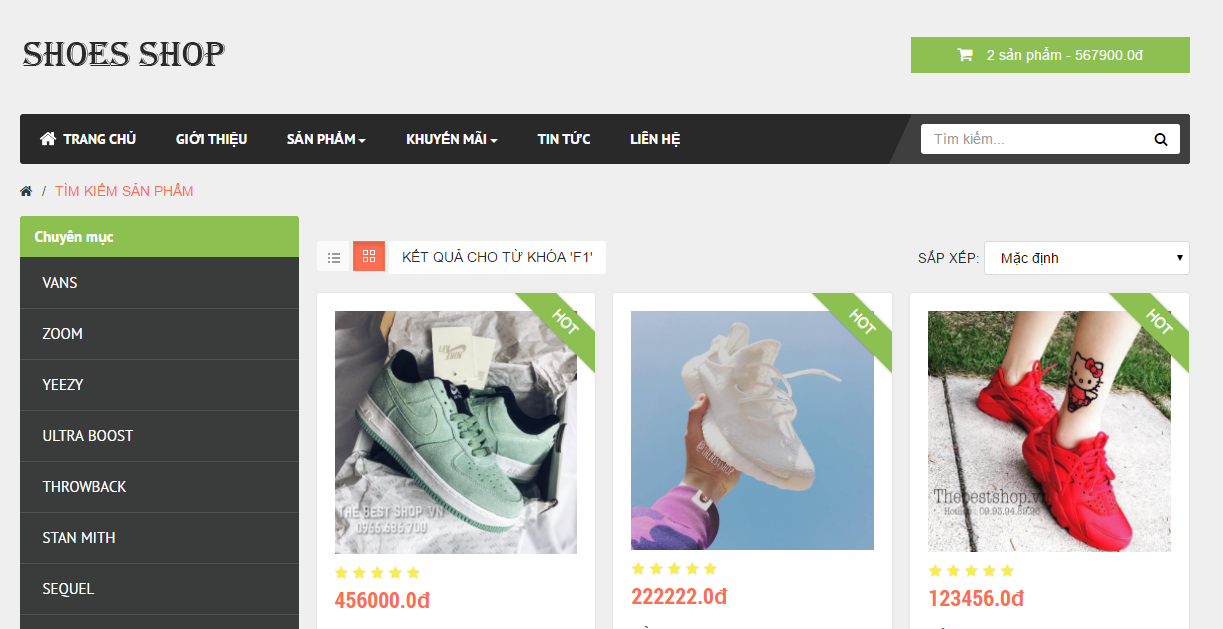
### Chi tiết sản phẩm



### Giỏ hang



### Tìm kiếm

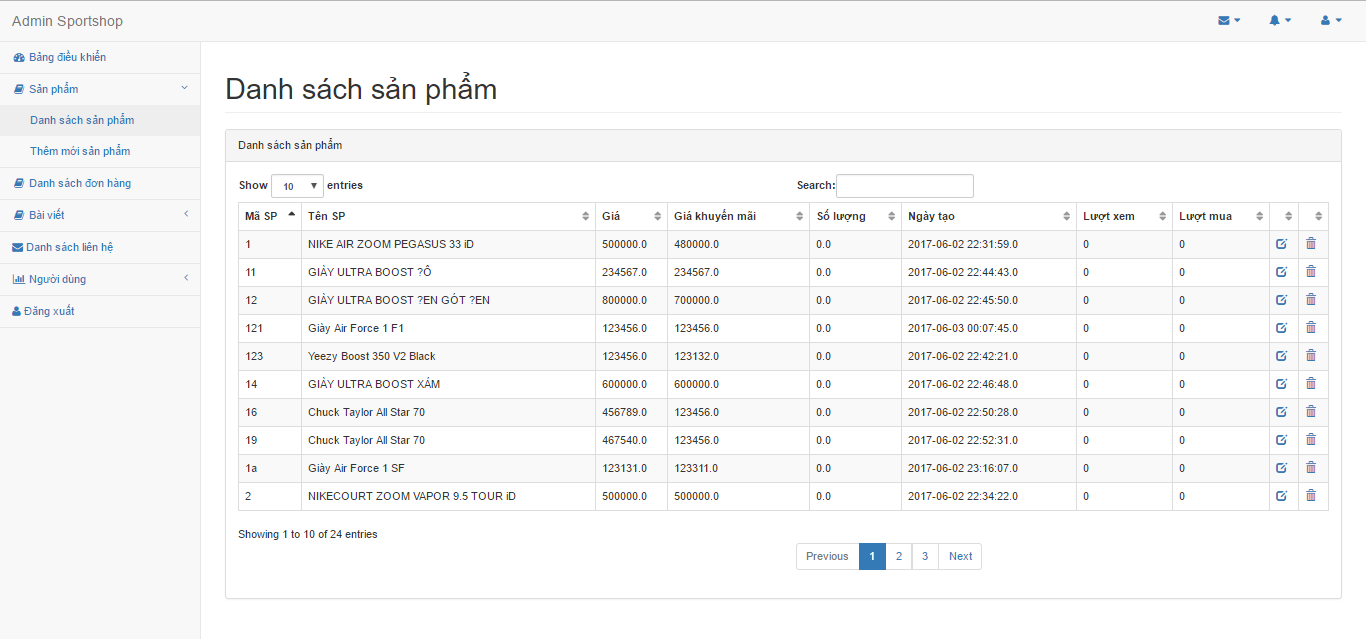


## Giao diện admin

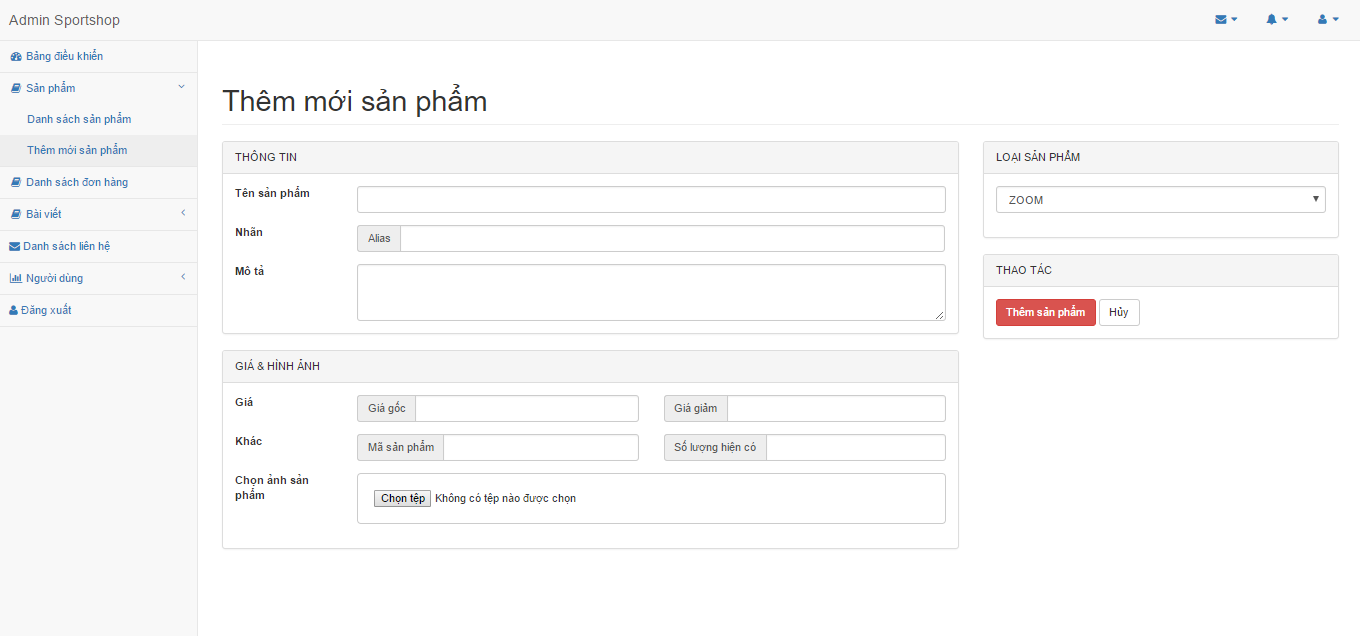
### Danh sách màn hình

|  |  |  |
| --- | --- | --- |
| STT | Tên Màn Hình | Mô Tả |
| 1 | Danh sách sản phẩm | Hiển thị tất cả sản phẩm |
| 2 | Thêm mới sản phẩm | Thêm sản phẩm vào hệ thống |
| 3 | Danh sách đơn hang | Hiển thị các yêu cầu đơn hàng từ người dùng |
| 4 | Danh Sách bài viết | Hiển thị các bài viết có trong hệ thống |
| 5 | Thêm mới bài viết | Cho phép người quản trị thêm bài viết vào hệ thống |
| 6 | Danh sách liên hệ | Hiển thị những yêu cầu liên hệ từ người dùng |
| 7 | Người dùng | Hiển thị danh sách người dùng |
| 8 | Thêm mới người dùng | Cho phép thêm user vào hệ thống |
| 9 | Trang login | Cho phép đăng nhập vào hệ thống |

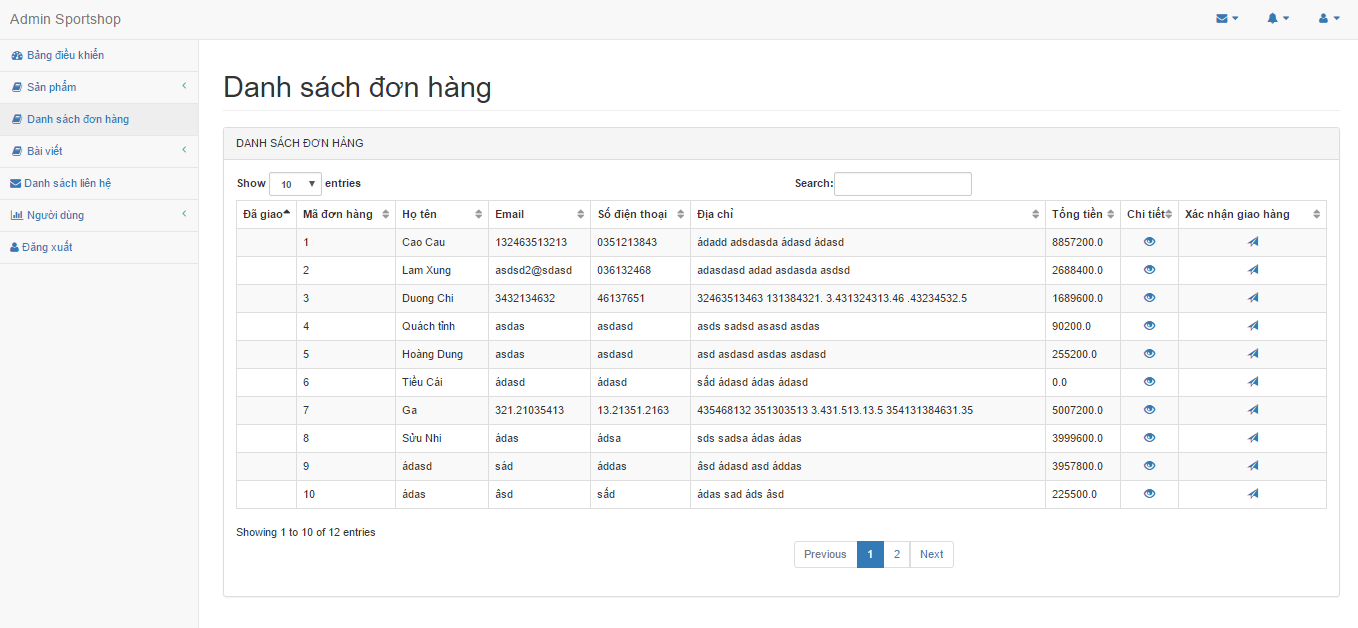
### Danh sách sản phẩm



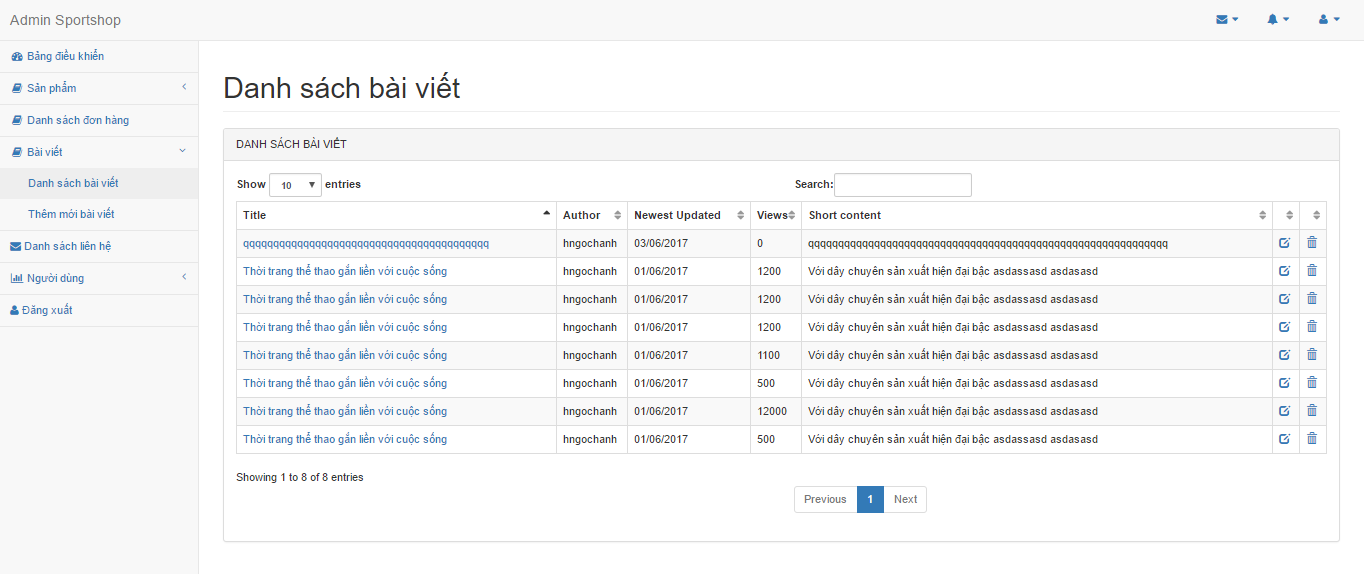
### Thêm mới sản phẩm



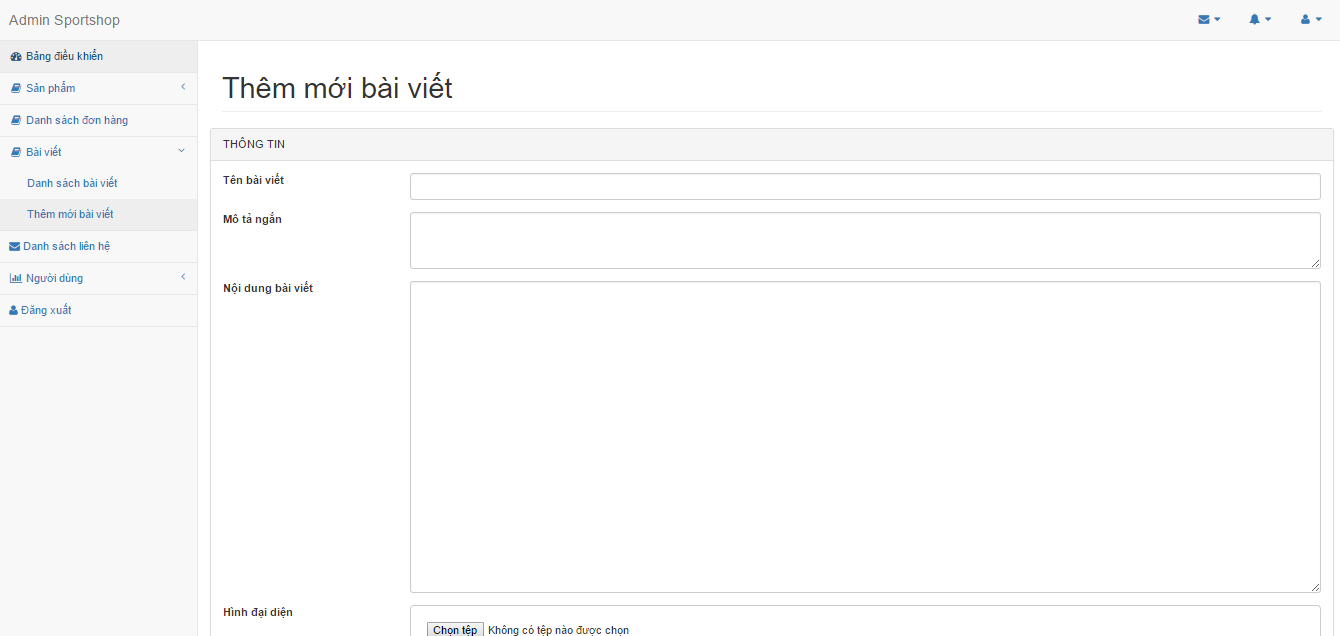
### Danh sách đơn hang



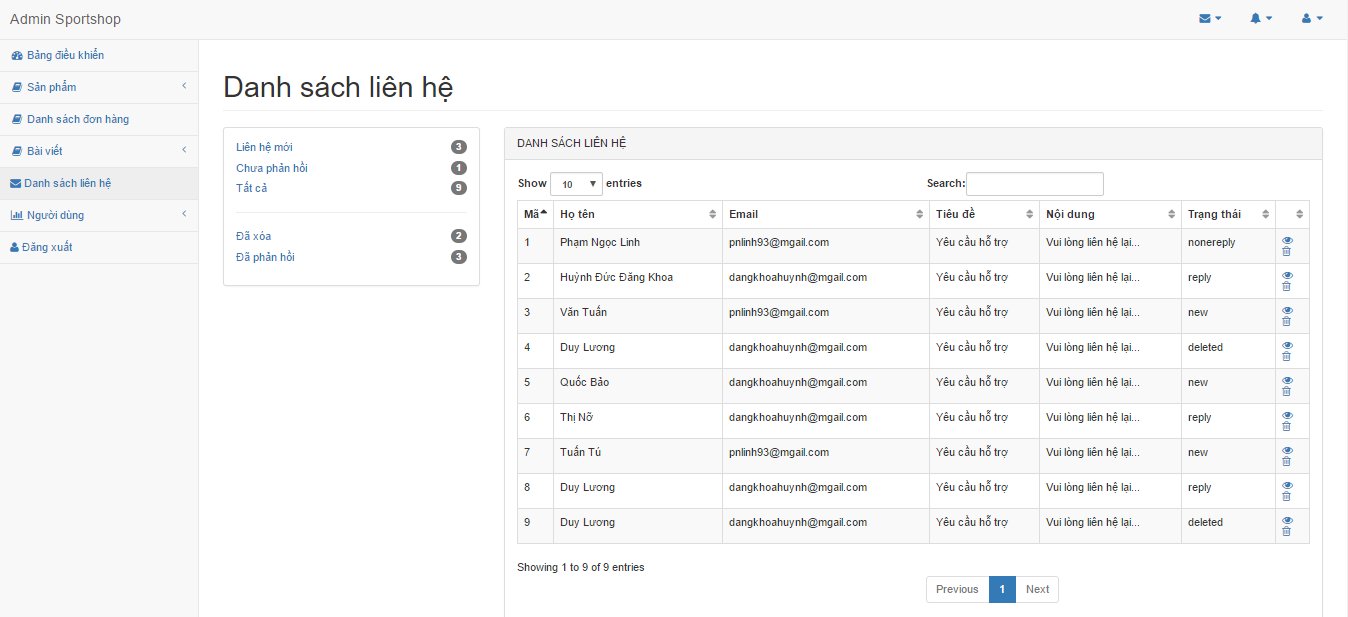
### Danh Sách bài viết



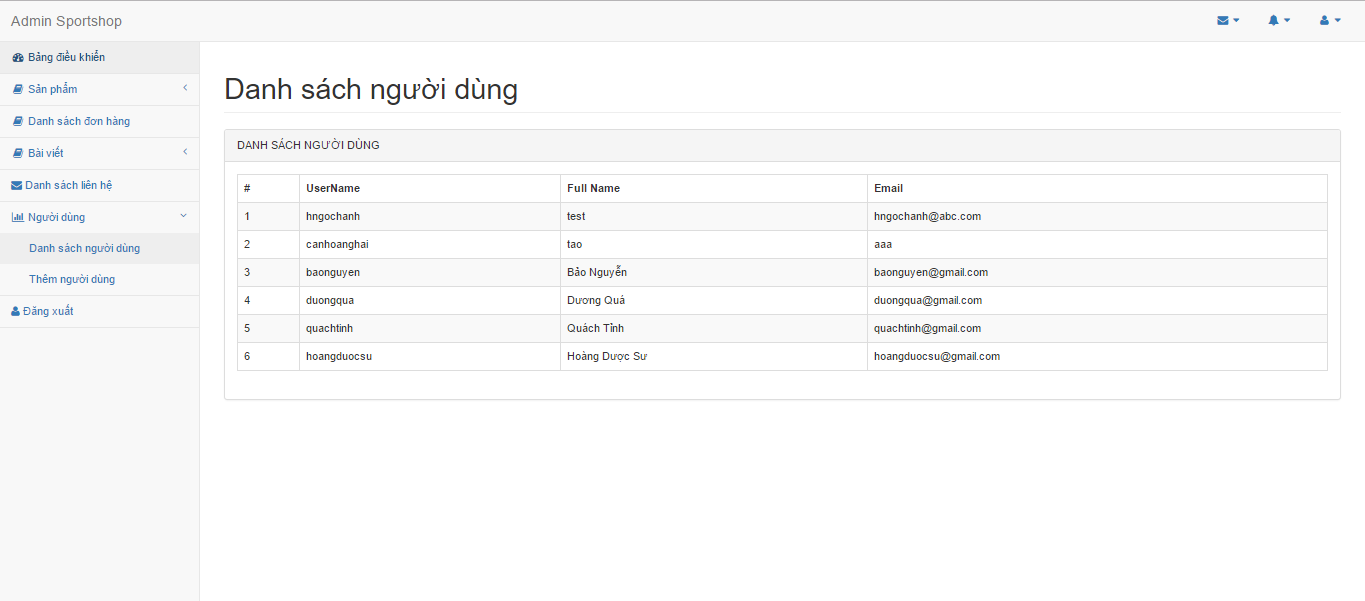
### Thêm mới bài viết



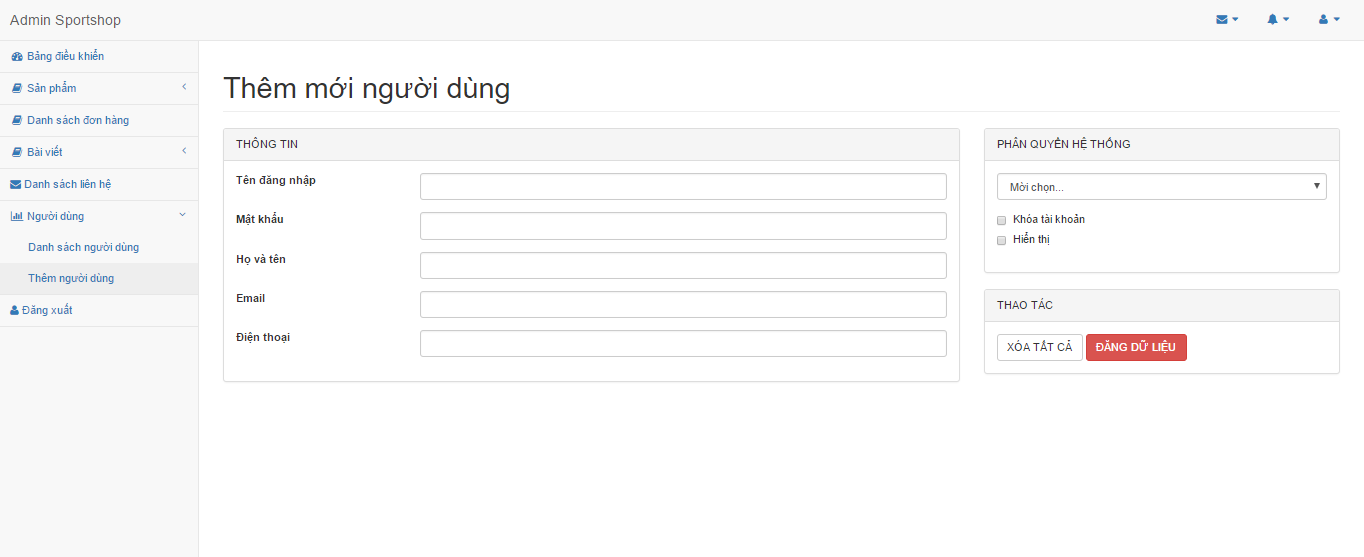
### Danh sách liên hệ



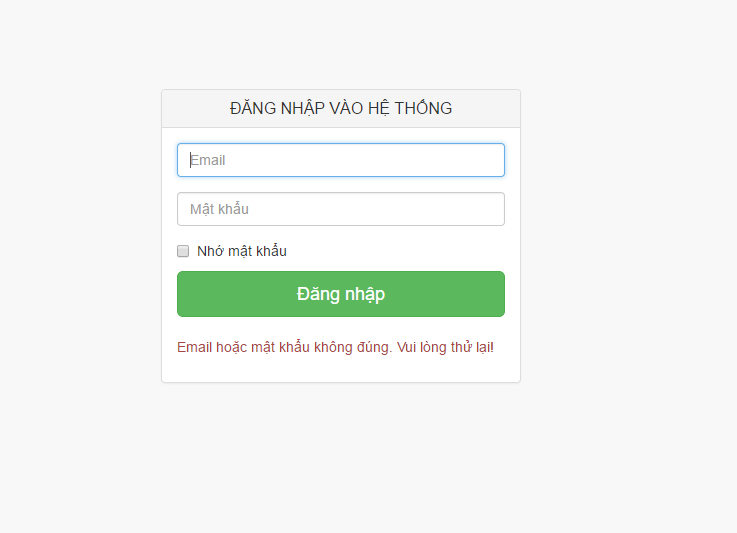
### Người dùng



### Thêm mới người dùng



### Trang login



# PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ Tên** | **Mức độ hoàn thành** | **Công việc** |
| 1 | Hoàng Ngọc Hạnh | 90% | * Site sản phẩm * Giỏ hàng * Trang thanh toán * Thiết kế database * Tin tức |
| 2 | Cấn Hoàng Hải | 90% | * Danh sách đơn hàng * Liên hệ * Chi tiết sản phẩm * Slider bar * Khuyến mãi |
| 3 | Nguyễn Quốc Bảo | 60% | * Đăng nhập * Phân quyền * Quản lý user |