

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**



NGUYỄN THỊ PHƯƠNG THẢO

**PHÁT TRIỂN HỆ THỐNG ĐẶT DỊCH VỤ DU LỊCH
DỰA TRÊN SPRINGMVC**

**ĐỒ ÁN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH**

TP. HỒ CHÍ MINH, 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH



NGUYỄN THỊ PHƯƠNG THẢO

PHÁT TRIỂN HỆ THỐNG ĐẶT DỊCH VỤ DU LỊCH
DỰA TRÊN SPRINGMVC

Mã số sinh viên: 1851010126

ĐỒ ÁN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH

Giảng viên hướng dẫn: ThS Dương Hữu Thành

TP. HỒ CHÍ MINH, 2021

**Ý KIẾN CHO PHÉP BẢO VỆ ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP
CỦA GIẢNG VIÊN HƯỚNG DẪN**

Giảng viên hướng dẫn:

Sinh viên thực hiện: **Lớp:**

Ngày sinh: **Nơi sinh:**

Tên đề tài:

.....

.....

.....

.....

Ý kiến của giảng viên hướng dẫn về việc cho phép sinh viên được bảo vệ đồ án/ khóa luận trước Hội đồng:

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thành phố Hồ Chí Minh, ngày ... tháng ... năm

Người nhận xét

ThS Dương Hữu Thành

LỜI CẢM ƠN

Trong suốt quá trình làm đồ án tốt nghiệp này, em đã nhận được sự chỉ dẫn rất nhiệt tình của các thầy cô, anh chị trong Khoa Công nghệ thông tin trường Đại học Mở Thành phố Hồ Chí Minh qua các buổi sinh hoạt đồ án ngành, hướng dẫn các kỹ thuật tránh lỗi đạo văn khi viết báo cáo đồ án. Qua đó, em đã tiếp thu được những kiến thức hữu ích để làm đồ án ngành theo hướng tốt nhất.

Lời đầu tiên em xin được gửi lời cảm ơn đến Ban giám hiệu Trường Đại học Mở Thành phố Hồ Chí Minh, các thầy cô, anh chị trong Khoa Công nghệ thông tin đã giúp đỡ, truyền đạt các kiến thức để làm tiền đề trong việc thực hiện và hoàn thành Đồ án ngành.

Đặc biệt, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến giảng viên **ThS Dương Hữu Thành** đã trực tiếp chỉ dẫn, hỗ trợ tài liệu, giải đáp các thắc mắc tận tình. Thầy luôn đồng hành, hướng dẫn em trong suốt quá trình chuẩn bị kiến thức, lên ý tưởng, thực hiện và hoàn thành tốt đồ án.

Cuối cùng, em xin gửi lời cảm ơn đến các bạn bè, gia đình, người thân đã hỗ trợ động viên em trong suốt quá trình làm đồ án.

Em xin trân trọng cảm ơn!

Tp.HCM, ngày 8 tháng 8 năm 2021

Sinh viên

Nguyễn Thị Phương Thảo

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT KHÓA LUẬN

Hiện nay, nền kinh tế của nước ta đang bị ảnh hưởng nghiêm trọng bởi hậu quả của COVID-19, trong đó ngành du lịch gần như ngưng trệ để ưu tiên cho các biện pháp chống dịch. Trải qua suốt nhiều tháng ngày chống chọi với bệnh tật, áp lực, căng thẳng và mệt mỏi ắt hẳn sau dịch sẽ có nhiều người chọn du lịch để lấy lại nguồn năng lượng tích cực cho bản thân. Vì vậy, trong thời gian tới nhu cầu tìm hiểu các điểm đến du lịch của người dân sẽ tăng cao.

Trong thời đại công nghệ số thì nhu cầu tìm hiểu các dịch vụ du lịch qua các kênh truyền thông kỹ thuật số sẽ được phát triển mạnh mẽ. Chính vì vậy, em muốn nghiên cứu phát triển một hệ thống đặt dịch vụ du lịch để giới thiệu những điểm đến tuyệt vời, cập nhật xu hướng tin tức du lịch mới nhất cho khách hàng tham khảo. Đưa ra những lịch trình hợp lý nhất với chi phí cực thấp để khách hàng có thể yên tâm tận hưởng kỳ nghỉ của mình một cách trọn vẹn nhất. Giúp khách hàng có thể nêu những nhận định, ý kiến cá nhân, đóng góp cho hệ thống hoàn thiện hơn.

MỤC LỤC

DANH MỤC HÌNH VẼ	8
MỞ ĐẦU	10
Chương 1. GIỚI THIỆU ĐỀ TÀI.....	11
1.1. Giới thiệu đề tài	11
1.2. Lý do chọn đề tài	12
1.3. Mục tiêu nghiên cứu đề tài	13
1.4. Bố cục đề tài	13
Chương 2. TỔNG QUAN SPRING FRAMEWORK	14
2.1. Tổng quan Spring Framework.....	14
2.1.1. Giới thiệu Spring Framework.....	14
2.1.2. Các đặc trưng quan trọng của Spring	14
2.2. Kiến trúc Spring Framework.....	15
2.2.1. Core Container	16
2.2.2. Data Access/Integration	16
2.2.3. Web	17
2.2.4. AOP.....	17
2.2.5. Test	17
2.3. Viết chương trình đầu tiên.....	17
2.4. IoC Container	19
2.4.1. ApplicationContext	20
2.4.2. Dependency Injection.....	21
2.4.3. Sử dụng XML cấu hình beans.....	21
2.4.4. Sử dụng Annotation cấu hình Beans	23
Chương 3. SPRINGMVC	25
3.1. Giới thiệu tổng quan về SpringMVC	25
3.2. Front Controller Design Pattern	26

3.3.	DispatcherServlet	27
3.3.1.	HTTP Request Handling	27
3.3.2.	View Resolver	28
3.4.	Cài đặt môi trường.....	28
3.4.1.	Cài bộ JDK để cài đặt môi trường Java.....	28
3.4.2.	Cài đặt cấu hình môi trường	28
3.4.3.	Cài đặt NetBeans IDE	29
3.4.4.	Cấu hình Apache Tomcat Server	30
3.5.	Viết chương trình đầu tiên với SpringMVC.....	30
3.6.	Controller.....	37
3.6.1.	@PathVariable	37
3.6.2.	@RequestParam	38
3.7.	Tag Libraries	39
3.8.	View Resolver	50
3.8.1.	Redirect	50
3.8.2.	Thuộc tính Flash.....	51
3.8.3.	Static Resources	52
3.8.4.	Multipart Request.....	53
3.9.	Template with tiles	57
3.10.	Spring Security	61
3.11.	Bean Validation	75
3.11.1.	Java Bean Validation.....	75
3.11.2.	Spring Validation	78
3.12.	Rest API.....	82
Chương 4.	HỆ THỐNG ĐẶT DỊCH VỤ DU LỊCH	84
4.1.	Mô tả hệ thống.....	84

4.2.	Cơ sở dữ liệu	84
4.3.	Chức năng đăng ký người dùng	85
4.4.	Chức năng đăng nhập	86
4.5.	Chức năng quản lý khách hàng	87
4.6.	Chức năng quản lý nhân viên	89
4.7.	Quản lý tin tức du lịch.....	89
4.8.	Chức năng quản lý thuê xe	92
4.9.	Quản lý thống kê	96
Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....		97
5.1.	Kết luận	97
5.2.	Hướng phát triển.....	97
TÀI LIỆU THAM KHẢO.....		98

DANH MỤC HÌNH VẼ

Hình 1. Kiến trúc Spring Framework (nguồn: https://www.tutorialspoint.com/spring).....	15
Hình 2. Ví dụ về cách thức hoạt động của IoC Container	19
Hình 3. Mô hình MVC trong ứng dụng Web	25
Hình 4. Minh họa cách thức hoạt động của Front Controller	26
Hình 5. Minh họa cách thức hoạt động của Dispatcher.....	27
Hình 6. Tạo mới biến môi trường cho Java	28
Hình 7. Cài đặt biến môi trường cho Java	29
Hình 8. Tạo mới project Java Maven phát triển Web Application.....	30
Hình 9. Thông tin project Java Maven phát triển Web Application.....	31
Hình 10. Thêm Server Apache Tomcat vào project ứng dụng Web	31
Hình 11. Chạy thử ứng dụng với Postman	83
Hình 12. Lược đồ cơ sở dữ liệu quan hệ của hệ thống đặt dịch vụ du lịch	85
Hình 13. Cấu trúc project QuanLyTourDuLich	85
Hình 14. Chức năng đăng ký người dùng trong hệ thống	86
Hình 15. Chức năng đăng nhập của hệ thống tour du lịch	87
Hình 16. Giao diện quản lý của người dùng.....	87
Hình 17. Chức năng quản lý khách hàng của hệ thống	88
Hình 18. Chức năng cập nhật người dùng của hệ thống.....	88
Hình 19. Chức năng xóa người dùng.....	89
Hình 20. Chức năng quản lý nhân viên của hệ thống.....	89
Hình 21. Giao diện xem tin tức của hệ thống	90
Hình 22. Giao diện xem chi tiết tin tức của hệ thống	90
Hình 23. Chức năng bình luận bài viết tin tức.....	91
Hình 24. Chức năng quản lý tin tức của hệ thống	91
Hình 25. Chức năng sửa tin tức trong hệ thống.....	92
Hình 26. Giao diện hiển thị các loại xe cho thuê.....	92
Hình 27. Giao diện hiển thị danh sách xe cho thuê	93
Hình 28. Giao diện mô tả chi tiết xe cho thuê	93
Hình 29. Chức năng đặt xe trong hệ thống.....	94

Hình 30. Rest API gọi hàm ajax hiển thị thông tin đặt xe	94
Hình 31. Chức năng thanh toán với Paypal	95
Hình 32. Giao diện thông báo đặt xe thành công	95
Hình 33. Chức năng quản lý xe trong hệ thống	96
Hình 34. Chức năng quản lý thống kê của hệ thống	96

MỞ ĐẦU

Trong cuộc Cách mạng công nghiệp 4.0 ở thời điểm hiện nay, việc áp dụng các công nghệ mới, tăng cường đổi mới sáng tạo là một trong những điều thiết yếu để tồn tại và phát triển trong xã hội. Ngày nay, mọi người đều có xu hướng tìm kiếm và trao đổi thông tin với nhau thông qua Internet. Các trang web có giao diện đẹp mắt, thông tin hữu ích, dễ tiếp cận với người dùng, từ khóa thân thiện đều là những yếu tố được người dùng ưu tiên truy cập và tương tác nhiều. Một trong những ưu điểm nổi bật của công nghệ số là giúp con người tiết kiệm chi phí và thời gian. Bạn sẽ không cần đi một đoạn đường xa để mua một món đồ mình cần mà chỉ cần một cái click chuột, món đồ sẽ được chuyển đến bạn trong thời gian ngắn. Bên cạnh đó, bạn còn có thể thanh toán thông qua các cổng thanh toán trực tuyến giúp giảm thiểu tối đa rủi ro như khi sử dụng tiền mặt. Hơn thế nữa, bạn có thể thoải mái bày tỏ quan điểm, ý kiến cá nhân của mình trên các diễn đàn công nghệ, lắng nghe, tích lũy được các thông tin hữu ích của người khác chia sẻ.

Qua đó, chúng ta thấy được rằng trải nghiệm của khách hàng là điều quan trọng nhất trong việc xây dựng, thiết kế hệ thống của lập trình viên. Chính vì vậy, lập trình viên, các nhà phát triển lập trình Web hiện nay luôn phải cập nhật những xu hướng mới, công nghệ kỹ thuật mới để mang đến cho khách hàng trải nghiệm tuyệt vời nhất. Để có thể xây dựng, thiết kế và cải tiến những tính năng mới của một website đòi hỏi lập trình viên phải có lượng kiến thức nền tảng tốt, luôn nỗ lực học hỏi và tư duy phân tích xử lý vấn đề logic và hiệu quả. Hiện nay, có rất nhiều ngôn ngữ lập trình web phổ biến như Python, Java, C++, JavaScript, PHP, Ruby,...

Chương 1. GIỚI THIỆU ĐỀ TÀI

Chương 1 trình bày thực trạng ngành Công nghệ thông tin hiện nay cũng như sự phổ biến của các ứng dụng lập trình Web. Bên cạnh đó chương này còn giới thiệu tổng quan về đề án, lý do chọn đề tài đề án, mục đích trong việc nghiên cứu đề án và bố cục của đề tài.

1.1. Giới thiệu đề tài

Với sự phát triển chóng mặt của công nghệ kỹ thuật hiện đại 4.0, ngành Công nghệ thông tin đang là ngành ‘hot’ được các doanh nghiệp săn đón cũng như là ngành được chú trọng trong chương trình đào tạo của các trường Đại học. Nó được xem là ngành đào tạo mũi nhọn để nâng cao năng lực, phát triển hệ thống khoa học của quốc gia cũng như đẩy mạnh phát triển đất nước. Có thể thấy Internet đặc biệt quan trọng đối với con người trong cuộc sống hiện nay. Nó đóng vai trò hết sức quan trọng trong cuộc sống thường ngày của chúng ta. Bạn sẽ không phải xếp hàng để chờ mua vé xem phim, hay phải đi những quãng đường xa để mua được món đồ chúng ta cần. Bạn không cần bỏ tiền để mua báo giấy và tiếp thu thông tin một cách chậm chạp. Thay vào đó, chúng ta có thể lướt web để đọc và cập nhật những tin tức mới một cách nhanh nhất...Chính vì vậy, phát triển ứng dụng Web đang là hướng đi được nhiều lập trình viên chọn lựa.

Có nhiều ngôn ngữ ra đời để hỗ trợ việc phát triển lập trình Web hữu ích như: JavaScript, Java, Python, PHP, C#, Ruby, Go, HTML, SQL, CSS,...Một ngôn ngữ lập trình được phụ thuộc vào nhiều yếu tố như tốc độ, tiện ích, nhiều tài nguyên, dễ học và sử dụng...Điển hình là Java – ngôn ngữ lập trình web mạnh mẽ hàng đầu ở thời điểm này. Java là ngôn ngữ phổ biến và được các lập trình viên ưa chuộng trong việc phát triển các ứng dụng web hiện nay. Java được biết đến với ngôn ngữ có tính hướng đối tượng đầy đủ nhất và đặc biệt nó có thể chạy trên nhiều nền tảng khác nhau. Khi nhắc đến Java, các Java Framework là đề tài mà các lập trình viên không thể nào không nhắc đến trong đó phải kể đến Framework SpringMVC.

SpringMVC là một trong những Java Framework có lịch sử lâu đời và tốt nhất cho đến thời điểm hiện tại. SpringMVC có tài liệu tuyệt vời và cộng đồng hỗ trợ đông đảo trên toàn thế giới. Nó giúp bạn tổ chức code một cách sạch sẽ và truy cập nhanh

hơn. Bên cạnh đó nó còn cung cấp các chức năng bảo mật đi kèm giúp lập trình viên có thể ứng dụng vào bất kỳ dự án hoặc nhiệm vụ nào. Đồ án ‘Phát triển hệ thống đặt dịch vụ du lịch dựa trên SpringMVC’ sẽ đề cập đến cách tổ chức cấu hình trong SpringMVC, thực hiện các chức năng quản lý, bảo mật, phân quyền cho người dùng. Hy vọng nó sẽ là tài liệu dễ tiếp cận, hữu ích cho các bạn bước đầu làm quen với Framework SpringMVC.

Hệ thống này bao gồm:

- Quản lý, phân quyền người dùng.
- Quản lý các tin tức du lịch.
- Cho phép bình luận các tin tức du lịch.
- Quản lý cho thuê xe du lịch và thanh toán trực tuyến.
- Thống kê số lượng doanh thu.

1.2. Lý do chọn đề tài

Với sự ảnh hưởng nặng nề của đại dịch COVID-19, mọi người đang phải chịu sự áp lực, căng thẳng và mệt mỏi sau thời gian dài cách ly. Sau một thời gian dài không được đi ra ngoài, nhiều người có nhu cầu đi du lịch để phục hồi sức khỏe tâm lý, nghỉ ngơi thư giãn lấy lại năng lượng tích cực. Đây mạnh phát triển du lịch là một trong những mục tiêu mà nhà nước hướng đến để phục hồi nền kinh tế. Ngày nay, mọi người có xu hướng tra cứu, tìm kiếm thông tin trên Internet. Nhu cầu khám phá các địa điểm du lịch theo lịch trình hợp lý, tiết kiệm chi phí, bảo đảm an toàn, tránh những rủi ro trong chuyến đi. Nắm bắt được tình hình trên, các công ty du lịch ra đời để mang đến cho khách hàng những chuyến đi, trải nghiệm tuyệt vời nhất.

Từ những vấn đề trên, em muốn nghiên cứu và phát triển một hệ thống đặt dịch vụ du lịch nhằm đáp ứng các yêu cầu của công ty cũng như phù hợp với thị hiếu của khách hàng. Qua quá trình học tập và nghiên cứu, em thấy rất hứng thú với Framework SpringMVC. Các ứng dụng làm bằng Framework này có thể tổ chức code một cách rõ ràng để dễ dàng quản lý, truy xuất dữ liệu khi cần. Ngoài ra, khi có lỗi hay cần trợ giúp thì luôn có một cộng đồng hỗ trợ rất hùng mạnh. Hy vọng sẽ đưa ra được những giải

pháp tối ưu nhằm hoàn thiện hệ thống để đem đến cho khách hàng những trải nghiệm tốt nhất.

1.3. Mục tiêu nghiên cứu đề tài

Có thể cung cấp tài liệu hữu ích cho các bạn có hứng thú và muốn tìm hiểu về SpringMVC, nắm bắt được các lý thuyết cơ bản, cách cấu hình và tạo một ứng dụng đơn giản. Bên cạnh đó, em muốn nghiên cứu, xây dựng một hệ thống đặt dịch vụ du lịch với những chức năng đảm bảo các nhu cầu cần thiết của khách hàng. Thiết kế các chức năng của hệ thống để dễ dàng quản lý.

1.4. Bố cục đề tài

Cấu trúc của đề tài này gồm có 4 chương:

Chương 1: Giới thiệu đề tài.

Chương 2: Tổng quan Spring Framework.

Chương 3: Spring MVC

Chương 4: Hệ thống đặt dịch vụ du lịch.

Chương 5: Kết luận và hướng phát triển.

Chương 2. TỔNG QUAN SPRING FRAMEWORK

Chương này giới thiệu tổng quan về Spring Framework, qua đây người đọc có thể nắm được khái niệm, cấu trúc, hiểu được các định nghĩa, các đặc trưng quan trọng và có thể viết được chương trình đầu tiên với Spring Framework.

2.1. Tổng quan Spring Framework

2.1.1. Giới thiệu Spring Framework

Framework là các kỹ thuật, các cấu hình, cấu trúc,...mà đã có đơn vị khác làm cho chúng ta rồi và chúng ta chỉ cần học cách sử dụng nó, và khi sử dụng nếu bạn thấy có khiếm khuyết chỗ nào thì bạn có thể trao đổi hay gửi yêu cầu góp ý đến những đơn vị đó để người ta cải thiện thêm.

Spring framework là một framework mã nguồn mở hỗ trợ các lập trình viên Java phát triển Java EE một cách dễ dàng, nhanh chóng. Nó cung cấp cho lập trình viên những thư viện, công cụ có sẵn giúp cho việc lập trình không mất nhiều thời gian để xây dựng mọi thứ từ ban đầu mà chỉ việc tập trung vào các công cụ có sẵn.

Vào đầu những năm 2000, chúng ta có một nền tảng để phát triển lập trình web với Java đó là JEE(J2EE hoặc Jakarta EE). Nhưng sau một khoảng thời gian dài, có thể do thiếu nhân lực mà nó rất hạn chế được cập nhật phiên bản mới vì vậy nó dần mờ nhạt và không còn được quan tâm nhiều. Vào 03/2004 một nhóm lập trình viên bao gồm: Rod Johnson(sau này được lên làm CEO), Juergen Hoeller(là người cuối cùng đến nay vẫn đang phát triển Spring Framework), Keith Donald và Colin SampaLeanu đã cùng nhau tạo ra một nền tảng khác để thay thế, khắc phục những nhược điểm của JEE ở thời điểm đó, nó là Spring Framework.

2.1.2. Các đặc trưng quan trọng của Spring

Inversion of Control (IoC): IoC là một kỹ thuật lập trình để tách các thành phần và lớp trong hệ thống được thực hiện thông qua việc đưa các đối tượng phụ thuộc vào một khung khi xây dựng chương trình.

Lightweight: Spring là một framework nhẹ về độ trong suốt và kích thước. Kích thước của khung cơ bản chỉ khoảng 1MB giúp chương trình bắt đầu nhanh hơn và giảm bớt độ phức tạp của mã nguồn.

Aspect-Oriented Programming (AOP): AOP là một mô hình lập trình trong đó các chức năng được tách biệt khỏi logic xử lý nghiệp vụ của chương trình chính giúp cho việc bảo trì nó trở nên dễ dàng hơn.

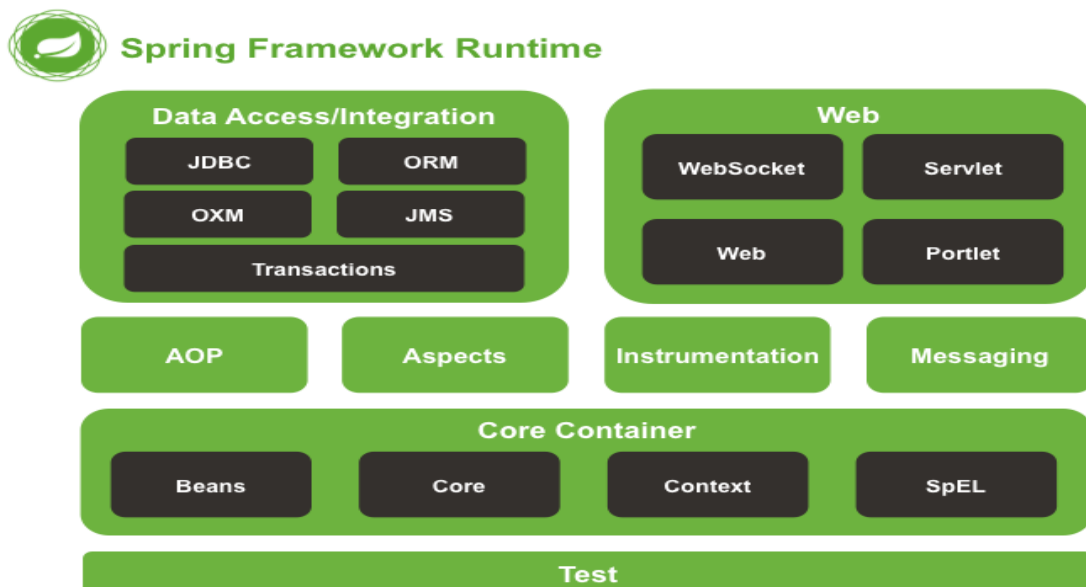
Spring MVC Framework: cung cấp kiến trúc theo mô hình Model-View-Controller (MVC) để phát triển các ứng dụng Web một cách linh hoạt và mạnh mẽ. Nó cô lập các khía cạnh khác nhau của ứng dụng như xử lý đầu vào, nghiệp vụ, giao diện...

JDBC exception handling: Tầng trừu tượng JDBC của Spring cung cấp một hệ thống xử lý ngoại lệ, điều này làm đơn giản hóa mã nguồn cần viết.

Spring Security: cung cấp cơ chế xác thực và phân quyền cho các ứng dụng. Nó có thể dễ dàng mở rộng để đáp ứng các yêu cầu.

2.2. Kiến trúc Spring Framework

Trong phần này trình bày về chi tiết của các module trong Spring Framework. Spring Framework cung cấp khoảng 20 module, lập trình viên dựa vào yêu cầu của ứng dụng để sử dụng chúng một cách hợp lý. Hình 1. dưới đây là kiến trúc tổng thể của Spring Framework.



Hình 1. Kiến trúc Spring Framework (nguồn:
<https://www.tutorialspoint.com/spring>)

2.2.1. Core Container

Bao gồm các module Core, Beans, Context, SpEL chi tiết như sau:

- Core module: đây là thành phần cơ bản quan trọng nhất của Spring Framework, cung cấp các đặc điểm như Inversion of Control (IoC) và Dependency Injection (DI).
- Bean module: cung cấp BeanFactory, là mẫu thiết kế Factory tổng quát cho phép bạn phân tách sự phụ thuộc cấu hình, đặc điểm kỹ thuật khỏi logic chương trình.
- Context module: được xây dựng dựa trên nền tảng cung cấp bởi các module Core và Beans. Nó là phương tiện giúp bạn truy cập bất kỳ đối tượng nào đã được xác định và cấu hình. Application Context sẽ móc nối (wire) đối tượng khi bạn gọi đến Spring container.
- SpEL (Spring Expression Language): cung cấp một ngôn ngữ biểu diễn mạnh mẽ hỗ trợ truy vấn và thao tác với một đối tượng trong thời gian thực thi chương trình. Nó hỗ trợ thiết lập các thuộc tính, gọi phương thức, gán thuộc tính, truy cập các phương thức và truy xuất các đối tượng theo tên từ IoC Container của Spring.

2.2.2. Data Access/Integration

Bao gồm các module JDBC, ORM, OXM, JMS và Transaction có chi tiết như sau:

- JDBC module: cung cấp một lớp trừu tượng JDBC giảm bớt một số mã nguồn JDBC nhàm chán.
- ORM (Object-relational mapping) module: cung cấp các API ORM gồm: JPA, JDO, Hibernate và iBatis.
- OXM (Object XML Mapping) module: cung cấp một lớp trừu tượng hỗ trợ ánh xạ Object/XML cho JAXB, Castor, XMLBeans, JiBX và XStream.
- JMS (Java Messaging Service) module: chứa các tính năng để tạo và gửi các thông điệp (message).
- Transaction module: hỗ trợ khai báo và quản lý chương trình cho các lớp hiện thực các interface và các lớp POJO.

2.2.3. Web

Tầng Web bao gồm các module Web, Web-Socket, Servlet, Portlet, Struts có chi tiết như sau:

- Web module: cung cấp các đặc trưng cơ bản như tải tệp lên, khởi tạo IoC Container và Application Context.
- Web-Socket module: cung cấp hỗ trợ cho giao tiếp hai chiều giữa máy khách và máy chủ trong các ứng dụng web.
- Servlet module: chứa hiện thực mô hình MVC cho các ứng dụng web.
- Web-Portlet module: cung cấp việc hiện thực mô hình MVC được sử dụng trong môi trường Portlet.
- Struts module: chứa các lớp hỗ trợ để tích hợp tầng Struts Web trong ứng dụng Spring.

2.2.4. AOP

AOP module: cung cấp hiện thực AOP, giúp chúng ta thêm một chức năng mới vào mã nguồn mà không cần thay đổi thiết kế sẵn có.

2.2.5. Test

Test module: hỗ trợ kiểm thử các ứng dụng của Spring bằng Junit và TestNG. Nó cung cấp các mock object để bạn có thể sử dụng và kiểm thử mã nguồn một cách riêng biệt.

2.3. Viết chương trình đầu tiên

Tạo một project Java Application, trong tập tin pom.xml bổ sung các dependencies sau:

```
<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-core</artifactId>

    <version>5.3.9</version>

</dependency>
```

```
<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-context</artifactId>

    <version>5.3.9</version>

</dependency>
```

Tạo tập tin HelloWorld.java trong gói com.ntpt.demo

```
public class HelloWorld {

    private String message;

    //Các phương thức getter/setter của message

}
```

Tạo tập tin Beans.xml trong src/main/resources có nội dung như sau:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="helloWorld" class=" com.ntpt.demo.HelloWorld">

        <property name="message" value="Hello World!"/>

    </bean>

</beans>
```

Tạo lớp Tester.java trong gói com.ntpt.demo và tạo phương thức main trong lớp.

```
public class Tester {

    public static void main(String[] args) {

        ApplicationContext context =

            new ClassPathXmlApplicationContext("Beans.xml");

    }

}
```

```

        HelloWorld h = (HelloWorld) context.getBean("helloWorld");

        System.out.println(h.getMessage());

    }

}

```

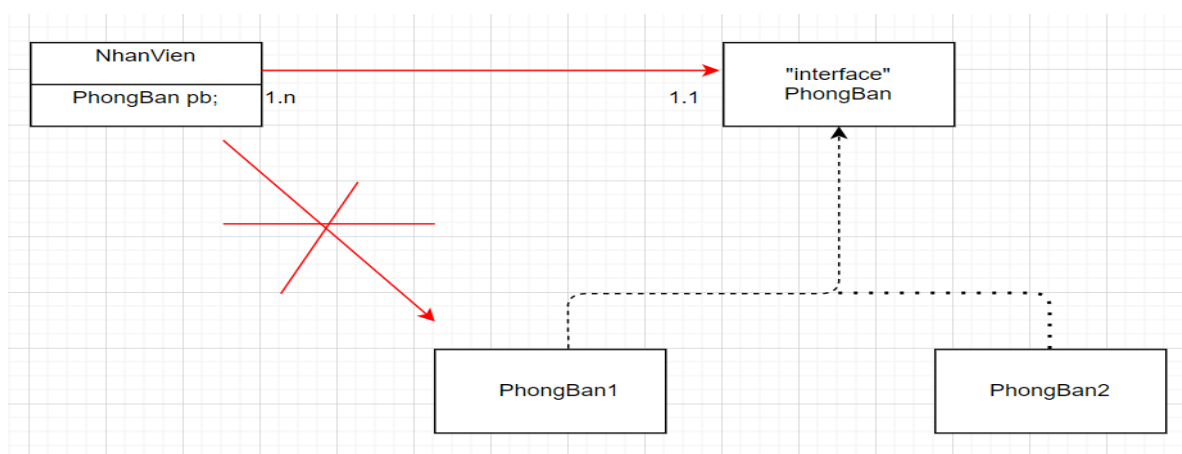
Thực thi chương trình, ta có kết quả:

Hello World!

2.4. IoC Container

Spring Container là vùng chứa quan trọng nhất của Spring Framework. Nó sẽ tạo ra các đối tượng, kết nối chúng lại với nhau, cấu hình và quản lý vòng đời của chúng từ khi tạo ra đến khi kết thúc. IoC là một kỹ thuật lập trình để tách các thành phần và lớp trong hệ thống được thực hiện thông qua việc đưa các đối tượng phụ thuộc vào một khung khi xây dựng chương trình. IoC Container sử dụng Dependency Injection (DI) (xem mục 2.4.2) để quản lý các thành phần xây dựng nên hệ thống, ứng dụng.

Xét ví dụ có hai lớp NhanVien và PhongBan, lớp NhanVien có thuộc tính kiểu lớp PhongBan để biết được NhanVien đó thuộc PhongBan nào, như vậy lớp NhanVien sẽ phụ thuộc vào lớp PhongBan. Vậy làm sao để giữ được tính độc lập giữa hai lớp. Dependency Injection (DI) sẽ giúp chúng ta thực hiện vấn đề này. Các lớp sẽ giao tiếp với nhau thông qua Interface chứ không thông qua Implementation. Sau đó, IoC Container sẽ có trách nhiệm tạo một đối tượng PhongBan và truyền tham chiếu của nó vào lớp SinhVien.



Hình 2. Ví dụ về cách thức hoạt động của IoC Container

IoC Container sử dụng DI (Dependency Injection) để liên kết các thành phần lại với nhau tạo nên một ứng dụng, gọi là các đối tượng Beans. Các tập tin XML sẽ cung cấp thông tin cho IoC Container. Có hai loại IoC Container:

Spring BeanFactory Container: là container đơn giản nhất và cũng là container cơ sở cung cấp hỗ trợ cơ bản cho DI. Tất cả các container khác đều phải hiện thực BeanFactory. Để sử dụng BeanFactory, chúng ta cần phải tạo mới một đối tượng XMLBeanFactory như dưới đây:

```
Resource resource = new ClassPathResource("applicationContext.xml");  
BeanFactory factory = new XMLBeanFactory(resource);
```

Spring ApplicationContext Container: là một giao diện con của BeanFactory. Để sử dụng ApplicationContext chúng ta cần tạo mới một đối tượng ClassPathXmlApplicationContext như dưới đây:

```
ApplicationContext context =  
    new ClassPathXmlApplicationContext(applicationContext.xml);
```

2.4.1. ApplicationContext

XML (eXtensible Markup Language): là một ngôn ngữ đánh dấu giống như HTML, được thiết kế để lưu trữ và vận chuyển dữ liệu. Bạn có thể xác định tất cả các bean và các phụ thuộc bắc cầu của chúng trong một tệp xml duy nhất.

ApplicationContext là một giao diện để cung cấp thông tin cấu hình cho một ứng dụng. Có nhiều lớp được cung cấp bởi Spring Framework để hiện thực ApplicationContext và đưa ra thông tin cấu hình trong các ứng dụng. Một số lớp hiện thực ApplicationContext như:

- ClassPathXmlApplicationContext: tải một cấu hình XML từ một classpath và quản lý các bean của nó.
- FileSystemXmlApplicationContext: là một lớp đọc cấu hình XML từ đường dẫn tuyệt đối.

- `XmlWebApplicationContext`: được sử dụng để tạo context cho ứng dụng Web bằng cách nạp tập tin cấu hình XML từ vị trí chuẩn trong thư mục webapp, mặc định tập tin nằm ở `/WEB-INF/applicationContext.xml`.

- `AnnotationConfigWebApplicationContext`: được sử dụng để tạo web application context bằng cách gắn annotation `@Configuration` vào các lớp Java.

2.4.2. Dependency Injection

Dependency Injection là một kỹ thuật trong đó một đối tượng nhận các đối tượng khác mà nó phụ thuộc vào. Thay vì bạn phải chỉ định yêu cầu thì DI sẽ cho bạn biết cái gì cần cho bạn. Điều này sẽ giúp tách biệt về việc xây dựng và sử dụng các đối tượng. Lập trình viên có thể tăng khả năng đọc và tái sử dụng mã nguồn.

Có ít nhất hai cách sử dụng để xử lý sự phụ thuộc giữa các đối tượng:

- `Constructor injection`: các thành phần phụ thuộc được cung cấp thông qua các phương thức khởi tạo với tham số.

- `Setter injection`: sử dụng các phương thức setter trên bean để truyền (injected) vào các thành phần phụ thuộc.

2.4.3. Sử dụng XML cấu hình beans

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd>
    <bean id="helloWorld" class="com.ntpt.demo.HelloWorld">
        <property name="message" value="Hello World!"/>
    </bean>
</beans>
```

2.4.3.1. Một số thuộc tính quan trọng trong thẻ bean:

`class`: là thuộc tính bắt buộc dùng để chỉ định tên lớp sử dụng tạo ra bean.

name: dùng để chỉ định tên duy nhất giúp nhận dạng bean, ngoài ra bạn có thể sử dụng id thay cho name với ý nghĩa tương tự trong sử dụng cấu hình dựa trên XML.

scope: xác định phạm vi truy cập của đối tượng bean, mặc định thuộc tính này có giá trị là singleton. Các giá trị của scope là:

- singleton: tạo một thể hiện beans duy nhất cho tất cả các lần sử dụng nó.
- prototype: tạo các thể hiện beans mới mỗi khi sử dụng nó.
- request: tạo thể hiện beans mới trong mỗi HTTP request, nó chỉ có hiệu lực trong Spring Web ApplicationContext.
- session: tạo thể hiện beans mới trong mỗi HTTP session, nó chỉ có hiệu lực trong Spring Web ApplicationContext.

init-method: chỉ định phương thức được gọi khi khởi tạo beans.

destroy-method: chỉ định phương thức được gọi trước khi beans bị hủy.

2.4.3.2. Thẻ <property>

Sử dụng các phương thức setter để inject. Có một số thuộc tính sau:

name: chỉ định tên thuộc tính Java bean.

value: thiết lập giá trị cho thuộc tính Java bean.

ref: dùng để tham chiếu tới các đối tượng bean khác.

2.4.3.3. Thẻ <constructor-arg>

Sử dụng các phương thức khởi tạo để inject. Thẻ này có các thuộc tính quan trọng sau:

index: dùng để chỉ định chỉ số trong danh sách đối số của phương thức khởi tạo.

type: dùng để chỉ định kiểu của đối số trong constructor.

value: giá trị là chuỗi chỉ định giá trị cho bean.

ref: dùng để tham chiếu tới các đối tượng bean khác.

2.4.3.4. Autowiring

Autowiring là một đặc trưng nổi bật trong Spring Framework, nó cho phép ta không cần cung cấp bean injection tường minh nhưng vẫn có thể dự đoán một cách thông minh đối tượng nào đang được tham chiếu tới, giúp giảm thiểu cấu hình Spring Bean. Nó được thể hiện thông qua thuộc tính autowire của thẻ <beans>, mặc định nó bị vô hiệu hóa (disable).

2.4.3.5. Cấu hình cho Java Collection

Spring hỗ trợ cung cấp 4 loại cấu hình Collections:

<list>: dùng để kết nối với các thuộc tính kiểu danh sách (list).

<set>: dùng để kết nối với các thuộc tính kiểu tập hợp (set), các phần tử trong danh sách không được phép trùng nhau.

<map>: dùng để kết nối với các thuộc tính kiểu Collection trong đó mỗi phần tử là một cặp key/value, và key/value có thể có kiểu dữ liệu bất kỳ.

<props>: tương tự như <map> nhưng cặp key/value có kiểu dữ liệu String.

2.4.4. Sử dụng Annotation cấu hình Beans

Từ phiên bản Spring 2.5 trở đi, cho phép sử dụng annotation để cấu hình trực tiếp trong các lớp thay vì sử dụng tập tin XML.

Để có thể sử dụng annotation ta cần bật nó lên trong tập tin cấu hình Beans.xml như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context = "http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
```

```
<context:annotation-config />
```

```
</beans>
```

Một số annotation quan trọng:

@Required: áp dụng vào các phương thức setter của các thuộc tính.

@Autowired: áp dụng cho các thuộc tính, phương thức khởi tạo, phương thức setter.

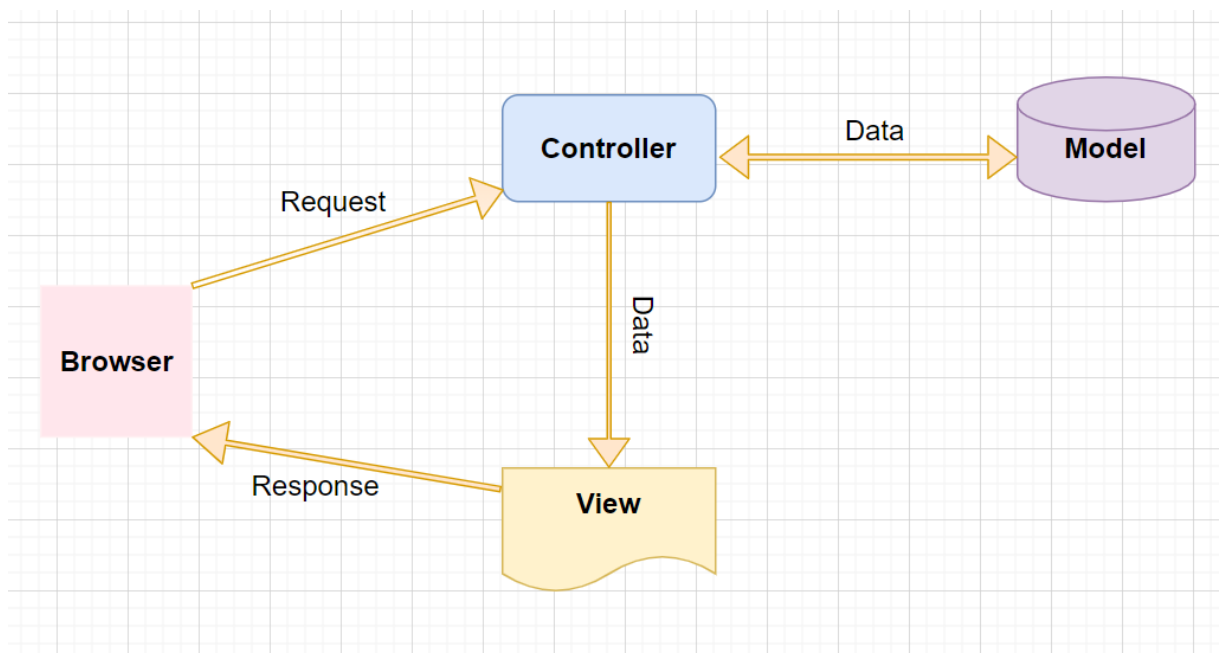
@Qualifier: khi có nhiều đối tượng beans cùng kiểu, thì sử dụng annotation này kết hợp với **@Autowired** để chỉ định đến cụ thể đối tượng beans nào.

Chương 3. **SPRINGMVC**

Sau khi đọc xong chương này, người đọc có thể hiểu được tổng quan về SpringMVC, biết được cơ chế hoạt động của một ứng dụng Web phát triển bằng SpringMVC. Ngoài ra, người đọc còn biết cách cài môi trường, tạo project và xây dựng được một project đơn giản.

3.1. Giới thiệu tổng quan về SpringMVC

SpringMVC là một Java Framework mã nguồn mở được dùng để sử dụng và xây dựng các ứng dụng web. Nó tuân thủ theo mô hình Model-View-Controller. Nó hiện thực được tất cả các tính năng cơ bản của Spring Framework như Inversion of Control (IoC) và Dependency Injection (DI). SpringMVC cung cấp một giải pháp để sử dụng mô hình MVC trong Spring Framework với sự trợ giúp của DispatcherServlet. DispatcherServlet đóng vai trò là một lớp nhận các yêu cầu chuyển đến và ánh xạ nó đến đúng tài nguyên như controllers, models và views.



Hình 3. Mô hình MVC trong ứng dụng Web

Các ưu điểm của SpringMVC:

- SpringMVC tách biệt các phần độc lập, không ảnh hưởng đến nhau.
- Nó sử dụng Servlet container có trọng lượng nhẹ để phát triển và triển khai ứng dụng.

- Nó cung cấp một cấu hình mạnh mẽ cho cả Framework và các lớp ứng dụng.
- Có thể tái sử dụng mã nguồn, thay vì tạo các đối tượng mới nó cho phép chúng ta sử dụng các đối tượng nghiệp vụ hiện có.
- Việc kiểm tra dữ liệu được thực hiện một cách dễ dàng.
- Nó cung cấp các annotation cụ thể để dễ dàng chuyển hướng trang.

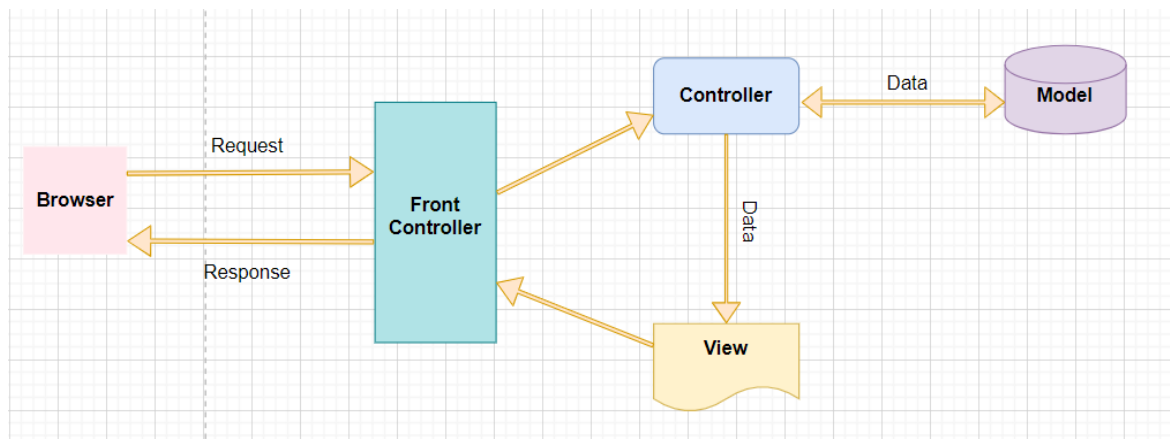
3.2. Front Controller Design Pattern

Front Controller Design Pattern cung cấp cơ chế xử lý tập trung các yêu cầu để tất cả các yêu cầu sẽ được xử lý bởi một trình xử lý duy nhất. Sau đây là các thực thể của design pattern này:

Front Controller: là trình xử lý duy nhất cho tất cả các loại yêu cầu gửi đến từ ứng dụng.

Dispatcher: Front Controller có thể sử dụng một đối tượng dispatcher để gửi yêu cầu đến trình xử lý cụ thể tương ứng.

View: là đối tượng mà các yêu cầu được thực hiện.



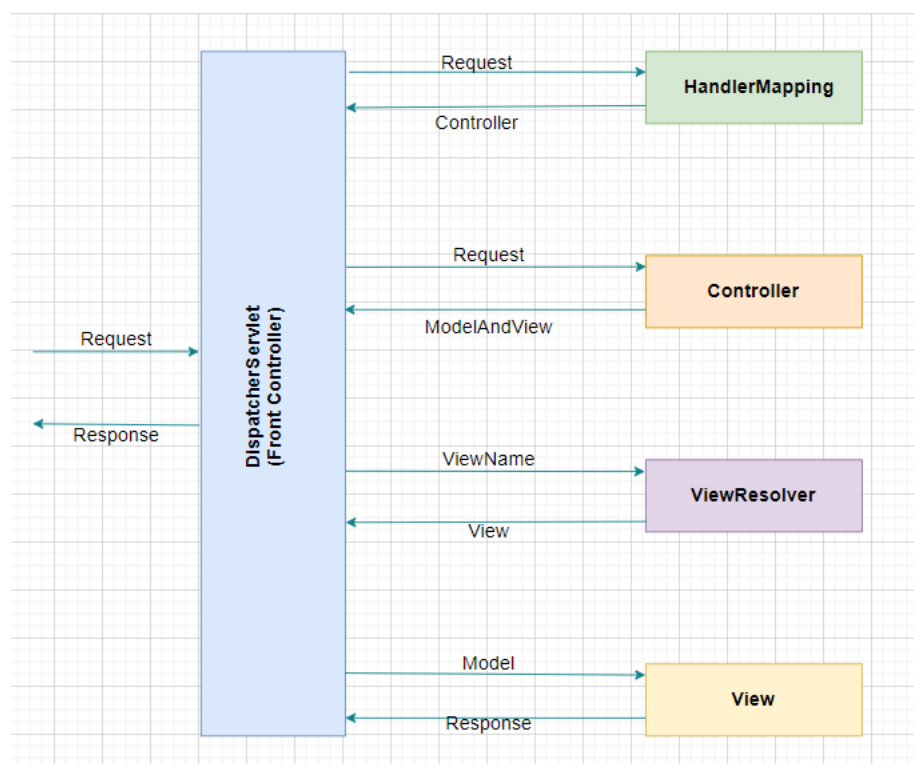
Hình 4. Minh họa cách thức hoạt động của Front Controller

Hình 4. cho ta thấy Front Controller là điểm tiếp xúc duy nhất với người dùng để xử lý tất cả các yêu cầu trong hệ thống. Front Controller sẽ sử dụng Handler Mapping để xem xét Controller nào sẽ xử lý request mà người dùng gửi đến và sau đó sẽ gọi các lớp Service tương ứng để xử lý yêu cầu. Sau khi nhận kết quả từ Controller, DispatcherServlet (xem mục 3.3.) sẽ tìm các view, sau đó sử dụng ViewResolver để

truyền dữ liệu vừa xử lý vào đó. View sẽ gửi phản hồi đến cho Front Controller và Front Controller gửi nó đến cho người dùng.

3.3. DispatcherServlet

DispatcherServlet được tích hợp hoàn toàn với IoC Container và cho phép sử dụng mọi tính năng mà Spring có. DispatcherServlet là một diễn đạt của Front Controller. Về cơ bản, một DispatcherServlet sẽ xử lý một yêu cầu HTTP Request được gửi đến, sau đó ủy quyền yêu cầu và xử lý yêu cầu theo các giao diện HandlerAdapter đã được cấu hình và triển khai trong ứng dụng cùng với các Annotation kèm theo và đối tượng Response.



Hình 5. Minh họa cách thức hoạt động của Dispatcher

3.3.1. HTTP Request Handling

DispatcherServlet chịu trách nhiệm chính trong việc gửi HTTP Request đến những nơi có chứa annotation `@Controller` hoặc `@RestController`. Sự khác biệt chính giữa `@Controller` và `@RestController` là cách Response được tạo ra. Phương thức xử lý request có thể chứa các loại tham số sau:

- `HttpServletRequest` hoặc `HttpServletResponse`.
- Các tham số trên url với annotation `@RequestParam`.

- Các thuộc tính model với annotation `@ModelAttribute`.
- Errors hoặc `BindingResult` để truy cập vào các kết buộc và kết quả kiểm tra cho đối tượng command.

3.3.2. View Resolver

`ViewResolver` được đính kèm với `DispatcherServlet` dưới dạng cài đặt cấu hình trên đối tượng `ApplicationContext`. Vị trí mặc định cho các view được chỉ định trong `WEB-INF`. `SpringMVC` cung cấp nhiều `ViewResolver` khác nhau để xác định view phù hợp, ví dụ như `InternalViewResolver`. Đường dẫn được chỉ định cho `InternalViewResolver` nằm trong thư mục con của `src/main/webapp`.

3.4. Cài đặt môi trường

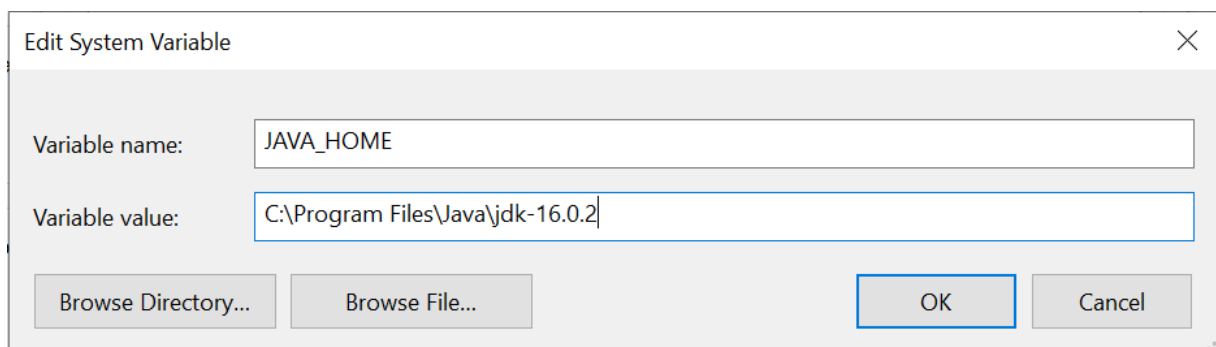
3.4.1. Cài bộ JDK để cài đặt môi trường Java

JDK (Java Development Kit) là một thành phần nền tảng quan trọng để xây dựng các ứng dụng Java. Trọng tâm của nó là trình biên dịch Java (Java compiler). Truy cập vào trang web sau để tải bộ JDK phiên bản mới nhất (ở thời điểm hiện tại viết tài liệu này là bộ JDK 16).

<https://www.oracle.com/java/technologies/javase-downloads.html>

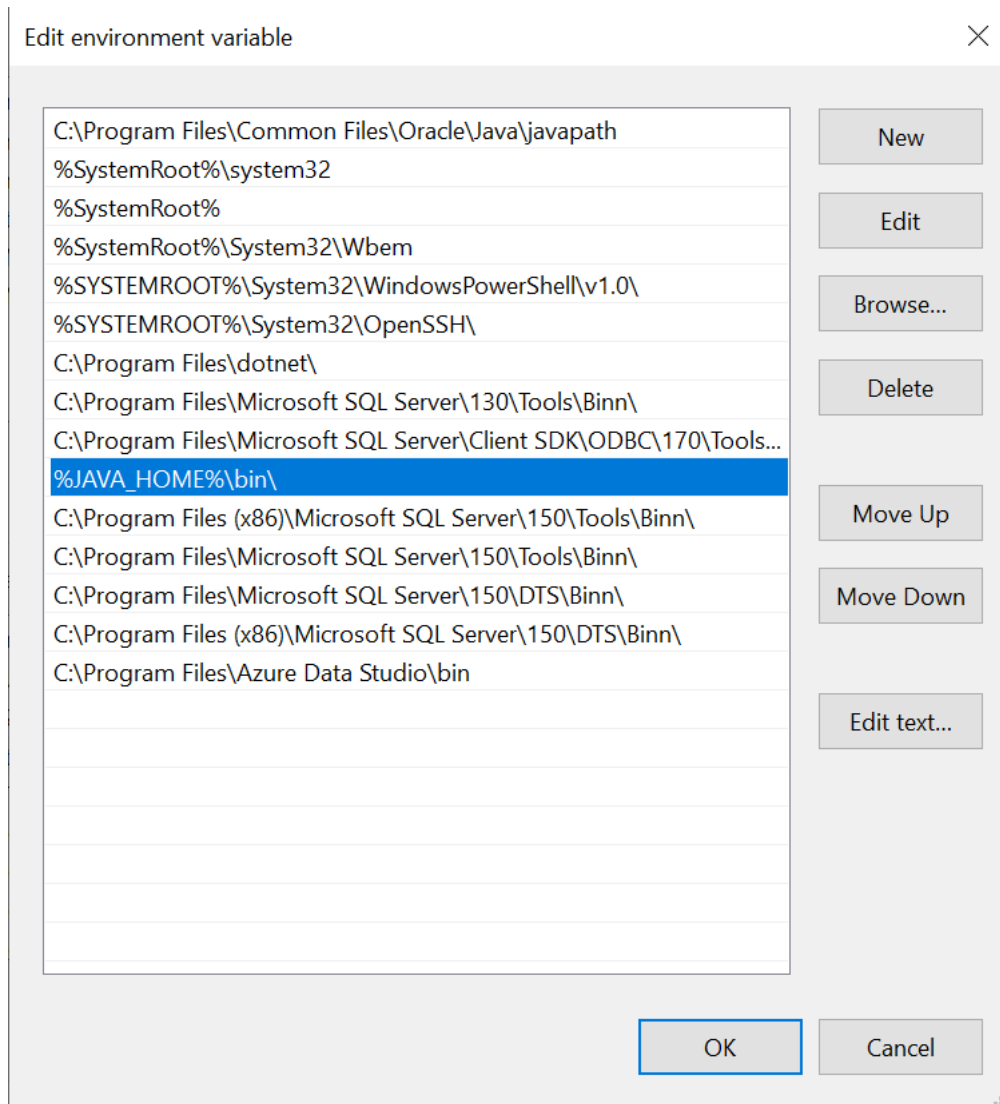
3.4.2. Cài đặt cấu hình môi trường

Đầu tiên, nhấp chuột phải vào biểu tượng This PC chọn Properties > click “Advanced system settings”. Cửa sổ System Properties hiện ra, ta chọn Environment Variables..., ở phần System variables, click New... để tạo biến môi trường mới, đặt tên, chỉ định đường dẫn đến bộ JDK vừa mới cài đặt và click OK.



Hình 6. Tạo mới biến môi trường cho Java

Cũng trong vùng đó, ra chọn biến Path và click nút Edit..., cửa sổ Edit environment variable hiện ra, click nút New để thêm một biến môi trường mới có giá trị là %JAVA_HOME%\bin\ nhằm trỏ đến thư mục bin của JDK. Tiếp tục click các nút OK để hoàn thành cài đặt môi trường.



Hình 7. Cài đặt biến môi trường cho Java

Mở cửa sổ Command Prompt để kiểm tra cài đặt thành công hay chưa bằng cách gõ các lệnh “java -version” và “javac -version”.

3.4.3. Cài đặt NetBeans IDE

Có nhiều công cụ IDE (Integrated Development Environment) để lập trình phát triển các ứng dụng Java như NetBeans IDE, IntelliJ IDE, Eclipse IDE,... Trong tất cả các minh họa trong đề án này sẽ được thực hiện trên NetBeans IDE. Ta có thể tải các

phiên bản NetBeans IDE mới nhất (ở thời điểm hiện tại viết tài liệu này là phiên bản 12.4) tại trang web sau:

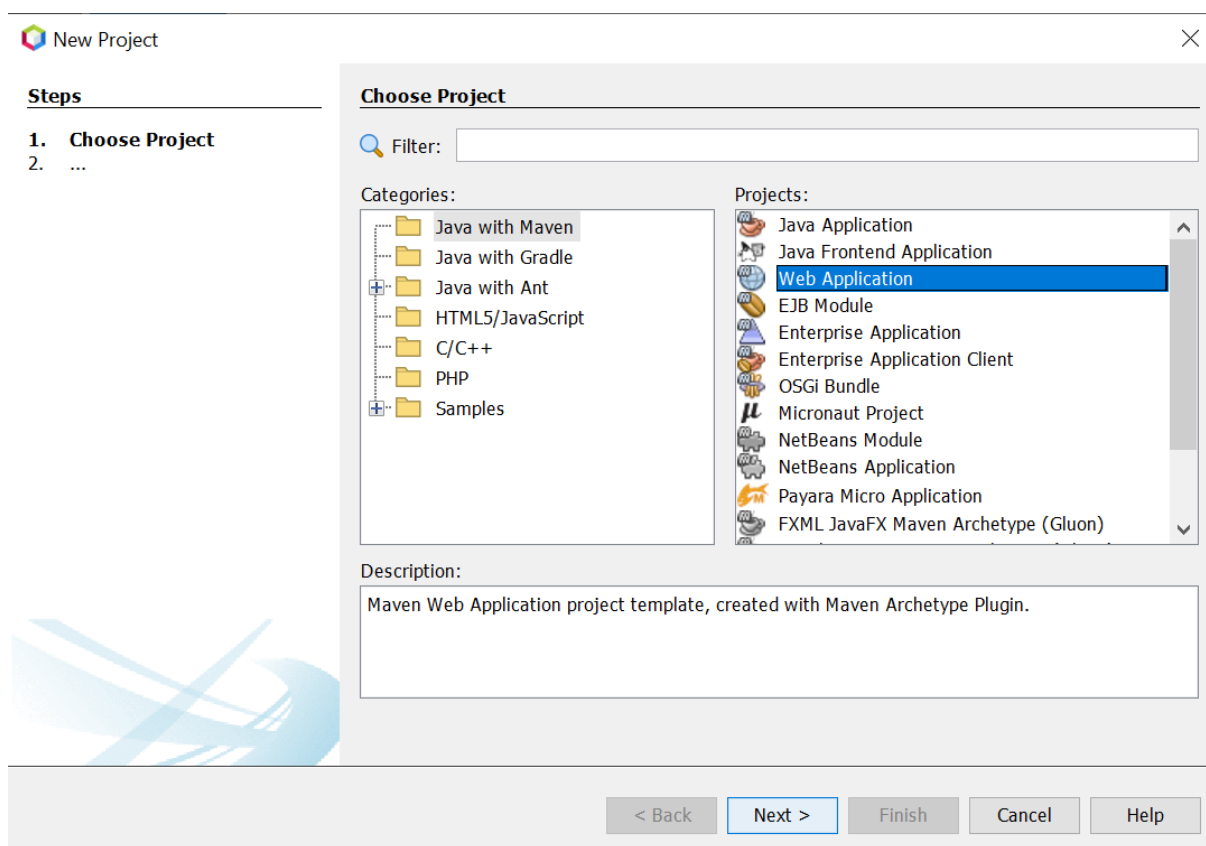
<https://netbeans.apache.org/download/>

3.4.4. Cấu hình Apache Tomcat Server

Apache Tomcat là một loại web server HTTP có khả năng hỗ trợ mạnh mẽ cho các ứng dụng Java thay vì các website tĩnh khác. Ta có thể tải Server Apache Tomcat ở trang sau <http://tomcat.apache.org/> ở tài liệu này sử dụng Apache Tomcat 9.

3.5. Viết chương trình đầu tiên với SpringMVC

Trong NetBeans tạo mới một project Java Maven phát triển Web Application.



Hình 8. Tạo mới project Java Maven phát triển Web Application

Nhấn Next và tiếp tục cài đặt các thông tin Project Name và Group Id.

New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Settings

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package: (Optional)

< Back **Next >** Finish Cancel Help

Hình 9. Thông tin project Java Maven phát triển Web Application

Click nút Next và cấu hình cho Server bằng cách click vào button Add > Apache Tomcat or TomEE, tại mục Server Location chỉ định đường dẫn tới vị trí thư mục giải nén Apache Tomcat Server đã tải về, cài đặt thông tin cho mục Username và Password sau đó click nút Finish.

New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Settings**

Settings

Server:

Java EE Version:

< Back Next > **Finish** Cancel Help

Hình 10. Thêm Server Apache Tomcat vào project ứng dụng Web

Nhấn Finish để hoàn thành quá trình tạo project. Một project Java Maven phát triển Web Application sẽ được tạo ra.

Trong Java Maven, tập tin pom.xml là nơi cấu hình định nghĩa các dependency cần thiết, Maven sẽ đọc tập tin này và tải các tập tin .jar từ thùng chứa trung tâm của Maven khi build project. Đầu tiên, thêm các dependency sau vào pom.xml và build project:

```
<dependency>

    <groupId>javax</groupId>

    <artifactId>javaee-web-api</artifactId>

    <version>8.0.1</version>

    <scope>provided</scope>
</dependency>
<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-webmvc</artifactId>

    <version>5.3.9</version>
</dependency>
<dependency>

    <groupId>javax.servlet</groupId>

    <artifactId>jstl</artifactId>

    <version>1.2</version>
</dependency>
```

Tạo gói com.ntpt.configs để lưu các lớp cấu hình project. Tạo lớp WebApplicationContextConfig.java:

```

package com.ntpt.configs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.ntpt.springmvcdemo")
public class WebApplicationContextConfig implements WebMvcConfigurer{

    @Override

    public void configureDefaultServletHandling(

        DefaultServletHandlerConfigurer configurer) {

        configurer.enable();

    }

    @Bean

    public InternalResourceViewResolver getInternalResourceViewResolver() {

        InternalResourceViewResolver resolver =

            new InternalResourceViewResolver();

```

```
resolver.setViewClass(JstlView.class);

resolver.setPrefix("/WEB-INF/jsp/");

resolver.setSuffix(".jsp");


return resolver;

}

}
```

Các annotation được chỉ định trong lớp trên:

@Configuration: chỉ ra lớp khai báo một hoặc nhiều phương thức **@Bean** và có thể được xử lý bởi Spring container để tạo ra các định nghĩa bean và yêu cầu các service cho bean đó trong thời gian chạy.

@EnableWebMVC: được sử dụng để cho phép hoạt động SpringMVC trong một ứng dụng.

@ComponentScan: được sử dụng để thông báo cho Spring biết và chỉ định các gói cơ sở bằng cách sử dụng các thuộc tính `basePackages` hoặc `basePackageClasses`.

Tiếp tục tạo lớp `DispatcherServletInitializer.java` nằm trong gói `com.ntpt.configs` và kế thừa `AbstractAnnotationConfigDispatcherServletInitializer`. Sau đó ghi đè tất cả các phương thức sau:

getRootConfigClasses(): được dùng để chỉ định các lớp cấu hình cho Root Application Context.

getServletConfigClasses(): được dùng để chỉ định các lớp cấu hình cho Dispatcher Servlet Application Context.

getServletMappings(): được dùng để chỉ định các Servlet Mappings cho DispatcherServlet.

```
package com.ntpt.configs;
```

```

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServlet
Initializer;

public class DispatcherServletInitializer

    extends AbstractAnnotationConfigDispatcherServletInitializer{

    @Override

    protected Class<?>[] getRootConfigClasses() {

        return null;

    }

    @Override

    protected Class<?>[] getServletConfigClasses() {

        return new Class[] {

            WebApplicationContextConfig.class

        };

    }

    @Override

    protected String[] getServletMappings() {

        return new String[] { "/" };

    }

}

```

Bước tiếp theo cần tạo JSP View, trong ví dụ này, các tập tin jsp sẽ nằm trong thư mục WEB-INF/jsp. Ta sẽ tạo thư mục WEB-INF/jsp nằm trong src/main/webapp, sau đó tạo tập tin hello.jsp trong thư mục jsps.

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

```

```
<html>

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <title>JSP Page</title>

</head>

<body>

    <h1>${message}</h1>

</body>

</html>
```

Trong đó, message là một biến và giá trị của nó sẽ được hiển thị trên trang web JSP, giá trị này sẽ được gán trong controller. Tiếp theo, tạo gói com.ntpt.controllers và lớp HomeController.java trong gói đó.

```
package com.ntpt.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {

    @RequestMapping("/")
    public String index(Model model){

        model.addAttribute("message", "Hello everyone!!!");

        return "hello";

    }

}
```

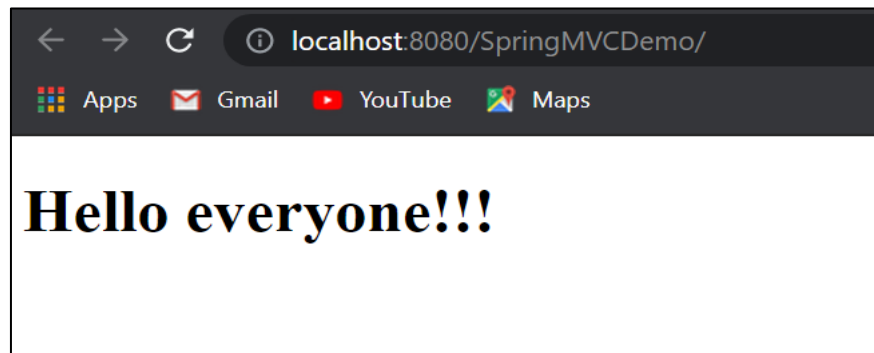
Một số annotation được sử dụng trong controller:

@Controller: chỉ định một lớp đóng vai trò là controller.

@RequestMapping: chỉ định một ánh xạ url tới một lớp hay một phương thức, có thuộc tính value để chỉ đường dẫn, thuộc tính method để chỉ định loại HTTP Request sẽ được xử lý (mặc định là HTTP GET). Ngoài ra Spring còn có các annotation @GetMapping, @PostMapping, @PutMapping và @DeleteMapping.

@ResponseBody: giá trị trả về của phương thức sẽ được ghi vào HTTP Response.

Để chạy project, nhấn chuột phải vào project, chọn Run để chạy server, ta sẽ có kết quả như sau:



3.6. Controller

Để định nghĩa Controller, ta chỉ cần thêm annotation @Controller vào lớp đó (org.springframework.stereotype.Controller). Nó có nhiệm vụ Mapping Request trên URL vào các phương thức xử lý tương ứng trong Controller. SpringMVC cung cấp các annotation sau sử dụng trong MVC Controller như: @RequestMapping (được xem là phương thức ánh xạ mặc định cho lớp Controller), @RequestParam, @ModelAttribute.

3.6.1. @PathVariable

@PathVariable dùng để truyền tham số trực tiếp trên đường dẫn của request.

Ví dụ tạo một tập tin demo.jsp trong WEB-INF như sau:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
```

```
<html>

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <title>JSP Page</title>

</head>

<body>

    <h1>${message}</h1>

</body>

</html>
```

Tiếp theo viết thêm phương thức vào lớp HomeController.java đã tạo trước đó.

```
@RequestMapping("/demo/{name}")

public String hello(Model model, @PathVariable(value = "name") String name){

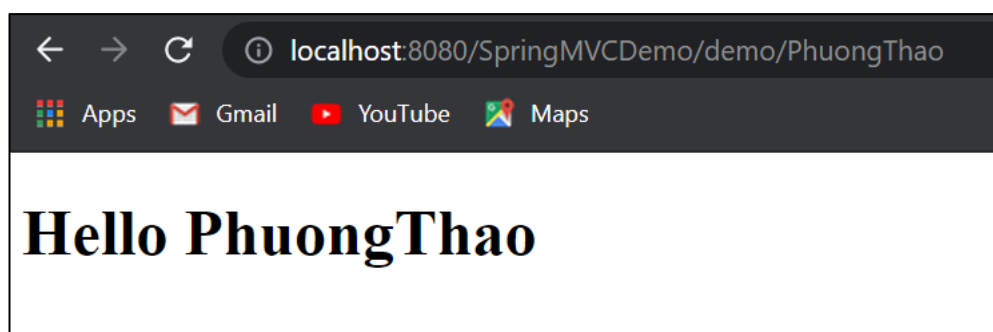
    model.addAttribute("message", "Hello " + name);

    return "demo";

}
```

Sau đó Run project và truy cập vào đường dẫn

<http://localhost:8080/SpringMVCDemo/demo/PhuongThao> (trong đó PhuongThao là tham số bất kì bạn có thể truyền vào) sẽ có kết quả như sau:



3.6.2. @RequestParam

@RequestParam dùng để lấy giá trị các tham số được truyền thông qua các tham số của HTTP GET. @RequestParam có thuộc tính rất quan trọng đó là required=false

để chỉ định đường dẫn có bắt buộc truyền tham số vào hay không. Ngoài ra còn có thuộc tính `defaultValue` để khai báo giá trị mặc định cho tham số.

Ví dụ cũng trong trang `hello.jsp`, ta tạo phương thức như sau:

```
@RequestMapping("/")

public String index(Model model,

    @RequestParam(value = "word", required = false) String word){

    if(word != null)

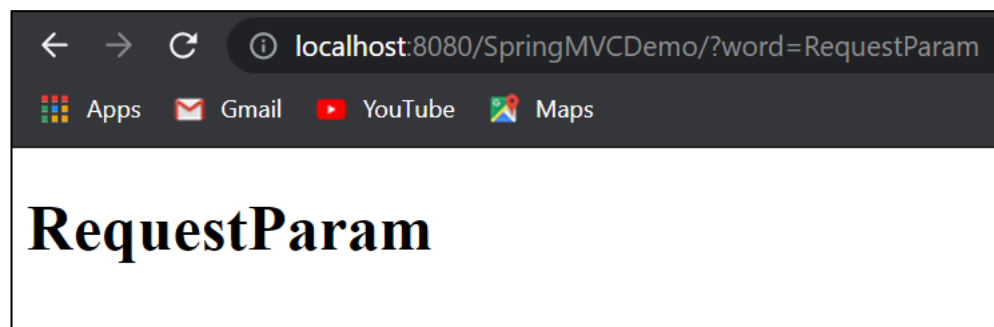
        model.addAttribute("message", word);

    return "hello";

}
```

Run project và truy cập vào đường dẫn dưới đây ta có kết quả như sau:

<http://localhost:8080/SpringMVCDemo/?word=RequestParam>



3.7. Tag Libraries

JavaServer Pages (JSP) là một công nghệ do Spring Framework cung cấp giúp triển khai các view, nó cung cấp một số thẻ để đánh giá lỗi, thiết lập chủ đề hay xuất các thông điệp (message). Mã nguồn Java được cho phép nhúng vào trong các tập tin JSP bằng cách chèn giữa cặp dấu `<% %>` hoặc thông qua các thẻ JSTL.

JSTL (JavaServer Pages Standard Tag Library) là một tập hợp các thẻ JSP được cung cấp bởi Oracle. Để import các thư viện thẻ vào tập tin `jsp`, ta sử dụng `taglib` với các thuộc tính: `prefix` là tên mà mình sử dụng để khai báo thẻ và `uri` là đường dẫn trỏ tới các tài nguyên thẻ.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Các lớp đối tượng liên kết với từng trường trong form của mình bằng việc sử dụng Model Attribute.

Ví dụ về gửi dữ liệu lên server bằng phương thức post.

Tạo gói com.ntpt.pojos và lớp Subject trong gói này như sau:

```
package com.ntpt.pojos;

public class Subject {

    private String id;

    private String name;

    //Các phương thức getter và setter

}
```

Trong lớp Controller tạo hai phương thức để gửi request lên server. Cả hai phương thức này có cùng URL nhưng khác nhau về thuộc tính method. Khi truy cập vào URL, HTTP Request sẽ mặc định gọi phương thức dạng GET nên hàm `getSubject()` sẽ được gọi. Sau khi nhấn submit, phương thức `addSubject()` sẽ được gọi thông qua request dạng POST. Ta tạo mới một Map để lưu các dữ liệu khi gọi phương thức POST, và tạo phương thức `getResult()` để lấy các dữ liệu vừa được gửi lên.

```
Map<String, String> result = new HashMap<>();

@RequestMapping(value = "/add")

public String getResult(Model model){

    model.addAttribute("subjects", result);

    return "add";

}

@RequestMapping(value = "/")

public String getSubject(Model model){
```

```

        Subject s = new Subject();

        model.addAttribute("subject", s);

        return "demo";
    }

@RequestMapping(value = "/", method = RequestMethod.POST)

    public String addSubject(Model model, @ModelAttribute(value = "subject")

                                                Subject s){

        if(result.get(s.getId()) == null){

            result.put(s.getId() , s.getName());

            return "redirect:/add";

        }else{

            model.addAttribute("message", "Mã môn đã tồn tại!");

            return "demo";

        }

    }
}

```

Trong tập tin demo.jsp, khai báo thư viện thẻ form của Spring.

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>Thêm môn học</title>

    </head>

```

```

<body>

  <form:form method="POST" modelAttribute="subject">

    <table>

      <tr>

        <td>Mã môn học</td>

        <td><form:input path="id" /></td>

      </tr>

      <tr>

        <td>Tên môn học</td>

        <td><form:input path="name" /></td>

      </tr>

      <h4 style="color: red">${message}</h4>

      <tr >

        <td >

          <input style="color: blue" type="submit" value="Thêm môn học" />

        </td>

      </tr>

    </table>

  </form:form>

</body>

</html>

```

Trong thẻ <form:form>, ta khai báo thuộc tính method là POST và thuộc tính modelAttribute có giá trị là subject dùng để lưu trữ đối tượng Subject vừa mới được tạo trong phương thức addSubject(). Trong các thẻ <form:input> có thuộc tính quan trọng là path, giá trị của thuộc tính này là tên trường của đối tượng Subject đã được lưu trong gói com.ntpt.pojo và giá trị nhập vào sẽ được kết buộc với trường tương

ứng. Khi submit form này thì phương thức `addSubject()` sẽ được gọi, đối tượng `subject` sẽ được truyền vào với các giá trị đã nhập trong form. Trước tham số “s” của đối tượng `Subject` sử dụng annotation `@ModelAttribute` để gán giá trị cho thuộc tính `value` của `@ModelAttribute` chính là giá trị được khai báo trong form. Chỉ định chuỗi trả về của phương thức sử dụng tiền tố `redirect` để Spring biết gọi một view đặc biệt `RedirectView`.

Tạo tập tin `add.jsp` để hiển thị kết quả sau khi thêm một môn học. Sử dụng thẻ `<c:forEach>` của Spring với `items` là `subjects` để chỉ định các mục sẽ được lặp lại.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Danh sách môn học</title>
  </head>
  <body>
    <table width="50%" border="1">
      <tr>
        <th width="20%">Mã môn học</th>
        <th width="80%">Tên môn học</th>
      </tr>
      <c:forEach items="${subjects.entrySet()}" var="s">
        <tr>
          <td>${s.getKey()}</td>
          <td>${s.getValue()}</td>
        </tr>
      </c:forEach>
    </table>
  </body>
</html>
```

```
</c:forEach>

</table>

</body>

</html>
```

Khi gọi dữ liệu lên trang add.jsp sẽ bị lỗi tiếng việt nên ta bổ sung phương thức sau vào DispatcherServletInitializer.java.

```
@Override
protected Filter[] getServletFilters() {

    CharacterEncodingFilter filter = new CharacterEncodingFilter();

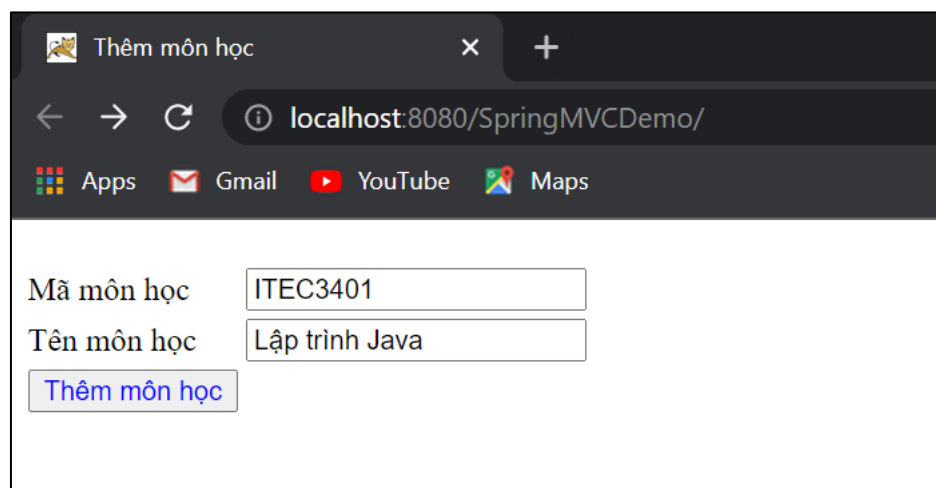
    filter.setEncoding("UTF-8");

    filter.setForceEncoding(true);

    return new Filter[] { filter };

}
```

Truy cập vào đường dẫn <http://localhost:8080/SpringMVCDemo/> và nhập các giá trị để thêm môn học.



Thêm môn học

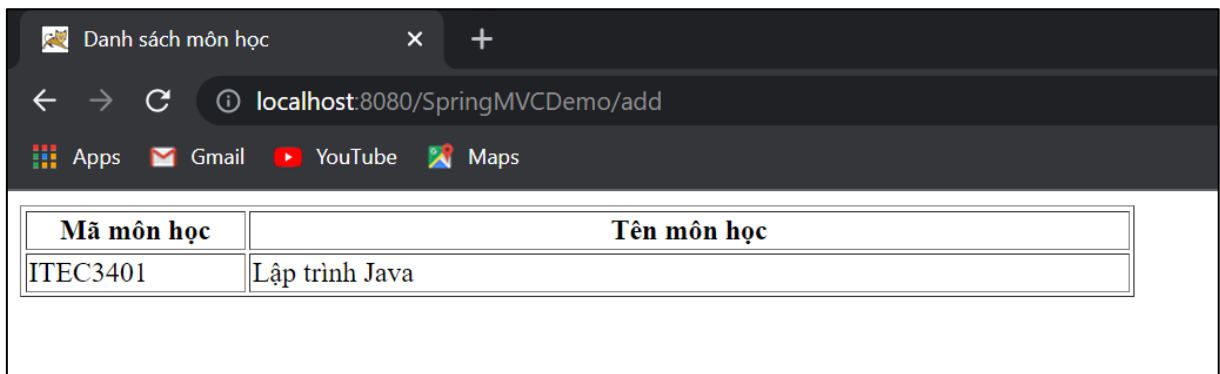
← → ↻ ⓘ localhost:8080/SpringMVCDemo/

Apps Gmail YouTube Maps

Mã môn học

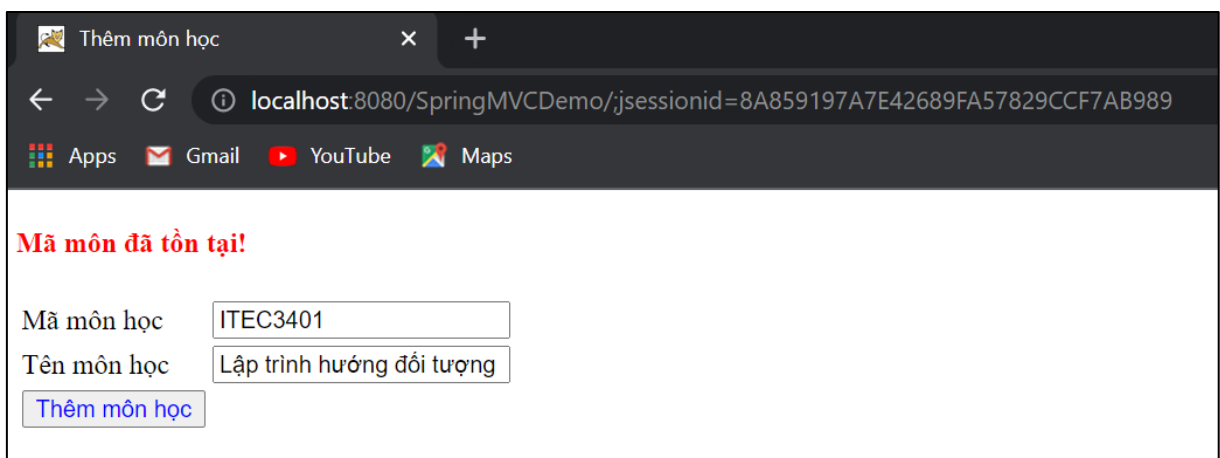
Tên môn học

Sau khi nhấn button Thêm môn học, kết quả sẽ chuyển sang trang Danh sách môn học và hiển thị môn vừa mới thêm vào.



Mã môn học	Tên môn học
ITEC3401	Lập trình Java

Nếu tiếp tục nhập mã môn học ITEC3401 để thêm thì sẽ hiển thị message thông báo mã môn đã tồn tại.



Mã môn đã tồn tại!

Mã môn học: ITEC3401

Tên môn học: Lập trình hướng đối tượng

[Thêm môn học](#)

WebDataBinder kế thừa lớp DataBinder dùng để liên kết các dữ liệu từ HttpServletRequest đến các đối tượng JavaBean, được thiết kế cho môi trường web nhưng không phụ thuộc vào API Servlet. Để điều chỉnh cách thức kết buộc dữ liệu(data binding), chúng ta tạo ra phương thức trong Controller với annotation là @InitBinder.

Giả sử trong lớp Subject.java ta thêm thuộc tính major.

```
public class Subject {
    private String id;
    private String name;
    private String major;

    //Các phương thức getter và setter
}
```

Trong trang demo.jsp, thêm hàng để nhập major trước nút submit.

```
<tr>

    <td>Chuyên ngành</td>

    <td><form:input path="major" /></td>

</tr>
```

Trong lớp Controller, thêm phương thức initBinder với annotation @InitBinder để chỉ định các trường được phép kết buộc dữ liệu.

```
@InitBinder

public void initBinder(WebDataBinder binder){

    binder.setAllowedFields("id", "name");

}
```

Thêm tham số kiểu BindingResult vào phương thức addUser() và thêm các dòng lệnh để kiểm tra kết buộc dữ liệu:

```
@RequestMapping(value = "/", method = RequestMethod.POST)

public String addUser(Model model, @ModelAttribute(value = "subject") Subject s,

    BindingResult res){

    if(res.getSuppressedFields().length > 0)

        throw new RuntimeException("Lỗi!");

    .....

}
```

Truy cập lại vào đường dẫn <http://localhost:8080/SpringMVCDemo/> và thêm một môn học mới, chương trình sẽ hiện thông báo có trường không được phép kết buộc dữ liệu. Để chương trình hoạt động lại bình thường, ta chỉ cần xóa đoạn HTML của trường major.

Thêm môn học

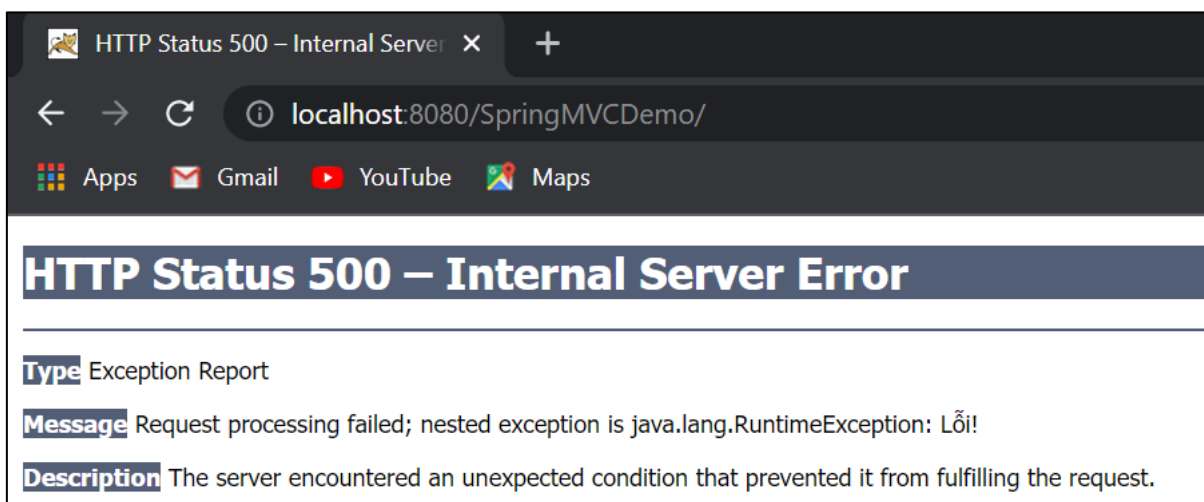
localhost:8080/SpringMVCDemo/

Apps Gmail YouTube Maps

Mã môn học

Tên môn học

Chuyên ngành



Để tiện lợi cho việc chỉnh sửa nội dung trang web hay phát triển trang web đa ngôn ngữ, ta cần sử dụng Properties File.

Trong tập tin demo.jsp, ta thêm dòng khai báo thêm thư viện Spring Tag và chỉ định văn bản từ ngoài được điền vào bằng thẻ `<spring:message>`. Thuộc tính code trong thẻ `<spring:message>` để chỉ định key và giá trị của nó sẽ được đọc khi thực thi chương trình trong tập tin property.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<!DOCTYPE html>
<html>
  <head>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Thêm môn học</title>

</head>

<body>

  <form:form method="POST" modelAttribute="subject">

    <table>

      <tr>

        <td><spring:message code="subject.id" /></td>

        <td><form:input path="id" /></td>

      </tr>

      <tr>

        <td><spring:message code="subject.name" /></td>

        <td><form:input path="name" /></td>

      </tr>

      <tr>

        <td><spring:message code="subject.add" /></td>

        <td><input style="color: blue" type="submit" value="<spring:message
code="subject.add" />" /></td>

      </tr>

    </table>

  </form:form>

</body>

```

</html>

Trong thư mục src/main/resources tạo tập tin message.properties có nội dung như sau:

subject.id = Mã môn học (Subject ID)

subject.name = Tên môn học (Subject Name)

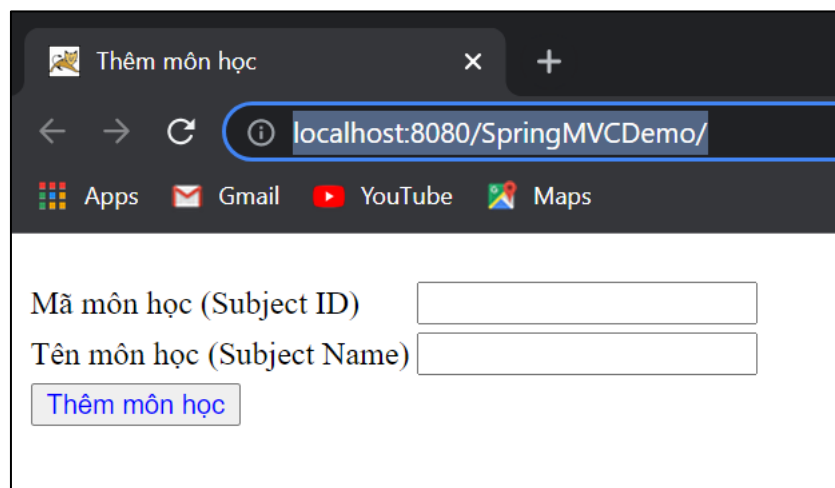
subject.add = Thêm môn học

Để kết nối dữ liệu trong tập tin properties với JSP view, ta cần cấu hình Bean cho lớp ResourceBundleMessageSource trong WebApplicationContextConfig.

@Bean

```
public MessageSource messageSource(){  
    ResourceBundleMessageSource resource =  
        new ResourceBundleMessageSource();  
    resource.setBasename("message");  
    return resource;  
}
```

Truy cập <http://localhost:8080/SpringMVCDemo/> ta có kết quả:



Thêm môn học

localhost:8080/SpringMVCDemo/

Apps Gmail YouTube Maps

Mã môn học (Subject ID)

Tên môn học (Subject Name)

3.8. View Resolver

3.8.1. Redirect

Redirect là kỹ thuật chuyển người dùng từ trang này sang trang khác với request tương ứng, thường được áp dụng khi submit form thành công để tránh tình trạng người dùng bấm submit nhiều lần hay refresh trình duyệt.

Spring cung cấp hai tiền tố để sử dụng chuyển trang là: forward và redirect.

- forward: là kỹ thuật chuyển người dùng từ trang này qua trang khác nhưng request được chuyển đến vẫn là request gốc ban đầu nên những thuộc tính được đặt trong model khi bắt đầu request vẫn còn giá trị.

- redirect: Spring sẽ tạo ra một request mới nên những thuộc tính được đặt trong model khi bắt đầu request sẽ bị mất đi.

Ví dụ trong trang demo.jsp tạo ra một siêu liên kết như sau:

```
<a href="<c:url value="/testRedirect"/>" > Redirect </a>
<a href="<c:url value="/testForward"/>" > Forward </a>
```

Tạo phương thức testRedirect() và testForward() trong lớp Controller.

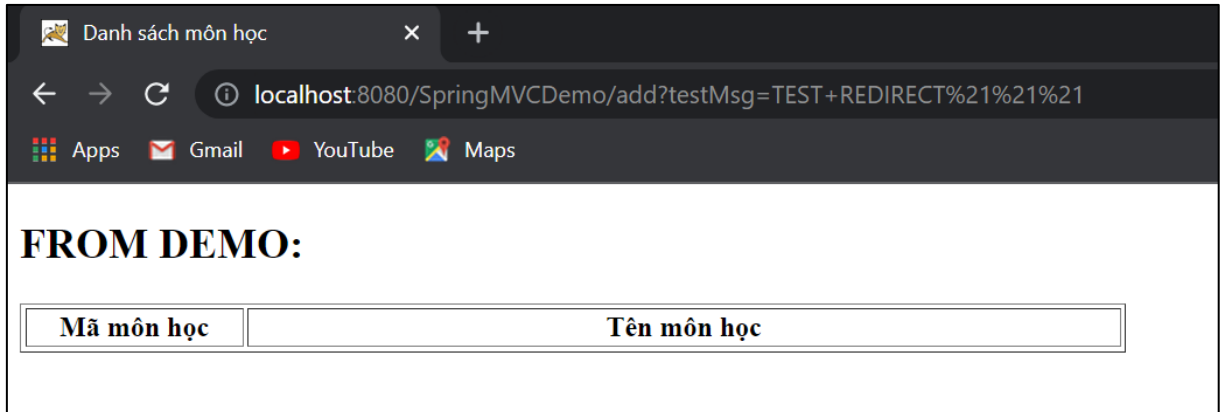
```
@RequestMapping(path = "/testRedirect")
public String testRedirect(Model model){
    model.addAttribute("testMsg", "TEST REDIRECT!!!");
    return "redirect:/add";
}

@RequestMapping(path = "/testForward")
public String testForward(Model model){
    model.addAttribute("testMsg", "TEST FORWARD!!!");
    return "forward:/add";
}
```

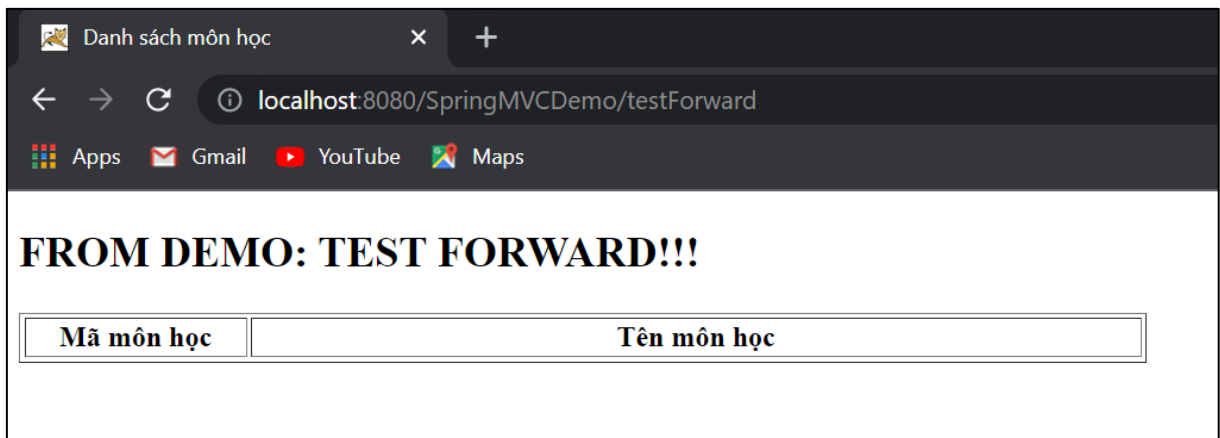
Trong trang add.jsp gọi giá trị của model chuyển đến.

```
<h2>FROM DEMO: ${testMsg}</h2>
```

Khi truy cập vào liên kết Redirect thuộc tính của model sẽ bị mất đi.



Truy cập vào liên kết Forward thuộc tính của model vẫn còn giá trị.

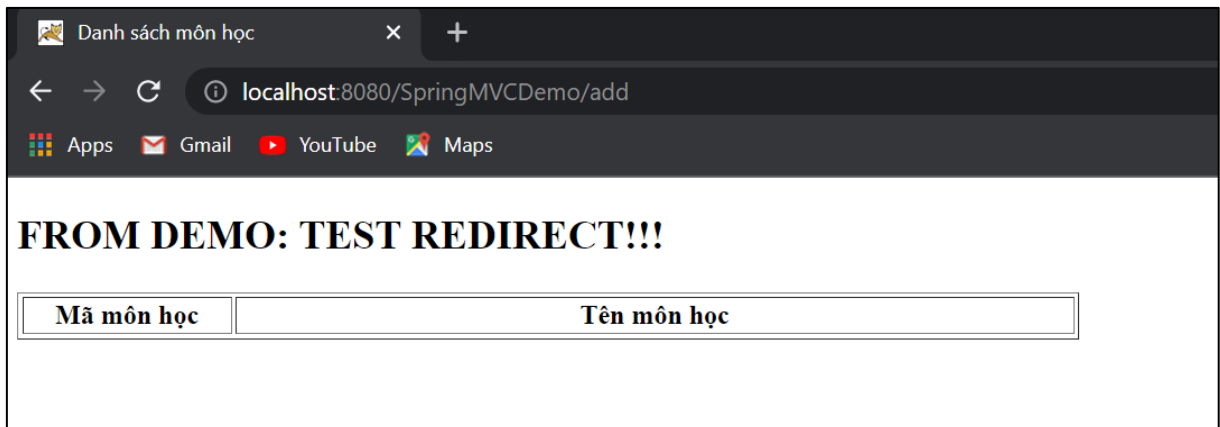


3.8.2. Thuộc tính Flash

Flash Attribute cung cấp cho ta cách truyền các model từ request này sang request khác bằng cách lưu trữ dữ liệu tạm thời trong session trước khi redirect và được xóa ngay sau khi redirect. Để sử dụng thuộc tính Flash ta cần thêm tham số RedirectAttributes cho các phương thức trong controller.

```
@RequestMapping(path = "/testRedirect")  
  
public String testRedirect(Model model, RedirectAttributes red){  
  
    red.addFlashAttribute("testMsg", "TEST REDIRECT!!!");  
  
    return "redirect:/add";  
  
}
```

Như vậy, khi truy cập vào <http://localhost:8080/SpringMVCDemo/add> nó sẽ hiển thị testMsg.



3.8.3. Static Resources

Để tích hợp các tài nguyên tĩnh trong ứng dụng web với Spring, ta tạo thư mục `src/main/webapp/resources/images` và sao chép các tập tin ảnh vào thư mục này.

Ghi đè phương thức `addResourceHandlers()` trong `WebApplicationContextConfig` để chỉ định vị trí các tài nguyên tĩnh. Trong đó, phương thức `addResourceHandler()` chỉ định request path trở tới thư mục chứa tài nguyên và phương thức `addResourceLocations()` chỉ định đường dẫn cơ sở của các tài nguyên đó.

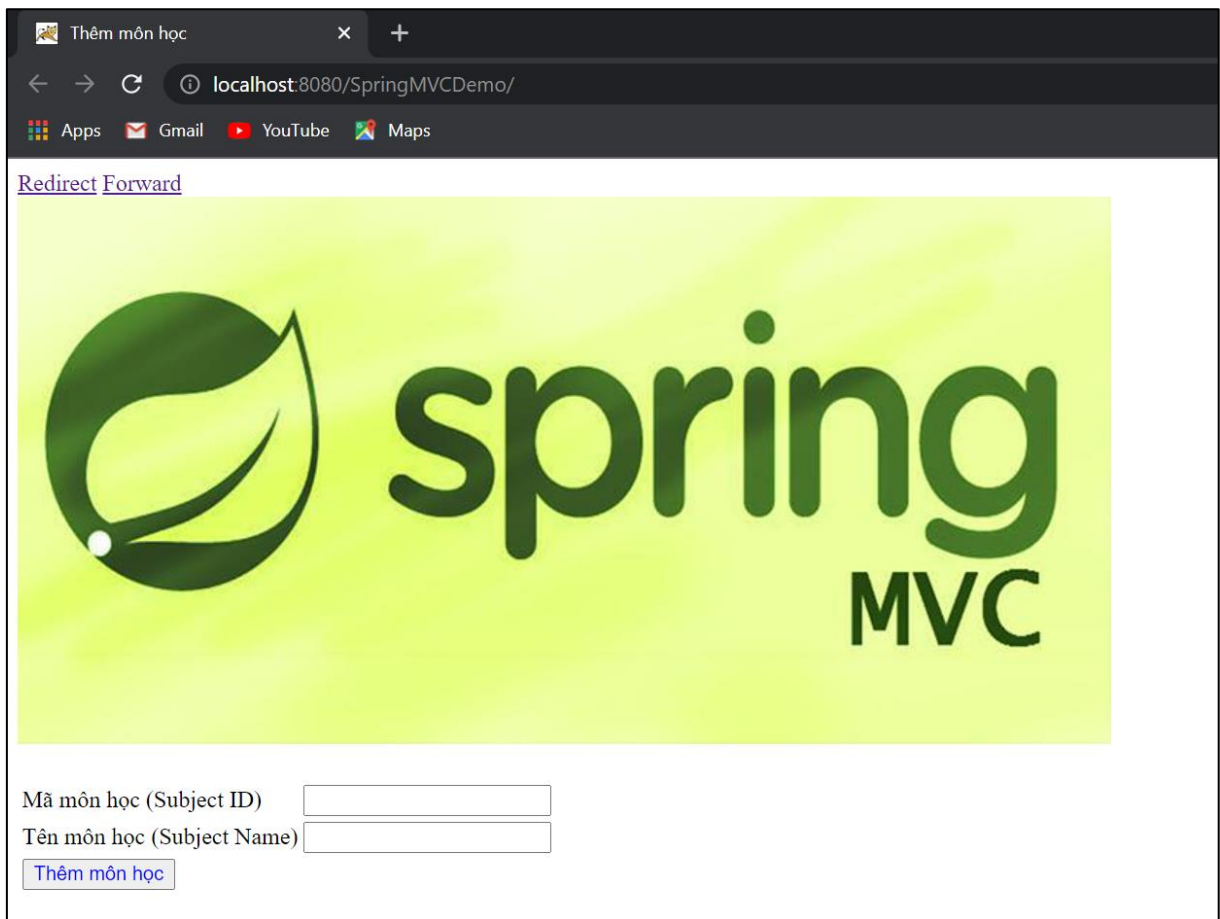
@Override

```
public void addResourceHandlers(ResourceHandlerRegistry registry){  
    registry.addResourceHandler("/img/**")  
        .addResourceLocations("/resources/images/");  
}
```

Trong trang `demo.jsp`, dùng thẻ `` để chỉ định đường dẫn tới ảnh như sau:

```
<div>  
    " alt="test"/>  
</div>
```

Truy cập <http://localhost:8080/SpringMVCDemo/> ta được:



3.8.4. Multipart Request

Multipart Request là một dạng request để gửi dữ liệu dưới dạng tập tin lên server.

Ta cần cấu hình Bean cho lớp CommonsMultipartResolver trong `WebApplicationContextConfig` để chỉ định một request có được chứa nội dung multipart và chuyển request thành các tham số và tập tin multipart hay không.

@Bean

```
public CommonsMultipartResolver multipartResolver() {  
  
    CommonsMultipartResolver resolver = new CommonsMultipartResolver();  
  
    resolver.setDefaultEncoding("UTF-8");  
  
    return resolver;  
}
```

Thêm hai dependency dưới đây vào tập tin pom.xml để hỗ trợ xử lý upload tập tin.

```

<dependency>

    <groupId>commons-fileupload</groupId>

    <artifactId>commons-fileupload</artifactId>

    <version>1.4</version>

</dependency>

<dependency>

    <groupId>commons-io</groupId>

    <artifactId>commons-io</artifactId>

    <version>2.11.0</version>

</dependency>

```

Trong tập tin message.properties bổ sung dòng sau:

```
subject.image=Ảnh minh họa
```

Thêm thuộc tính img kiểu MultipartFile vào lớp Subject trong gói com.ntpt.pojo.

```

public class Subject {

    private String id;

    private String name;

    private MultipartFile img;

    //Các thuộc tính getter, setter

}

```

Trong tập tin demo.jsp thêm khai báo thêm enctype="multipart/form-data" trong thẻ <form:form> để chỉ định dữ liệu trong form sẽ được mã hóa thành multipart request khi submit form. Khai báo thuộc tính type="file" trong thẻ <form:input> để tạo button hiển thị lựa chọn tập tin.

```

<form:form method="POST" modelAttribute="subject"

                enctype="multipart/form-data">

```



```

<table>

    .....

    <tr>

        <td><spring:message code="subject.image" /></td>

        <td><form:input path="img" type="file" /></td>

    </tr>

    .....

</table>

</form:form>

```

Chỉnh sửa phương thức addSubject() trong Controller.

```

@RequestMapping(value = "/", method = RequestMethod.POST)

public String addUser(Model model, @ModelAttribute(value = "subject") Subject s,

    HttpServletRequest request){

    if(result.get(s.getId()) == null){

        MultipartFile img = s.getImg();

        String rootDir =

            request.getSession().getServletContext().getRealPath("/");

        if (img != null && !img.isEmpty()) {

            try {

                img.transferTo(new File(rootDir + "resources/images/" + s.getId() +

                                                                    ".png"));

            } catch (IOException | IllegalStateException ex) {

                System.err.println(ex.getMessage());

            }

        }

    }

```

```

        result.put(s.getId() , s.getName());

        return "redirect:/add";

    }else{

        model.addAttribute("message", "Mã môn đã tồn tại!");

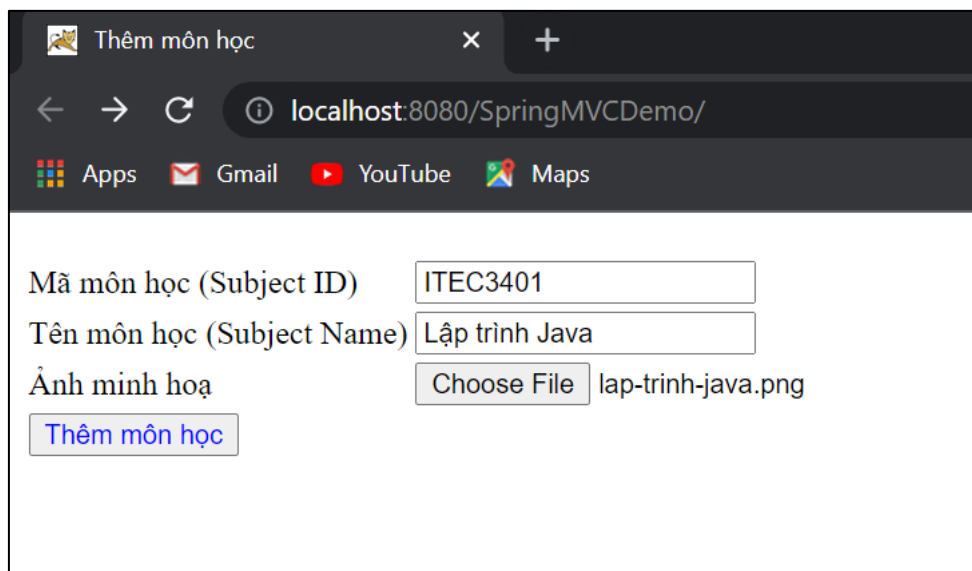
        return "demo";

    }

}

```

Truy cập <http://localhost:8080/SpringMVCDemo/> để thêm tập tin.



Thêm môn học

← → ↻ ⓘ localhost:8080/SpringMVCDemo/

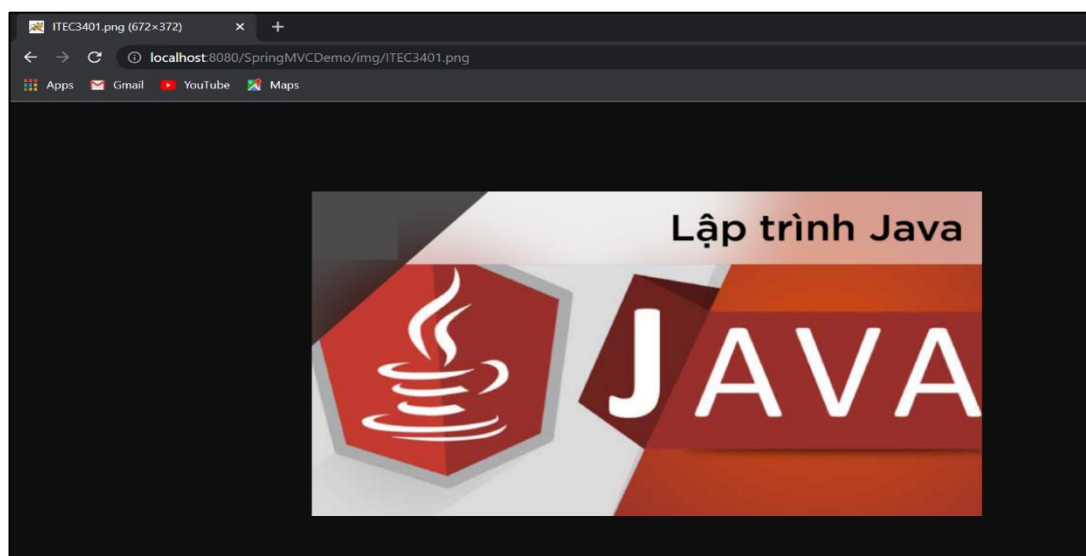
Apps Gmail YouTube Maps

Mã môn học (Subject ID)

Tên môn học (Subject Name)

Ảnh minh họa lap-trinh-java.png

Truy cập đường dẫn <http://localhost:8080/SpringMVCDemo/img/ITEC3401.png/> để xem hình ảnh vừa được upload.



3.9. Template with tiles

Apache Tiles là một framework mã nguồn mở giúp đơn giản hóa việc phát triển giao diện người dùng cho ứng dụng web. Tiles cho phép người dùng định nghĩa các trang con có thể tập hợp thành một trang web hoàn chỉnh khi thực thi chương trình. Điều này giúp tăng khả năng tái sử dụng và giảm thiểu số lượng trùng lặp mã nguồn.

Để sử dụng ta bổ sung dependency sau vào tập tin pom.xml

```
<dependency>

    <groupId>org.apache.tiles</groupId>

    <artifactId>tiles-extras</artifactId>

    <version>3.0.8</version>

</dependency>
```

Ví dụ tạo ra một trang web gồm ba phần chính: header, content và footer. Trước hết tạo tập tin tiles.xml trong thư mục WEB-INF, đây là một tập tin rất quan trọng trong Apache Tiles. Ta dùng thẻ <definition> để chỉ định các định nghĩa, mỗi định nghĩa có nhiều thuộc tính được chỉ định bằng thẻ <put-attribute> nằm trong <definition>.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE tiles-definitions PUBLIC "-//Apache
Software Foundation//DTD Tiles Configuration 3.0//EN"
"http://tiles.apache.org/dtds/tiles-config_3_0.dtd">

<tiles-definitions>

    <definition name="baseLayout" template="/WEB-INF/layout/base.jsp">

        <put-attribute name="title" value="" />

        <put-attribute name="header" value="/WEB-INF/layout/header.jsp" />

        <put-attribute name="content" value="" />

        <put-attribute name="footer" value="/WEB-INF/layout/footer.jsp" />
```

```

</definition>

<definition name="demo" extends="baseLayout">

    <put-attribute name="title" value="Trang chủ" />

    <put-attribute name="content" value="/WEB-INF/jsp/demo.jsp" />

</definition>

</tiles-definitions>

```

Tạo thư mục layout trong thư mục WEB-INF, trong thư mục này lần lượt tạo các tập tin base.jsp, header.jsp, footer.jsp để định nghĩa các template trong tiles.xml. Để chỉ định giá trị cho thuộc tính <put-attribute> ta thông qua thẻ <tiles:insertAttribute name=" " />.

Tập tin base.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib prefix="tiles" uri="http://tiles.apache.org/tags-tiles"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title><tiles:insertAttribute name="title" /></title>

        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

    </head>

    <body>

        <tiles:insertAttribute name="header" />

        <tiles:insertAttribute name="content" />

```

```
<tiles:insertAttribute name="footer" />

</body>

</html>
```

Tập tin header.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>

<h2>ĐÂY LÀ PHẦN HEADER</h2>
```

Tập tin footer.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>

<h2>ĐÂY LÀ PHẦN FOOTER</h2>
```

Để cấu hình cho Spring Tiles, ta tạo lớp TilesConfig.java trong gói com.ntpt.configs.

```
@Configuration

public class TilesConfig {

    @Bean

    public UrlBasedViewResolver viewResolver() {

        UrlBasedViewResolver viewResolver = new UrlBasedViewResolver();

        viewResolver.setViewClass(TilesView.class);

        viewResolver.setOrder(-2);

        return viewResolver;

    }

    @Bean

    public TilesConfigurer tilesConfigurer() {

        TilesConfigurer configurer = new TilesConfigurer();

        configurer.setDefinitions("/WEB-INF/tiles.xml");

    }

}
```

```

    configurer.setCheckRefresh(true);

    return configurer;
}
}

```

Trong lớp DispatcherServletInitializer ta chỉnh sửa phương thức `getRootConfigClasses()` như sau:

```

@Override

protected Class<?>[] getRootConfigClasses() {

    return new Class[] {

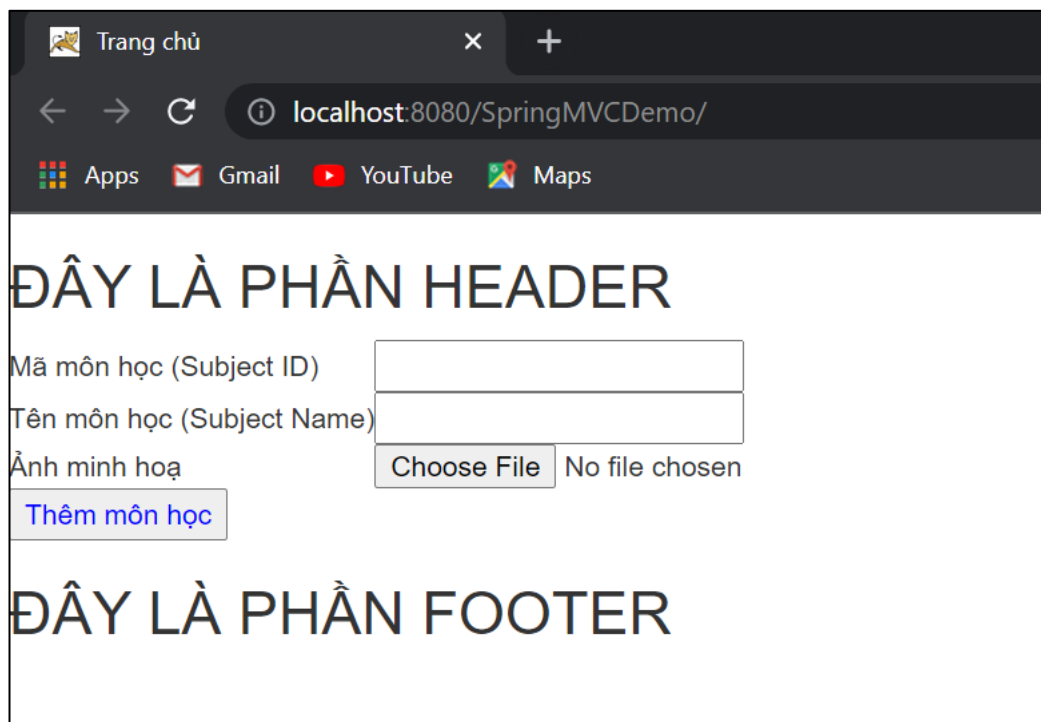
        TilesConfig.class

    };

}

```

Tiếp theo, trong tập tin `demo.jsp` ta loại bỏ phần html đã được chỉ định trong tập tin `base.jsp`. Chạy thử chương trình, ta được kết quả sau đây:



Trang chủ

localhost:8080/SpringMVCDemo/

Apps Gmail YouTube Maps

ĐÂY LÀ PHẦN HEADER

Mã môn học (Subject ID)

Tên môn học (Subject Name)

Ảnh minh họa No file chosen

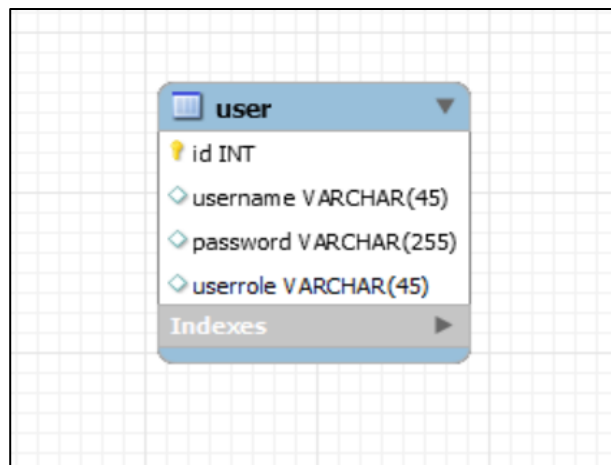
ĐÂY LÀ PHẦN FOOTER

3.10. Spring Security

Spring Security là một framework cung cấp cơ chế kiểm soát và xác thực dùng để bảo mật các ứng dụng dựa trên Spring. Spring Security tập trung vào việc cung cấp cả xác thực (authentication) và phân quyền (authorization) cho các ứng dụng Java.

Ví dụ về chương trình đăng nhập và đăng ký với Spring Security kết hợp tương tác với cơ sở dữ liệu quan hệ MySQL thông qua Hibernate.

Đầu tiên ta tạo cơ sở dữ liệu đơn giản với tên demo và trong đó tạo bảng user.



Trong tập tin pom.xml, ta thêm các dependency dưới đây:

```
<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-web</artifactId>

    <version>5.5.2</version>

</dependency>

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-config</artifactId>

    <version>5.5.2</version>

</dependency>

<dependency>
```

```

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.5.7.Final</version>

</dependency>

<dependency>

    <groupId>mysql</groupId>

    <artifactId>mysql-connector-java</artifactId>

    <version>8.0.26</version>

</dependency>

<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-orm</artifactId>

    <version>5.3.10</version>

</dependency>

```

Trong thư mục src/main/resources, ta tạo thêm tập tin database.properties để chứa thông tin cấu hình kết nối với cơ sở dữ liệu.

```

hibernate.dialect=org.hibernate.dialect.MySQLDialect

hibernate.showSql=true

hibernate.connection.driverClass=com.mysql.cj.jdbc.Driver

hibernate.connection.url=jdbc:mysql://localhost:3306/demo

hibernate.connection.username=root

hibernate.connection.password=12345678

```

Tiếp theo, ta tạo lớp HibernateConfig.java và SpringSecurityConfig.java trong gói com.ntpt.config để chứa các tập tin cấu hình bằng mã nguồn java.

```

@Configuration

@PropertySource("classpath:database.properties")

```



```

public class HibernateConfig {

    @Autowired

    private Environment env;

    @Bean

    public LocalSessionFactoryBean getSessionFactory() {

        LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

        sessionFactory.setPackagesToScan("com.ntpt.pojos");

        sessionFactory.setDataSource(dataSource());

        sessionFactory.setHibernateProperties(hibernateProperties());

        return sessionFactory;
    }

    @Bean

    public DataSource dataSource() {

        DriverManagerDataSource dataSource = new DriverManagerDataSource();

        dataSource.setDriverClassName(

            env.getProperty("hibernate.connection.driverClass"));

        dataSource.setUrl(env.getProperty("hibernate.connection.url"));

        dataSource.setUsername(env.getProperty("hibernate.connection.username"));

        dataSource.setPassword(env.getProperty("hibernate.connection.password"));

        return dataSource;
    }

    private Properties hibernateProperties() {

        Properties props = new Properties();

        props.put(DIALECT, env.getProperty("hibernate.dialect"));

        props.put(SHOW_SQL, env.getProperty("hibernate.showSql"));
    }
}

```

```

        return props;
    }

    @Bean
    public HibernateTransactionManager transactionManager() {
        HibernateTransactionManager transactionManager = new
                                HibernateTransactionManager();

        transactionManager.setSessionFactory(getSessionFactory().getObject());

        return transactionManager;
    }
}

```

```

@EnableWebSecurity

@ComponentScan(basePackages = {"com.ntpt ",})

public class SpringSecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired

    private UserDetailsService userDetailsService;

    @Bean

    public BCryptPasswordEncoder passwordEncoder() {

        return new BCryptPasswordEncoder();

    }

    @Override

    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.userDetailsService(userDetailsService).

                                passwordEncoder(passwordEncoder());

    }
}

```

```

    }

    @Override

    protected void configure(HttpSecurity http) throws Exception {

        http.formLogin().loginPage("/login").usernameParameter("username").

                                passwordParameter("password");

        http.formLogin().defaultSuccessUrl("/").failureUrl("/login?error");

        http.logout().logoutSuccessUrl("/login");

        http.csrf().disable();

    }

}

```

Trong lớp DispatcherServletInitializer ta chỉnh sửa phương thức
getRootConfigClasses() như sau:

```

@Override

protected Class<?>[] getRootConfigClasses() {

    return new Class[] {

        TilesConfig.class,

        SpringSecurityConfig.class,

        HibernateConfig.class

    };

}

```

Tạo lớp SecurityWebApplicationInitializer.java trong gói com.dht.config

```

public class SecurityWebApplicationInitializer extends

                                AbstractSecurityWebApplicationInitializer{

}

```

Trong gói com.ntpt.pojos tạo lớp User.java và kết nối với cơ sở dữ liệu bằng annotation @Table(name = "user").

```
@Entity
@Table(name = "user")

public class User implements Serializable{

    public static final String USER = "ROLE_USER";

    public static final String ADMIN = "ROLE_ADMIN";

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    @Size(min = 1, max = 45, message = "{user.username.error.sizeMsg}")

    private String username;

    @NotEmpty(message = "{user.password.error.sizeMsg}")

    private String password;

    private String userrole;

    //Các phương thức getter và setter

}
```

Tạo trang register.jsp trong WEB-INF/jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<div class="container">

    <h1 class="text-center text-uppercase">
```

```

    <spring:message code="user.register" />

</h1>

<c:url value="/register" var="action" />

<form:form action="\${action}" modelAttribute="user" method="post" >

    <form:errors path="*" cssClass="alert alert-danger" element="div" />

    <div class="form-group">

        <label for="usernameId">

            <spring:message code="user.username" />

        </label>

        <form:input path="username" id="usernameId" cssClass="form-control" />

        <form:errors path="username" cssClass="text-danger" />

    </div>

    <div class="form-group">

        <label for="passwordId">

            <spring:message code="user.password" />

        </label>

        <form:input id="passwordId" path="password" cssClass="form-control"
type="password" />

        <form:errors path="password" cssClass="text-danger" />

    </div>

    <div class="form-group">

        <form:button class="pull-right">

            <spring:message code="user.register" />

        </form:button>

    </div>

</form:form>

```

</div>

Tiếp theo ta tạo gói `com.ntpt.repository` để xử lý truy vấn dữ liệu từ cơ sở dữ liệu, trong gói này ta tạo interface `UserRepository` như sau:

```
public interface UserRepository {  
  
    void addUser(User user);  
  
    List<User> getUsers(String username);  
  
}
```

Tạo gói `com.ntpt.repository.impl` để hiện thực hóa giao diện, trong gói này tạo lớp `UserRepositoryImpl.java`

```
@Repository  
@Transactional  
  
public class UserRepositoryImpl implements UserRepository{  
  
    @Autowired  
  
    private SessionFactory sessionFactory;  
  
    @Override  
  
    public void addUser(User user) {  
  
        sessionFactory.getCurrentSession().save(user);  
  
    }  
  
    @Override  
  
    public List<User> getUsers(String username) {  
  
        Session session = sessionFactory.getCurrentSession();  
  
        CriteriaBuilder builder = session.getCriteriaBuilder();  
  
        CriteriaQuery<User> cr = builder.createQuery(User.class);  
  
        Root<User> root = cr.from(User.class);  
  
        CriteriaQuery<User> query = cr.select(root);  
  

```

```

        if(!username.isEmpty())

            query.where(builder.equal(root.get("username"), username));

        List<User> users = session.createQuery(query).getResultList();

        return users;

    }

}

```

Tạo gói com.ntpt.service để chứa các xử lý nghiệp vụ tương tác với cơ sở dữ liệu, trong gói này ta tạo interface UserService như sau:

```

public interface UserService extends UserDetailsService{

    void addUser(User user);

    User getUserByUsername(String username);

}

```

Tạo gói com.ntpt.service.impl để hiện thực hóa giao diện, trong gói này tạo lớp UserServiceImpl.java

```

@Service

public class UserServiceImpl implements UserService{

    @Autowired

    private UserRepository userRepository;

    @Autowired

    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Override

    public void addUser(User user) {

        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));

        user.setUserrole(User.USER);

        this.userRepository.addUser(user);

    }

}

```

@Override

```
public User getUserByUsername(String username) {  
    return userRepository getUsers(username).get(0);  
}
```

@Override

```
public UserDetails loadUserByUsername(String username) throws  
                                UsernameNotFoundException {  
    List<User> users = userRepository getUsers(username);  
    if (users.isEmpty())  
        throw new UsernameNotFoundException("User không tồn tại!");  
    User u = users.get(0);  
    Set<GrantedAuthority> authorities = new HashSet<>();  
    authorities.add(new SimpleGrantedAuthority(u.getUserrole()));  
    return new org.springframework.security.core.userdetails.User(u.getUsername(),  
u.getPassword(), authorities);  
}
```

Tạo trang login.jsp trong WEB-INF/jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>  
<% @taglib prefix="form" uri="http://www.springframework.org/tags/form" %>  
<% @taglib prefix="spring" uri="http://www.springframework.org/tags" %>  
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
  
<div class="container">
```



```

<h1 class="text-center text-uppercase">

    <spring:message code="user.login" />

</h1>

<c:if test="${param.error != null}">

    <div class="alert alert-danger">

        <p><spring:message code="user.login.error1" /></p>

    </div>

</c:if>

<c:url value="/login" var="action" />

<form:form action="${action}" method="post" >

    <div class="form-group">

        <label for="usernameId">

            <spring:message code="user.username" />

        </label>

        <input name="username" id="usernameId" class="form-control" />

    </div>

    <div class="form-group">

        <label for="passwordId">

            <spring:message code="user.password" />

        </label>

        <input id="passwordId" name="password" class="form-control"
type="password" />

    </div>

    <div class="form-group">

        <button class="pull-right" type="submit">

            <spring:message code="user.login" />


```

```
</button>

</div>

</form:form>

</div>
```

Mở header.jsp thêm hai siêu liên kết đăng nhập và đăng ký như dưới đây:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<nav class="navbar navbar-expand-sm bg-light">

  <ul class="navbar-nav">

    <c:choose>

      <c:when test="${pageContext.request.userPrincipal.name == null}">

        <li class="nav-item">

          <a href="<spring:url value="/register" />">

            <i class="fa fa-check"></i>

            <spring:message code="message.register" />

          </a>

        </li>

        <li class="nav-item">

          <a href="<spring:url value="/login" />">

            <i class="fa fa-user"></i>

            <spring:message code="message.login" />

          </a>

        </li>

      </c:when>
```

```

<c:when test="{pageContext.request.userPrincipal.name != null}">

    <li class="nav-item">

        <a href="#"><i class="fa fa-user"></i>

            <spring:message code="message.welcome" />

            {pageContext.request.userPrincipal.name}

        </a>

    </li>

    <li class="nav-item">

        <a href="{spring:url value="/logout" />">

            <i class="fa fa-sign-out"></i>

            <spring:message code="message.logout" />

        </a>

    </li>

</c:when>

</c:choose>

</ul>

</nav>

```

Tạo UserController trong com.ntpt.controllers như sau:

```

@Controller

public class UserController {

    @Autowired

    private UserService userService;

    @GetMapping(value = "/register")

    public String registerView(Model model) {

        model.addAttribute("user", new User());
    }
}

```

```

        return "register";
    }

    @PostMapping(value = "/register")
    public String registerProcess(@ModelAttribute(name = "user") @Valid User user,
        BindingResult result) {
        if (result.hasErrors())
            return "register";

        userService.addUser(user);

        return "redirect:/login";
    }

    @GetMapping(value = "/login")
    public String loginView(Model model) {
        model.addAttribute("user", new User());

        return "login";
    }
}

```

Bổ sung thêm các khai báo sau trong message.properties

```

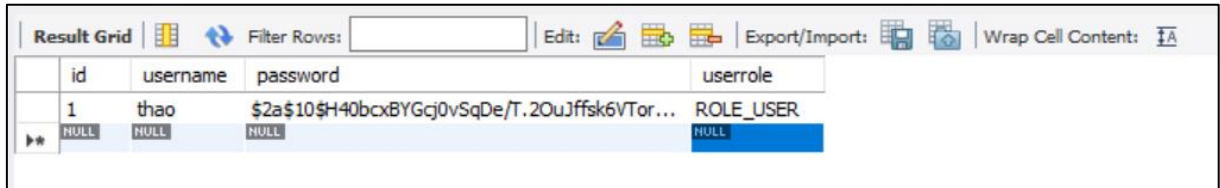
user.login = Đăng nhập
user.username=Tên đăng nhập
user.password=Mật khẩu
user.register=Đăng ký
user.username.error.sizeMsg=Phải nhập tên đăng nhập
user.password.error.sizeMsg=Phải nhập mật khẩu
message.login=Đăng nhập
message.register=Đăng ký

```

message.logout=Đăng xuất

user.login.error1=Username hoặc password không hợp lệ

Chạy thử ứng dụng và đăng ký tạo user, ta sẽ có user như sau dưới cơ sở dữ liệu:



	id	username	password	userrole
	1	thao	\$2a\$10\$H40bcx8YGcj0vSqDe/T.2OuJffsk6VTor...	ROLE_USER
**	NULL	NULL	NULL	NULL

3.11. Bean Validation

Bean Validation được dùng để thiết lập ràng buộc các đối tượng thông qua các annotation. Tại đây, chúng ta có thể chứng thực độ dài, số, biểu thức chính quy,...

Ngoài ra chúng ta cũng có thể tạo ra các chứng thực tùy chỉnh.

3.11.1. Java Bean Validation

Java Bean Validation thực hiện chứng thực dữ liệu dựa trên các annotation mà chúng ta gắn vào các lớp POJO. Ta sử dụng Hibernate Validator để thực hiện chứng thực.

Spring hỗ trợ các annotation dùng để kiểm tra dữ liệu như @Size, @Max, @Min, @NotNull nằm trong gói javax.validation.constraints. Thuộc tính message của mỗi annotation dùng để thông báo lỗi đến người dùng khi vi phạm ràng buộc được cấu hình trong tập tin Properties File.

Ví dụ kiểm tra dữ liệu đầu vào của chức năng thêm môn học.

Bổ sung dependency dưới đây vào tập tin pom.xml.

```
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.2.0.Final</version>
</dependency>
```

Tiếp theo, ta cần cấu hình Bean cho phép kiểm tra dữ liệu, bổ sung cấu hình LocalValidatorFactoryBean vào lớp WebApplicationContextConfig.

@Bean

```
public LocalValidatorFactoryBean validator(){  
  
    LocalValidatorFactoryBean validator = new LocalValidatorFactoryBean();  
  
    validator.setValidationMessageSource(messageSource());  
  
    return validator;  
}
```

@Override

```
public Validator getValidator() {  
  
    return validator();  
}
```

Trong lớp Subject.java bổ sung các annotation để kiểm tra dữ liệu cho các thuộc tính.

```
public class Subject {  
  
    @Min(value = 2, message = "{subject.id.minErr}")  
  
    private String id;  
  
    @Size(min = 2, max = 100, message = "{subject.name.lengErr}")  
  
    private String name;  
  
    private MultipartFile img;  
  
    .....  
}
```

Thêm các khai báo sau vào tập tin message.properties để hiển thị nội dung lỗi.

subject.id.minErr = Mã môn học tối thiểu là 2

subject.name.lengErr = Tên môn học tối thiểu là 2, tối đa là 100.

Trong phương thức xử lý addSubject() của lớp Controller, thêm annotation @Valid để controller biết cần kiểm tra dữ liệu khi request và lưu kết quả sau khi kiểm tra bằng đối tượng BindingResult.

```
@RequestMapping(value = "/", method = RequestMethod.POST)

public String addSubject(Model model, @ModelAttribute(value = "subject")

    @Valid Subject s, BindingResult valid, HttpServletRequest request){

    if(valid.hasErrors())

        return "demo";

    else{

        .....

    }

}
```

Để hiển thị thông tin lỗi ra view ta sử dụng thẻ <form:errors>, trong đó thuộc tính path chỉ định tên thuộc tính của đối tượng khi dữ liệu của nó vi phạm ràng buộc. Ngoài ra, để hiển thị lỗi ở tất cả các trường ta chỉ định giá trị * cho thuộc tính path.

```
<form:form method="POST" modelAttribute="subject" enctype="multipart/form-
data">

    <table>

        <td><spring:message code="subject.id" /></td>

        <td><form:input path="id" /></td>

        <td><form:errors path="id" cssClass="text-danger"

                                element="div"/></td>

    <tr>

        <td><spring:message code="subject.name" /></td>

        <td><form:input path="name" /></td>

        <td><form:errors path="name" cssClass="text-danger"
```

`element="div"/></td>`

`</tr>`

.....

`<form:form>`

Truy cập đường dẫn <http://localhost:8080/SpringMVCDemo/> , nhập các thông tin vi phạm ràng buộc và bấm submit ta có kết quả hiển thị thông báo lỗi dưới đây.

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/SpringMVCDemo/'. The page title is 'Thêm môn học'. The form contains three input fields: 'Mã môn học (Subject ID)', 'Tên môn học (Subject Name)', and 'Ảnh minh họa'. The 'Mã môn học' field has a red error message 'Mã môn học tối thiểu là 2'. The 'Tên môn học' field has a red error message 'Tên môn học tối thiểu là 2, tối đa là 100.'. The 'Ảnh minh họa' field has a 'Choose File' button and the text 'No file chosen'. A blue 'Thêm môn học' button is at the bottom left.

3.11.2. Spring Validation

Spring Validation là một cơ chế chứng thực do Spring cung cấp, nó linh hoạt và dễ mở rộng hơn Bean Validation.

Để tự tạo ra một validation ta cần phải hiện thực giao diện Validator. Trong đó có hai phương thức trừu tượng quan trọng:

- supports(): cho validator biết có được phép kiểm tra một lớp chỉ định hay không.
- validate(): là phương thức dùng để chỉ định những điều kiện kiểm tra dữ liệu.

Ví dụ minh họa dùng Spring Validation kiểm tra dữ liệu tên môn học (name) phải chứa từ “Môn”.

Tạo gói com.ntpt.validator và lớp SubjectNameValidator.java nằm trong gói đó.

@Component

```
public class SubjectNameValidator implements Validator{
```

@Override

```
    public boolean supports(Class<?> type) {
```



```

        return Subject.class.isAssignableFrom(type);
    }

    @Override
    public void validate(Object o, Errors errors) {

        Subject s = (Subject) o;

        if(!s.getName().contains("Môn"))

            errors.rejectValue("name", "subject.name.Err");

    }
}

```

Bổ sung thông báo lỗi trong tập tin message.properties

```
subject.name.Err = Tên môn học phải chứa từ "Môn"
```

Trong lớp `WebApplicationContextConfig` chỉ định gói vừa tạo.

```

@ComponentScan(basePackages = {

    "com.ntpt.controllers",

    "com.ntpt.validator"

})

```

Trong lớp `Controller` khai báo thuộc tính tham chiếu đến đối tượng `SubjectNameValidator` và phương thức `@InitBinder` như dưới đây:

```

@Autowired

private SubjectNameValidator subjectNameValidator;

@InitBinder

public void initBinder(WebDataBinder binder){

    binder.setValidator(subjectNameValidator);

}

```

Chạy thử chương trình và nhập thông tin vi phạm ràng buộc ta có kết quả sau:

Thêm môn học

Mã môn học (Subject ID)

Tên môn học (Subject Name) Tên môn học phải chứa từ "Môn"

Ảnh minh họa No file chosen

Ta thấy, những validator đã được thiết lập trước đó thông qua Bean Validation sẽ không còn tác dụng khi sử dụng Spring Validation. Để có thể sử dụng kết hợp Bean Validation và Spring Validation ta làm như sau:

Tạo lớp `WebAppValidator.java` trong gói `com.ntpt.validator`

```
@Component

public class WebAppValidator implements Validator{

    @Autowired

    private javax.validation.Validator beanValidator;

    private Set<Validator> springValidators;

    @Override

    public boolean supports(Class<?> type) {

        return Subject.class.isAssignableFrom(type);

    }

    @Override

    public void validate(Object o, Errors errors) {

        Set<ConstraintViolation<Object>> beans = this.beanValidator.validate(o);

        for(ConstraintViolation<Object> obj: beans)

            errors.rejectValue(obj.getPropertyPath().toString(), obj.getMessageTemplate(),

                                obj.getMessage());

    }

}
```

```

        for(Validator v: this.springValidators)

            v.validate(o, errors);

    }

    public void setSpringValidators(Set<Validator> springValidators) {

        this.springValidators = springValidators;

    }

}

```

Trong lớp `WebApplicationContextConfig` ta cấu hình Bean cho lớp `WebAppValidator`.

```

@Bean

public WebAppValidator subjectValidator(){

    Set<Validator> springValidators = new HashSet<>();

    springValidators.add(new SubjectNameValidator());


    WebAppValidator validator = new WebAppValidator();

    validator.setSpringValidators(springValidators);


    return validator;

}

```

Trong lớp `Controller`, ta thiết lập lại thuộc tính cho `WebDataBinder` như sau:

```

@Autowired

private WebAppValidator subjectValidator;

@InitBinder

public void initBinder(WebDataBinder binder){

```

```
binder.setValidator(subjectValidator);  
  
}
```

Chạy lại chương trình, ta có kết quả kết hợp giữa Bean Validation và Spring Validation.

Thêm môn học

localhost:8080/SpringMVCDemo/

Mã môn học (Subject ID)

Tên môn học (Subject Name)

Ảnh minh họa No file chosen

Mã môn học tối thiểu là 2

Tên môn học tối thiểu là 2, tối đa là 100.

Tên môn học phải chứa từ "Môn"

3.12. Rest API

REST(REpresentational State Transfer) được Roy Fielding trình bày lần đầu tiên vào năm 2000 trong luận văn của mình. REST có các nguyên tắc và ràng buộc riêng của nó. Các nguyên tắc này phải thỏa mãn RESTful Web Service.

REST quy định cách sử dụng các phương thức HTTP như GET, POST, PUT, DELETE,...và cách định dạng các URL cho ứng dụng web để quản lý các resource. Các ứng dụng web dựa trên REST sẽ trả dữ liệu về theo hai kiểu định dạng chính là JSON hoặc XML.

Ví dụ tạo Rest API để lấy danh sách người dùng.

Mở tập tin pom.xml thêm dependency dưới đây:

```
<dependency>  
  
  <groupId>com.fasterxml.jackson.core</groupId>  
  
  <artifactId>jackson-databind</artifactId>  
  
  <version>2.12.2</version>  
  
</dependency>
```

Tạo lớp ApiController trong lớp com.ntpt.controllers, chỉ định annotation @RestController.

```
@RestController
```

```
public class ApiController {
```

```
    @Autowired
```

```
    private UserService userService;
```

```
    @GetMapping("/api/user")
```

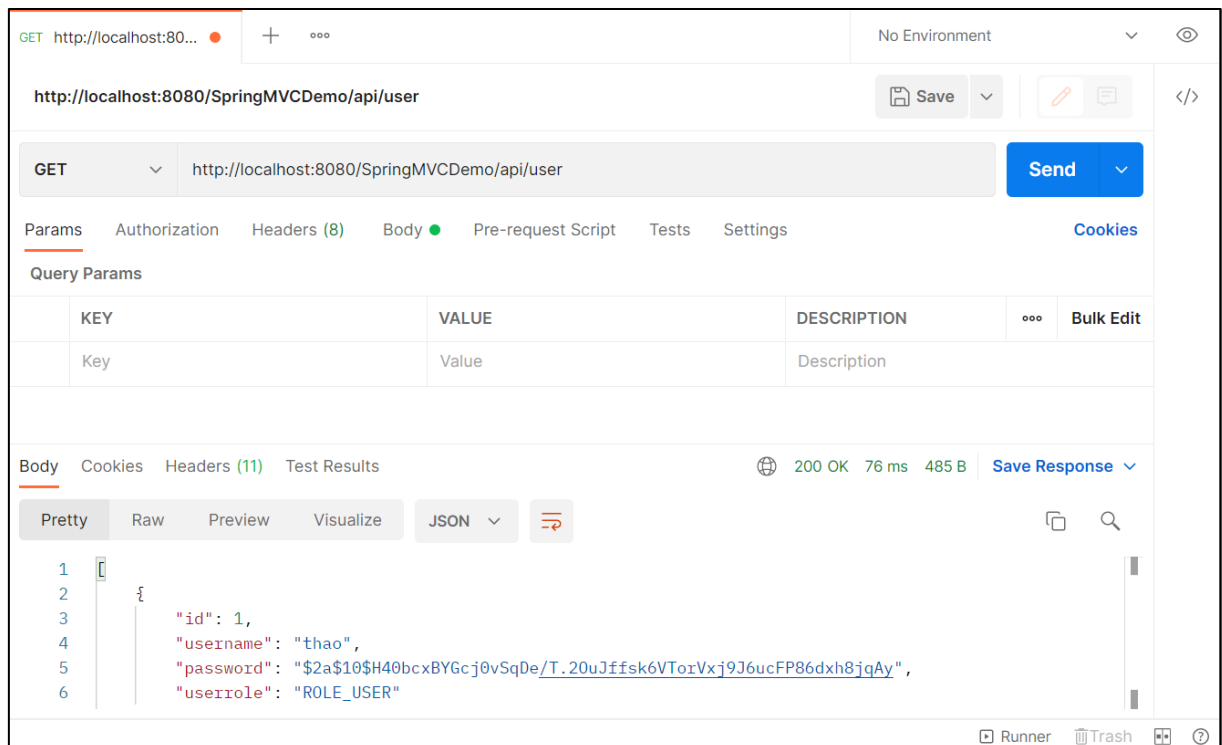
```
    public ResponseEntity<List<User>> getUsers(){
```

```
        return new ResponseEntity<>(this.userService.getUsers(), HttpStatus.OK);
```

```
    }
```

```
}
```

Dùng Postman ta sẽ lấy được danh sách user sau:



Hình 11. Chạy thử ứng dụng với Postman

Chương 4. **HỆ THỐNG ĐẶT DỊCH VỤ DU LỊCH**

Trong chương này trình bày một hệ thống đặt dịch vụ du lịch sử dụng SpringMVC kết hợp với cơ sở dữ liệu quan hệ MySQL thông qua Hibernate.

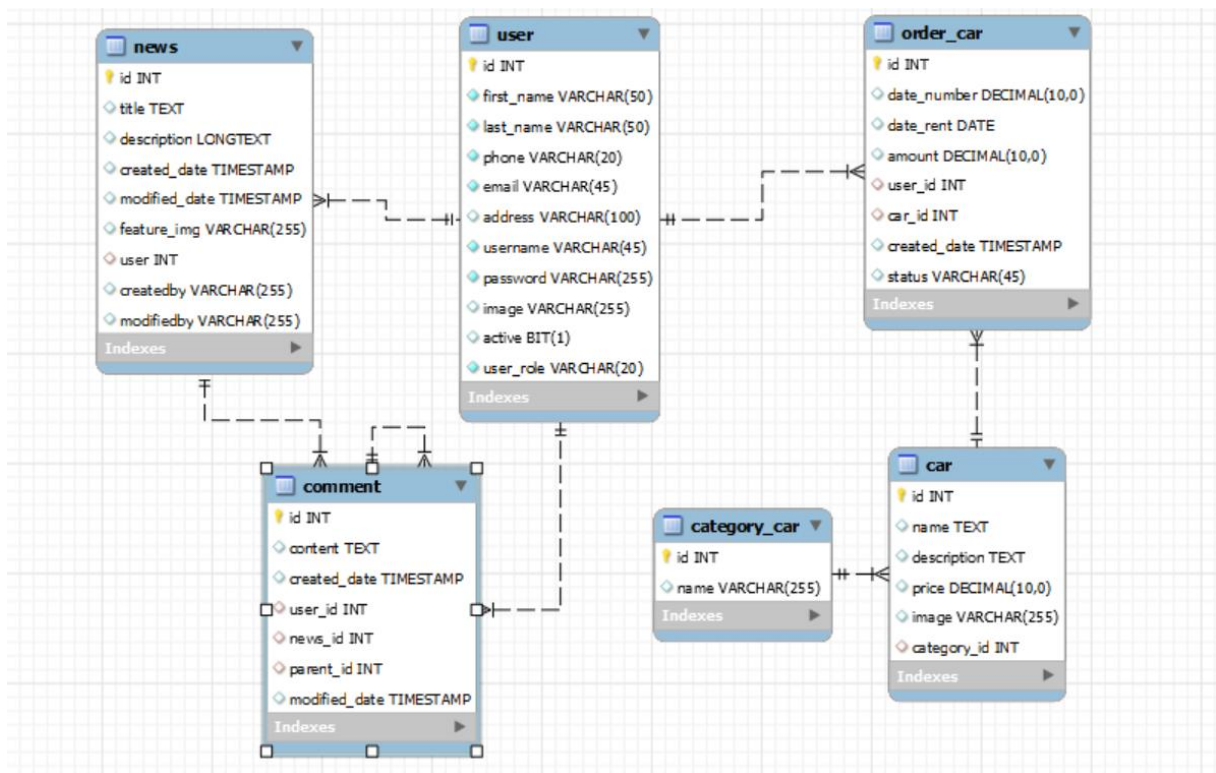
4.1. Mô tả hệ thống

Hệ thống đặt dịch vụ du lịch có ba phân hệ người dùng admin, nhân viên và khách hàng với các chức năng như sau:

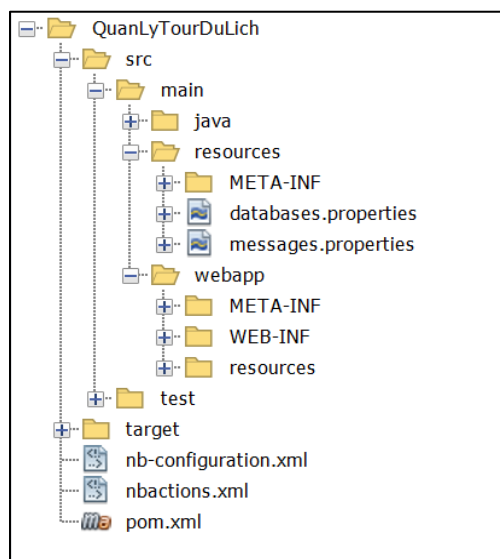
- Đăng nhập(có vai trò admin, nhân viên, khách hàng).
- Đăng ký khách hàng, nhân viên do admin thêm khi cần trong trang quản lý nhân viên.
- Nhân viên được phép thêm, xóa, sửa, tìm kiếm khách hàng, tin tức du lịch, dịch vụ thuê xe du lịch.
- Admin được phép thêm, xóa, sửa, tìm kiếm nhân viên, khách hàng, tin tức du lịch, dịch vụ thuê xe du lịch.
- Khách hàng có thể xem tin tức du lịch(phân trang, mỗi trang tối đa 6 tin tức), 5 tin tức mới nhất được hiển thị ở trang chủ. Trong trang chi tiết tin tức, người dùng có thể bình luận tin tức, trả lời bình luận của người khác và sửa, xóa bình luận của mình.
- Khách hàng có thể xem dịch vụ thuê xe theo số chỗ(7 chỗ, 16 chỗ, 45 chỗ). Muốn đặt xe khách hàng cần phải có tài khoản và đăng nhập.
- Admin có thể xem thống kê, doanh thu.

4.2. Cơ sở dữ liệu

Để xây dựng hệ thống này, trước tiên ta xây dựng cơ sở dữ liệu quan hệ MySQL với tên tourdb và cấu hình các thuộc tính trong tập tin properties để kết nối với hệ thống ứng dụng Web SpringMVC.



Hình 12. Lược đồ cơ sở dữ liệu quan hệ của hệ thống đặt dịch vụ du lịch



Hình 13. Cấu trúc project QuanLyTourDuLich

4.3. Chức năng đăng ký người dùng

Người dùng có thể đăng ký tài khoản cho mình bằng cách truy cập vào trang đăng ký, nhân viên sẽ do admin thêm trong trang quản lý nhân viên. Khi đăng ký, khách hàng phải nhập đầy đủ các thông tin, email và tên đăng nhập của người dùng phải là email và tên đăng nhập chưa được đăng ký trong hệ thống. Nếu người dùng nhập sai hoặc thiếu bất kì thông tin nào sẽ xuất hiện thông báo lỗi. Bên cạnh đó, hệ

thông còn sử dụng Spring Validation để bắt lỗi nếu người dùng nhập mật khẩu và xác nhận mật khẩu khác nhau. Sau khi nhập đầy đủ thông tin, người dùng nhấn vào nút đăng ký, hệ thống sẽ chuyển đến trang đăng nhập cho người dùng.

The image shows a registration form titled "ĐĂNG KÝ" (Register) with a background of a mountain landscape. The form contains the following fields and validation messages:

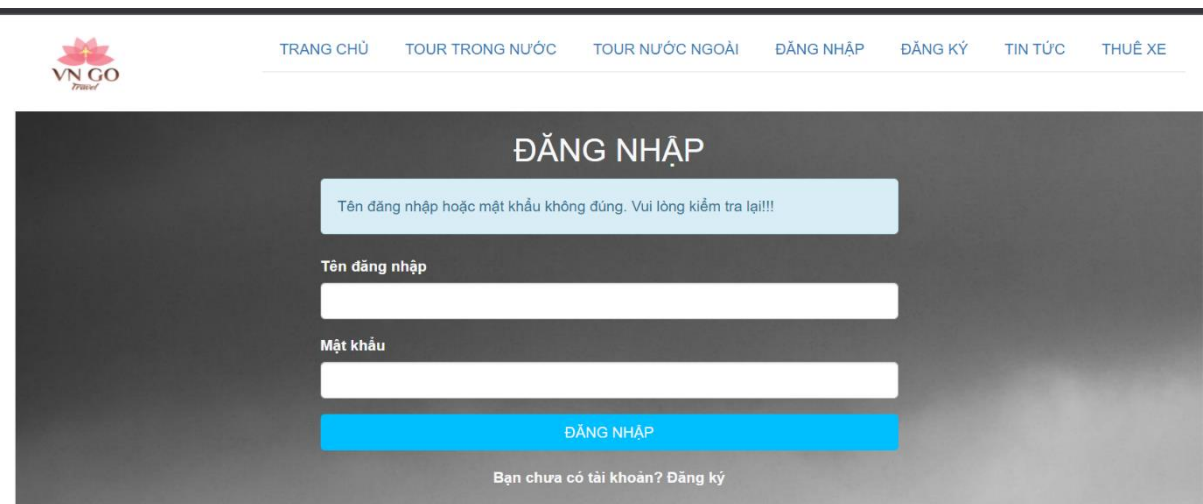
- Tên** (Name): Input field with error message "Phải nhập tên và có tối đa 45 ký tự" (Must enter name and have a maximum of 45 characters).
- Họ** (Surname): Input field with error message "Phải nhập họ và có tối đa 45 ký tự" (Must enter surname and have a maximum of 45 characters).
- Điện thoại** (Phone): Input field with error message "Số điện thoại không hợp lệ" (Invalid phone number).
- Email**: Input field with error message "Email không hợp lệ" (Invalid email).
- Địa chỉ** (Address): Input field with error message "Phải nhập địa chỉ" (Must enter address).
- Tên đăng nhập** (Username): Input field with error message "Phải nhập tên đăng nhập và có tối đa 45 ký tự" (Must enter username and have a maximum of 45 characters).
- Mật khẩu** (Password): Input field with error message "Các mật khẩu nhập không khớp" (Entered passwords do not match).
- Xác nhận mật khẩu** (Confirm Password): Input field.
- Ảnh đại diện** (Profile Picture): File upload field with "Choose File" button and "No file chosen" text.

At the bottom of the form is a blue button labeled "ĐĂNG KÝ" (Register).

Hình 14. Chức năng đăng ký người dùng trong hệ thống

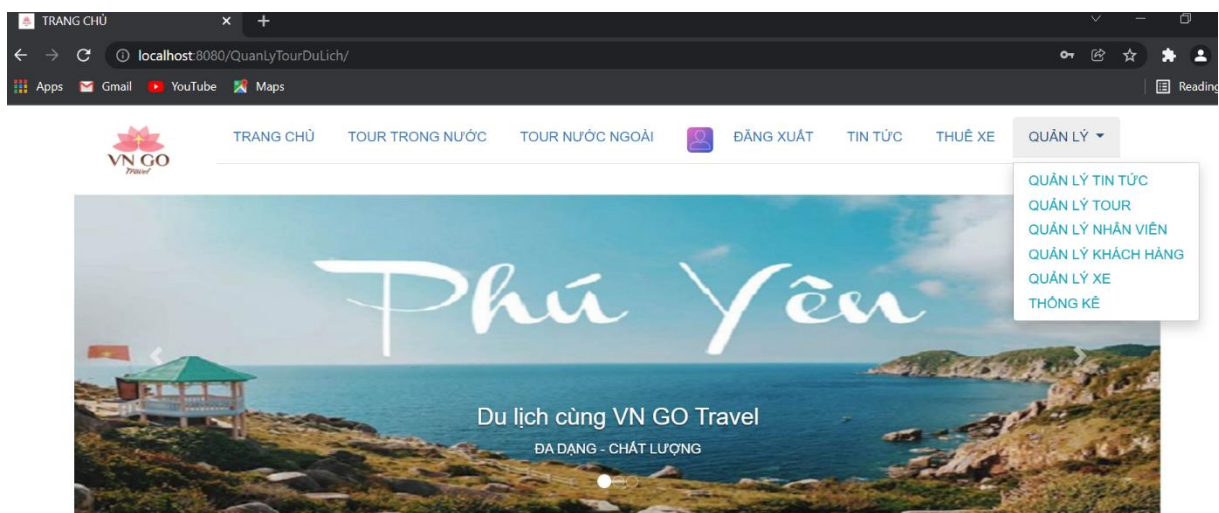
4.4. Chức năng đăng nhập

Để đăng nhập vào hệ thống, người dùng vào trang đăng nhập và điền đúng thông tin tên đăng nhập, mật khẩu đã đăng ký. Nếu nhập sai tên đăng nhập hoặc mật khẩu, trang web sẽ thông báo lỗi cho bạn nhập lại. Trong trường hợp người dùng chưa có tài khoản để đăng nhập, chỉ cần click vào nút đăng ký phía dưới nút đăng nhập, hệ thống sẽ chuyển người dùng đến trang đăng ký. Khi đăng nhập thành công, trang web sẽ chuyển đến trang chủ.



Hình 15. Chức năng đăng nhập của hệ thống tour du lịch

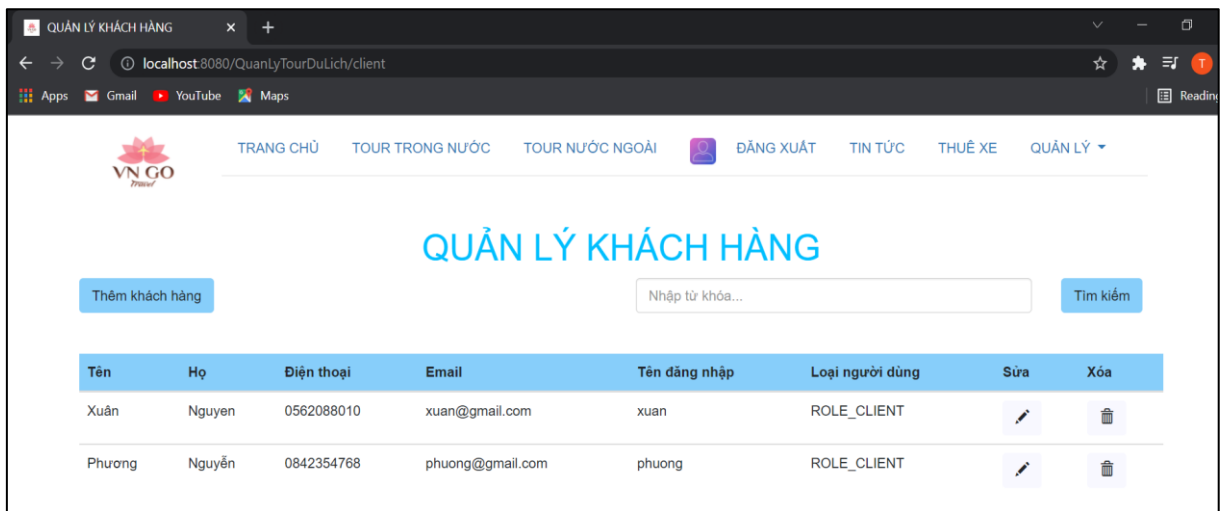
Nếu đăng nhập bằng tài khoản admin hoặc nhân viên, thanh tiêu đề sẽ xuất hiện chức năng quản lý để người dùng có thể quản lý hệ thống.



Hình 16. Giao diện quản lý của người dùng

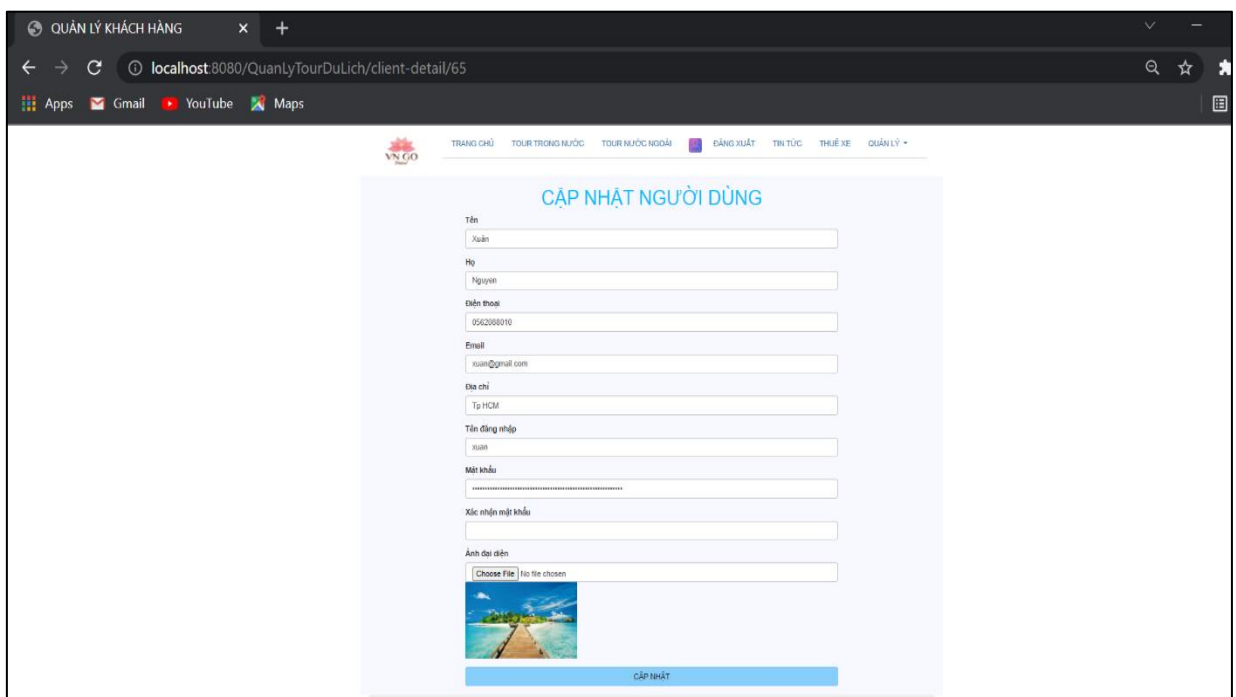
4.5. Chức năng quản lý khách hàng

Admin và nhân viên có thể quản lý khách hàng trong trang quản lý. Hệ thống sẽ sử dụng Spring Security để kiểm tra loại người dùng khi truy cập vào trang này. Trong trang quản lý khách hàng, người dùng có thể tìm kiếm khách hàng theo tên, họ, số điện thoại, email và tên đăng nhập của khách hàng.



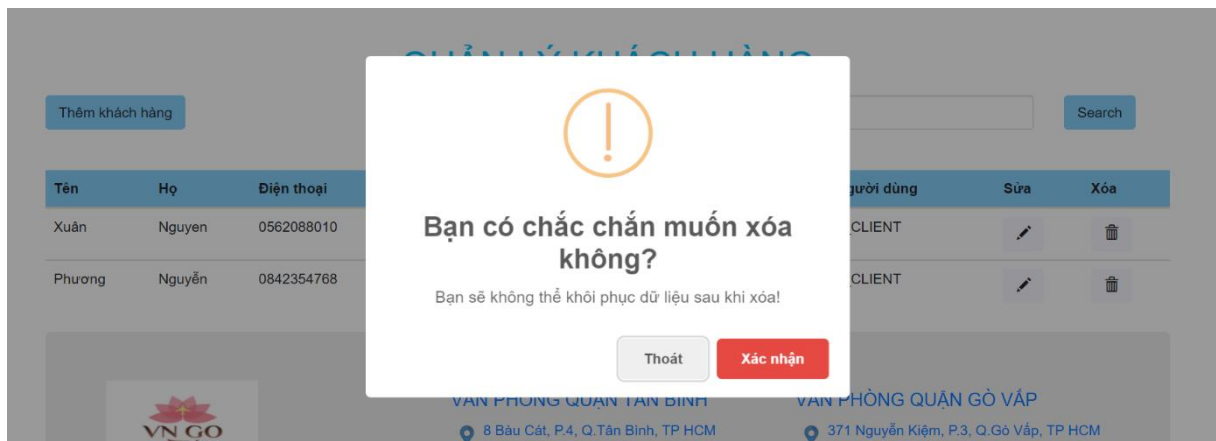
Hình 17. Chức năng quản lý khách hàng của hệ thống

Tại đây, người dùng có thể thêm, xóa, sửa khách hàng. Khi thêm hoặc sửa khách hàng, hệ thống sẽ chuyển sang trang cập nhật người dùng để bạn có thể thực hiện chức năng.



Hình 18. Chức năng cập nhật người dùng của hệ thống

Khi thực hiện chức năng xóa, hệ thống sẽ xuất hiện thông báo xác nhận người dùng có muốn xóa hay không. Khi ấn nút xác nhận, hệ thống sẽ xóa người dùng bằng cách gọi hàm Ajax.



Hình 19. Chức năng xóa người dùng

4.6. Chức năng quản lý nhân viên

Người dùng đăng nhập bằng tài khoản admin mới có thể vào trang quản lý nhân viên và thực hiện các chức năng tìm kiếm, thêm, sửa, xóa nhân viên.

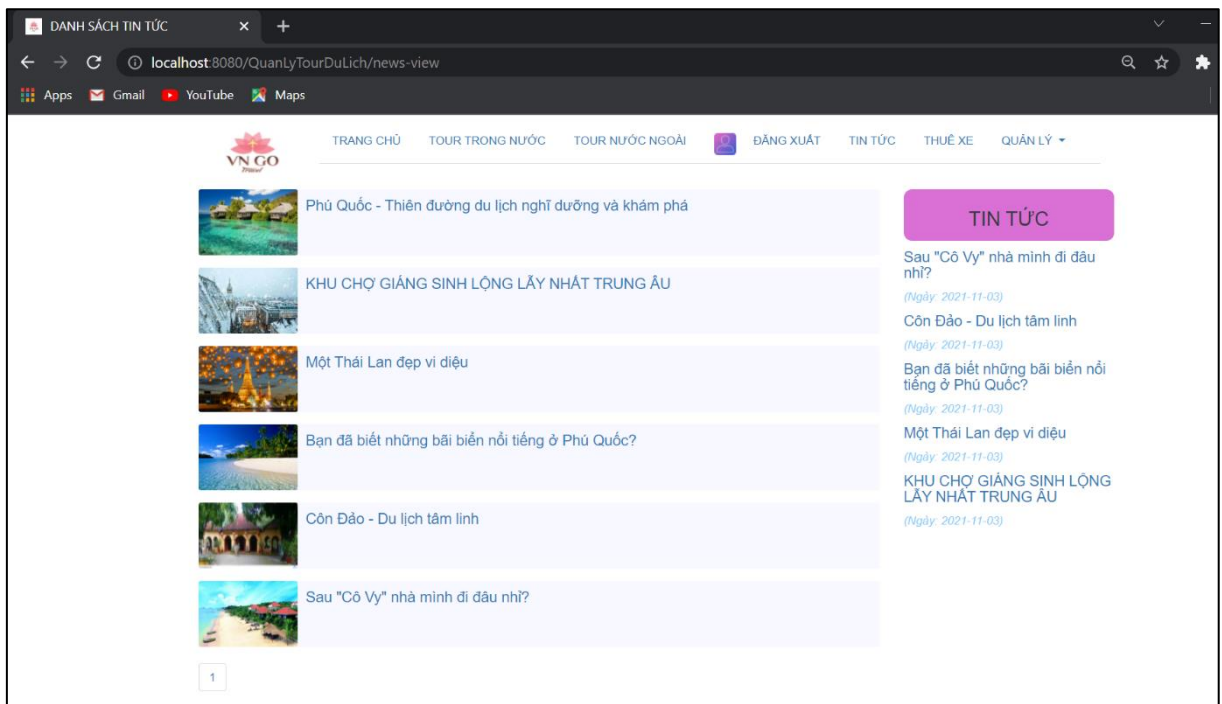
QUẢN LÝ NHÂN VIÊN							
Thêm nhân viên		Nhập từ khóa...				Tìm kiếm	
Tên	Họ	Điện thoại	Email	Tên đăng nhập	Loại người dùng	Sửa	Xóa
Thảo	Nguyen	0562088010	f@x	ư	ROLE_STAFF		
a	a	0842354768	a@a	a	ROLE_STAFF		
ccccc	c	0562088010	c@c	c	ROLE_STAFF		
Lan	Nguyễn Xuân	0842354768	lan@gmail.com	lan	ROLE_STAFF		
Mai	Nguyễn	0842354768	mai@gmail.com	mai	ROLE_STAFF		

Hình 20. Chức năng quản lý nhân viên của hệ thống

Nếu người dùng chưa đăng nhập hoặc đăng nhập bằng tài khoản nhân viên hay khách hàng mà truy cập vào trang này, hệ thống sẽ tự động chuyển về trang đăng nhập bằng cách sử dụng Spring Security.

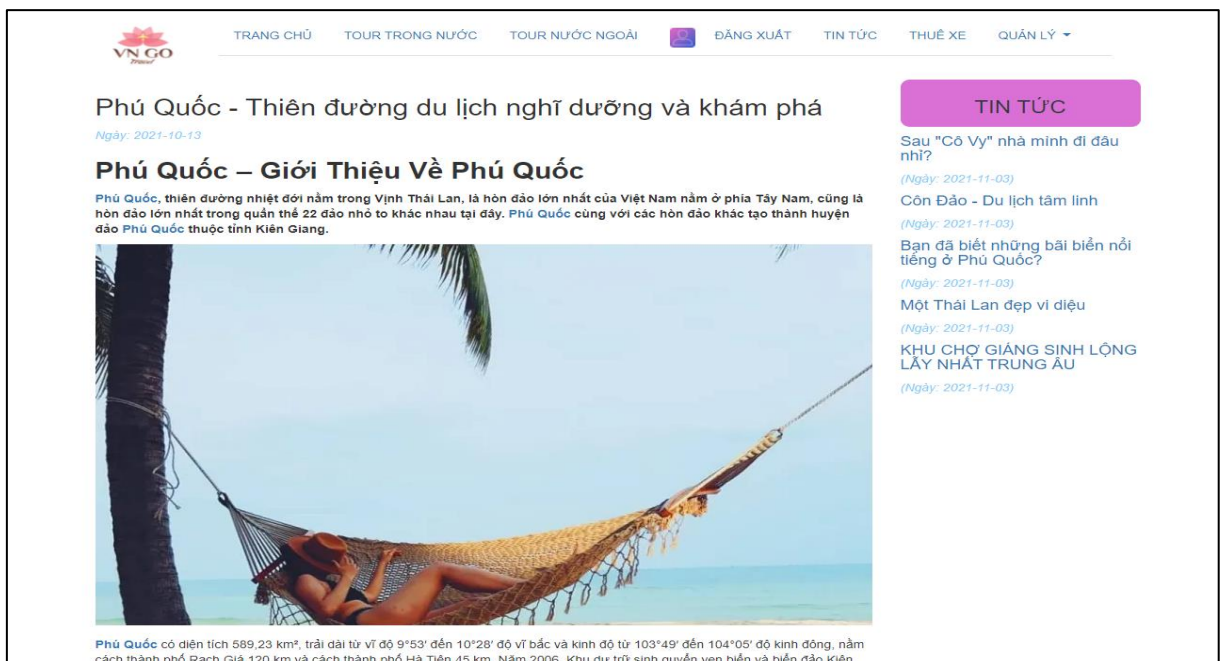
4.7. Quản lý tin tức du lịch

Khách hàng có thể xem tin tức bằng cách nhấn vào mục tin tức trên thanh tiêu đề, sau khi nhấn vào mục tin tức, hệ thống sẽ chuyển sang trang danh sách tin tức. Trong trang này, hệ thống thực hiện phân trang, mỗi trang tối đa 6 tin tức. Bên góc phải trang chủ và trang danh sách tin tức hiển thị 5 tin tức mới nhất.



Hình 21. Giao diện xem tin tức của hệ thống

Muốn coi chi tiết tin tức, bạn chỉ cần nhấp vào tiêu đề của từng tin tức.



Hình 22. Giao diện xem chi tiết tin tức của hệ thống

Trong trang chi tiết tin tức, người dùng có thể gửi bình luận, phản hồi bình luận của người khác. Hệ thống có sử dụng thêm thư viện Moment.js để hiển thị thời gian của bình luận. Bên cạnh đó, người dùng còn có thể sửa hay xóa bình luận của mình.

những nơi lý tưởng cho các hoạt động khám phá thiên nhiên cùng các hoạt động trên biển như du thuyền, câu cá, lặn ngắm san hô và khám phá đảo hoang kỳ thú...

Nhập bình luận của bạn...

Gửi

 **Xuân**
Dịch vụ tốt!!!
một tháng trước
[Phản hồi](#)

 **Xuân**
reply
một tháng trước
[Phản hồi](#)

Nhập bình luận của bạn...

Gửi

Hình 23. Chức năng bình luận bài viết tin tức

Người dùng đăng nhập bằng tài khoản admin và nhân viên có thể thêm, sửa, xóa, tìm kiếm tin tức dựa vào tiêu đề trong trang quản lý tin tức.



[TRANG CHỦ](#)
[TOUR TRONG NƯỚC](#)
[TOUR NƯỚC NGOÀI](#)
[ĐĂNG XUẤT](#)
[TIN TỨC](#)
[THUÊ XE](#)
[QUẢN LÝ](#)

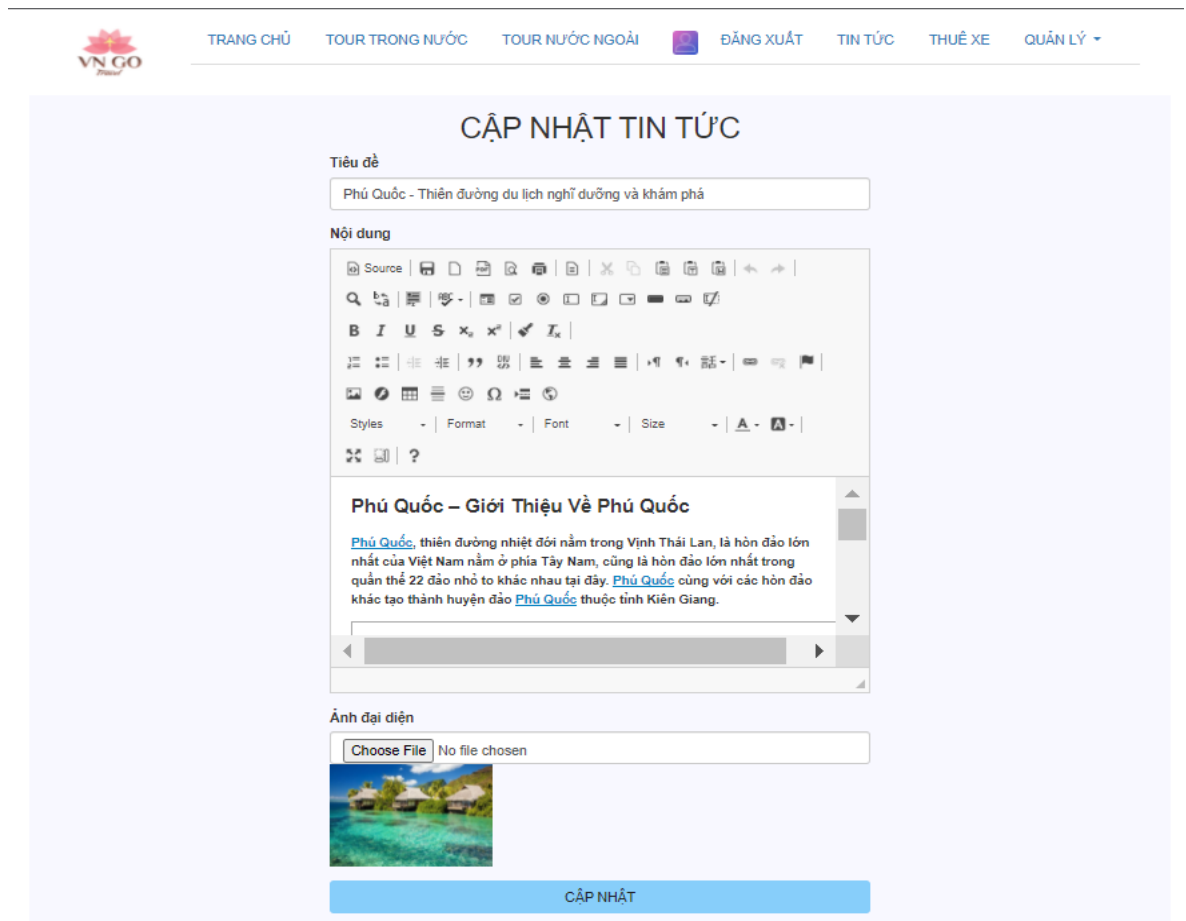
QUẢN LÝ TIN TỨC

[Thêm tin tức](#)

[Tìm kiếm](#)

Tiêu đề	Ngày tạo	Người tạo	Sửa	Xóa
Phú Quốc - Thiên đường du lịch nghỉ dưỡng và khám phá	2021-10-13	thao		
KHU CHỢ GIẢNG SINH LỘNG LẦY NHẬT TRUNG ÂU	2021-11-03	thao		
Một Thái Lan đẹp vì điệu	2021-11-03	thao		
Bạn đã biết những bãi biển nổi tiếng ở Phú Quốc?	2021-11-03	thao		

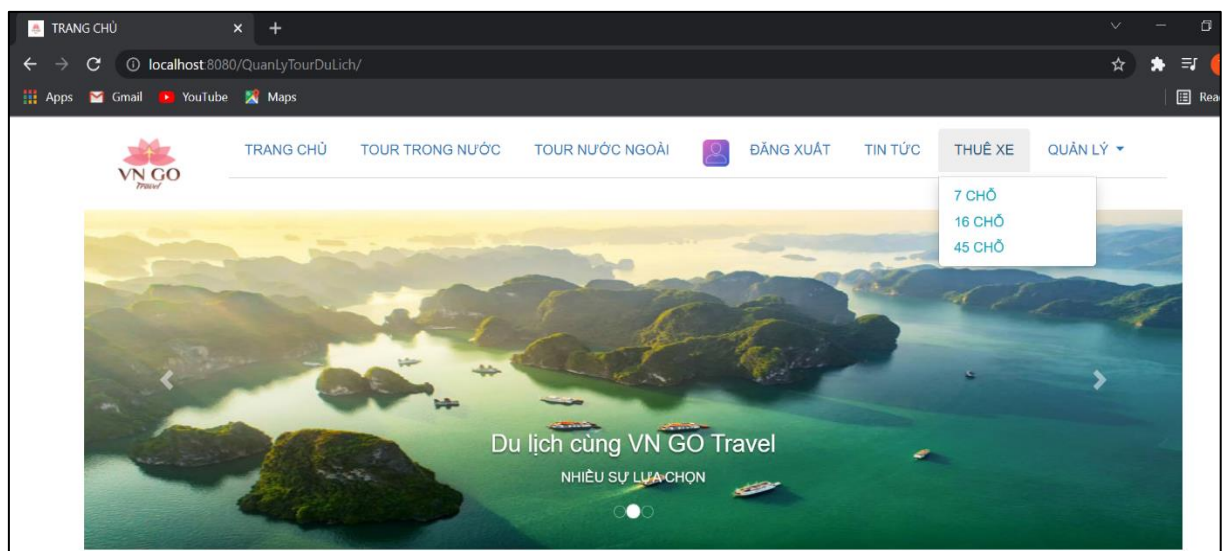
Hình 24. Chức năng quản lý tin tức của hệ thống



Hình 25. Chức năng sửa tin tức trong hệ thống

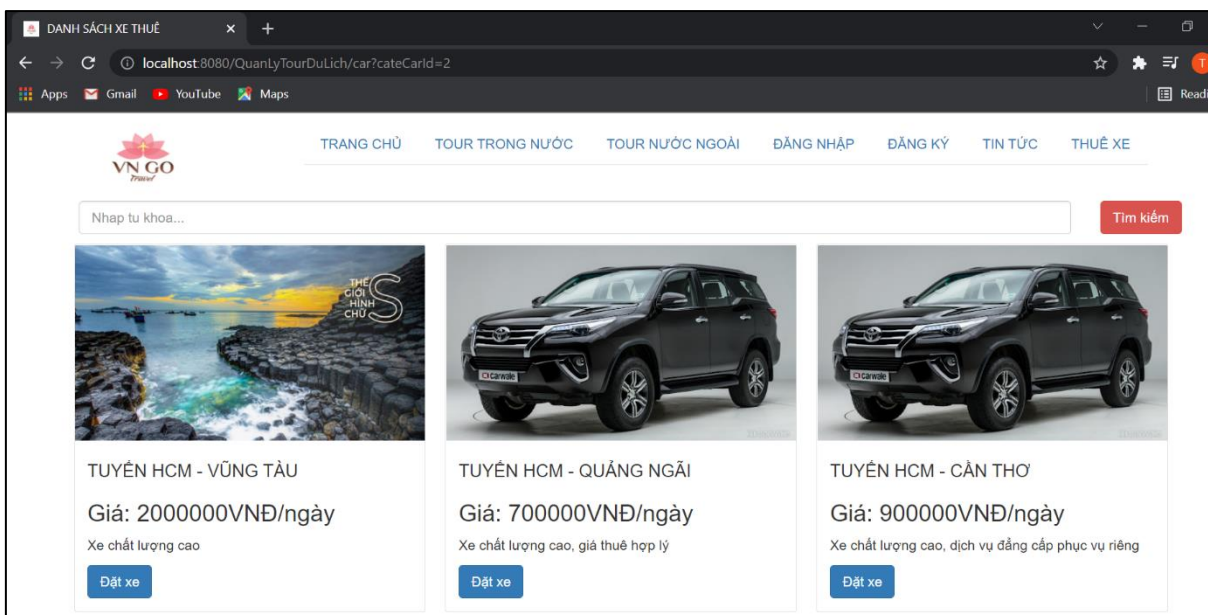
4.8. Chức năng quản lý thuê xe

Có 3 loại xe để người dùng có thể thuê (7 chỗ, 16 chỗ, 45 chỗ), khi nhấn vào từng loại xe, hệ thống sẽ chuyển đến trang hiển thị các xe thuê của loại đó.



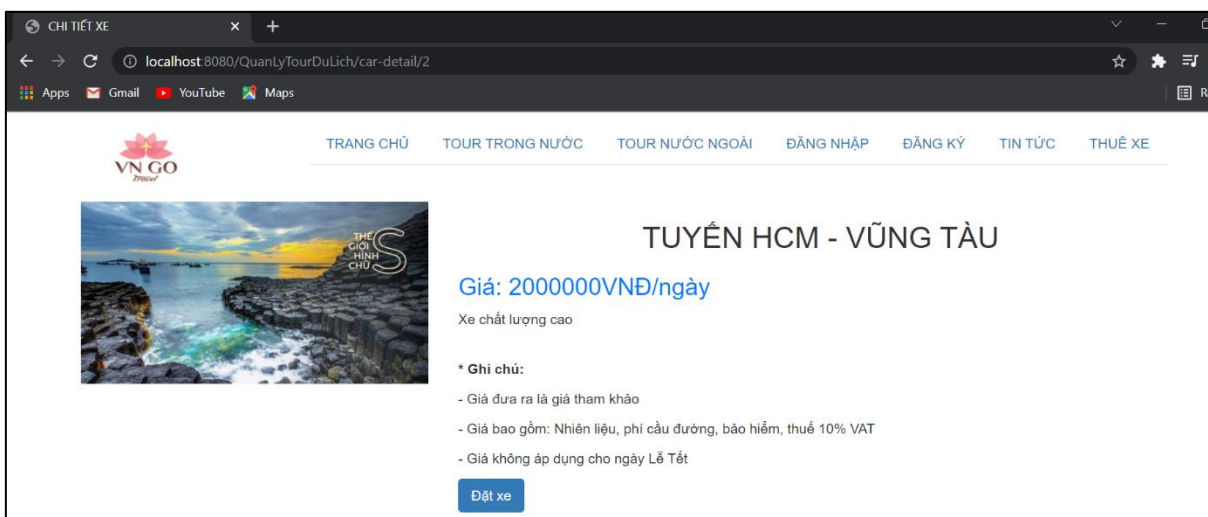
Hình 26. Giao diện hiển thị các loại xe cho thuê

Tại trang danh sách xe thuê, người dùng có thể tìm kiếm xe theo tên và giá.



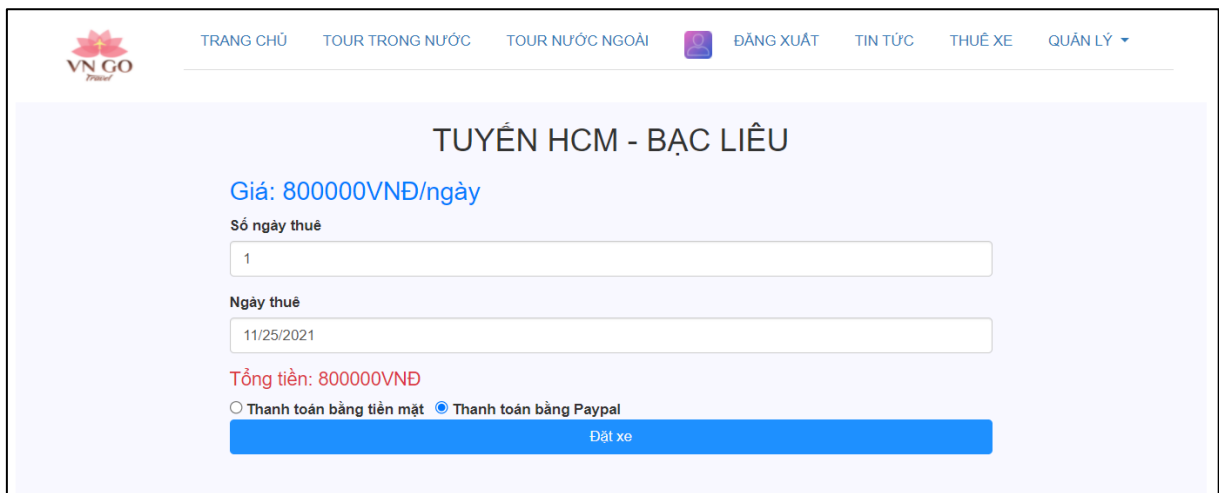
Hình 27. Giao diện hiển thị danh sách xe cho thuê

Để xem chi tiết xe, người dùng chỉ cần nhấn vào từng xe, hệ thống sẽ chuyển sang trang chi tiết xe. Người dùng cần phải đăng nhập để có thể đặt xe, nếu chưa đăng nhập mà nhấn nút đặt xe, hệ thống sẽ chuyển người dùng về trang đăng nhập.



Hình 28. Giao diện mô tả chi tiết xe cho thuê

Tại trang thuê xe, người dùng cần nhập thông tin số ngày thuê và ngày thuê để tiến hành đặt xe. Khi người dùng nhập số ngày thuê, hệ thống sẽ tự động cập nhật tổng tiền cần thanh toán để hiển thị cho người dùng. Sau đó, người dùng chọn phương thức thanh toán và tiến hành đặt xe.



TUYẾN HCM - BẠC LIÊU

Giá: 800000VNĐ/ngày

Số ngày thuê:

Ngày thuê:

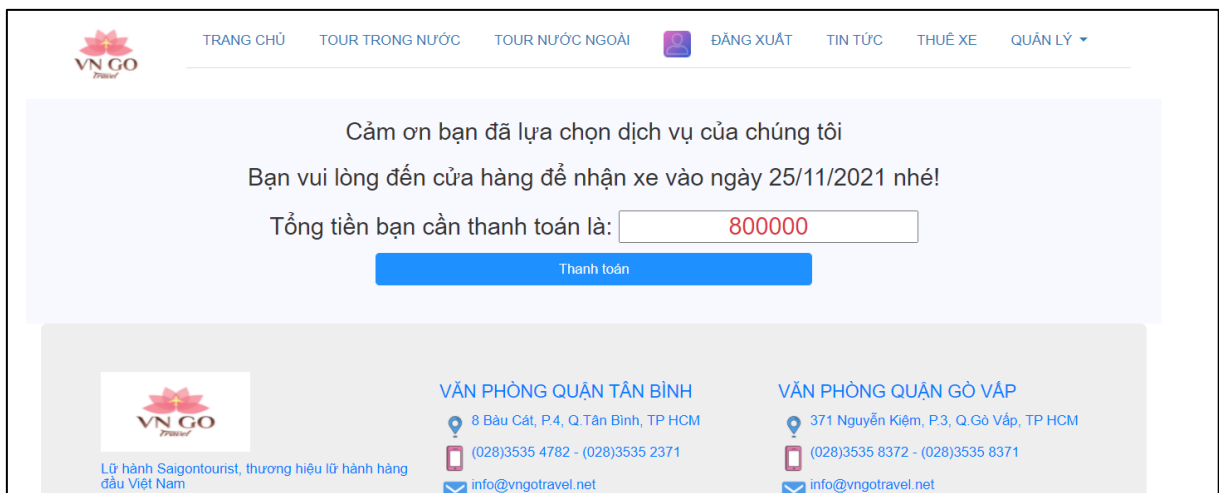
Tổng tiền: 800000VNĐ

☐ Thanh toán bằng tiền mặt
 ☒ Thanh toán bằng Paypal

[Đặt xe](#)

Hình 29. Chức năng đặt xe trong hệ thống

Khi nhấn vào nút đặt xe, hệ thống sẽ dùng Rest API gọi hàm ajax và hiển thị số tiền cần thanh toán.



Cảm ơn bạn đã lựa chọn dịch vụ của chúng tôi

Bạn vui lòng đến cửa hàng để nhận xe vào ngày 25/11/2021 nhé!

Tổng tiền bạn cần thanh toán là:

[Thanh toán](#)

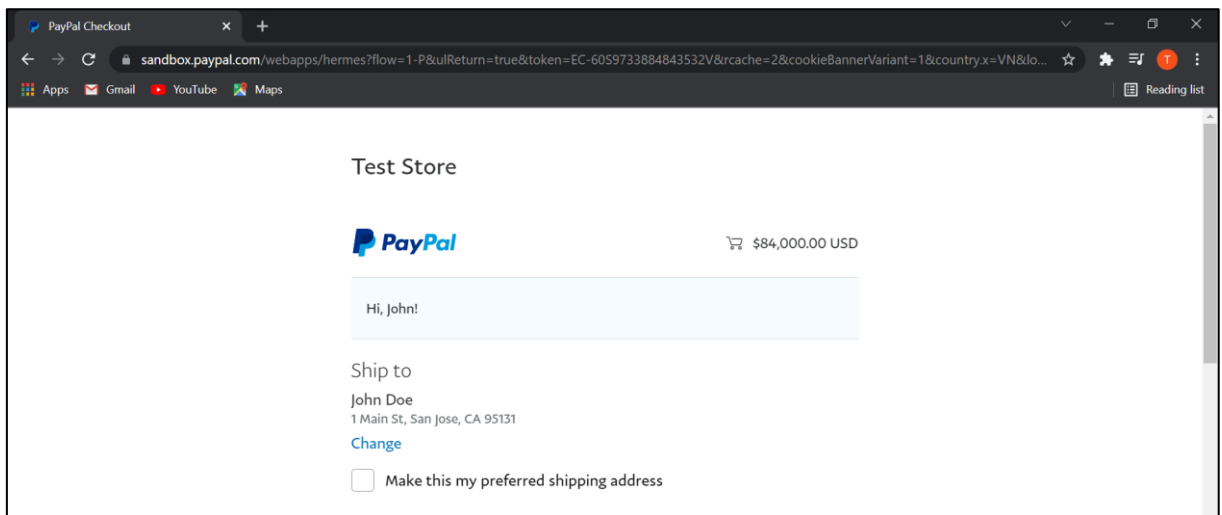
VN GO Travel
Lữ hành Saigontourist, thương hiệu lữ hành hàng đầu Việt Nam

VĂN PHÒNG QUẬN TÂN BÌNH
8 Bàu Cát, P.4, Q. Tân Bình, TP HCM
(028)3535 4782 - (028)3535 2371
info@vngotravel.net

VĂN PHÒNG QUẬN GÒ VẤP
371 Nguyễn Kiệm, P.3, Q. Gò Vấp, TP HCM
(028)3535 8372 - (028)3535 8371
info@vngotravel.net

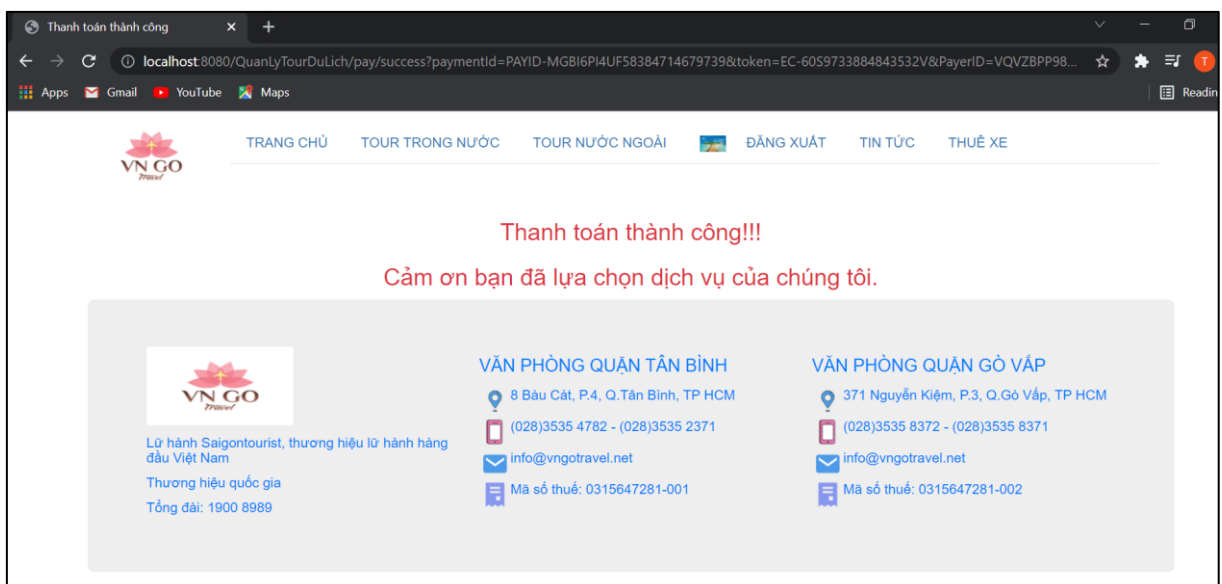
Hình 30. Rest API gọi hàm ajax hiển thị thông tin đặt xe

Nếu người dùng chọn phương thức thanh toán bằng Paypal, khi nhấn vào nút thanh toán, hệ thống sẽ chuyển đến trang Paypal để bạn có thể thanh toán trực tuyến.



Hình 31. Chức năng thanh toán với Paypal

Sau khi thanh toán thành công, hệ thống sẽ chuyển người dùng về trang thanh toán thành công và thông báo người dùng đã đặt xe xong.



Hình 32. Giao diện thông báo đặt xe thành công

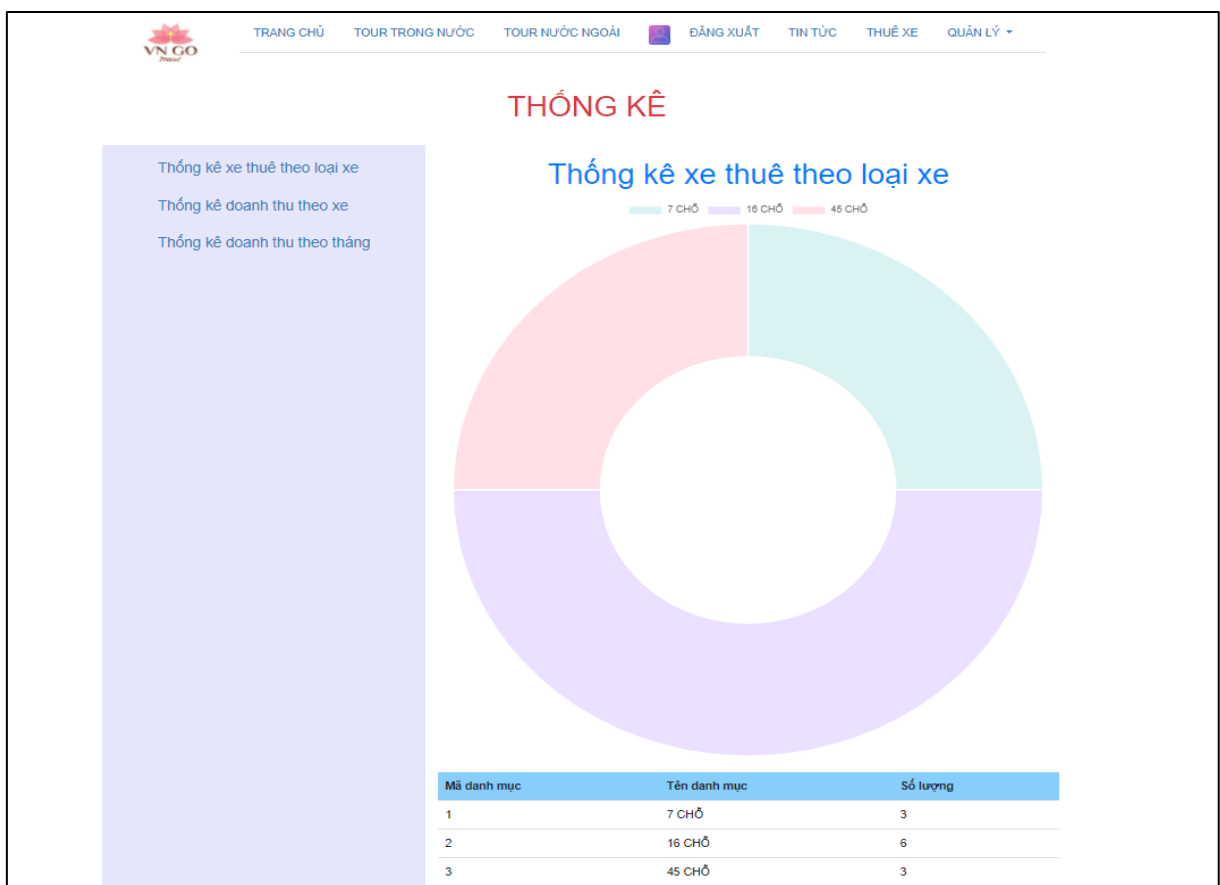
Để quản lý xe, người dùng cần đăng nhập bằng tài khoản admin hoặc nhân viên. Tại trang quản lý xe, người dùng có thể tìm kiếm theo tên, giá, xem thông tin xe theo loại, thực hiện thêm, xóa, sửa các xe và phân trang hiển thị tối đa 6 xe.

QUẢN LÝ XE					
Thêm xe		Nhập từ khóa...		Tìm kiếm	
				Loại xe ▾	
STT	Tuyến xe	Giá xe	Loại xe	Sửa	Xóa
12	TUYẾN HCM - HÀ NỘI	4000000	45 CHỖ		
11	TUYẾN HCM - HUẾ	700000	16 CHỖ		
10	TUYẾN HCM - ĐÀ NẴNG	700000	16 CHỖ		
9	TUYẾN HCM - BÀC LIÊU	800000	16 CHỖ		
8	TUYẾN HCM - ĐẮK LẮK	1000000	45 CHỖ		
7	TUYẾN HCM - CẦN THƠ	900000	16 CHỖ		

Hình 33. Chức năng quản lý xe trong hệ thống

4.9. Quản lý thống kê

Người dùng đăng nhập bằng tài khoản admin có thể xem thống kê xe thuê theo loại xe, thống kê doanh thu theo xe và thống kê doanh thu theo tháng.



Hình 34. Chức năng quản lý thống kê của hệ thống

Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chương này dùng để trình bày các kết quả đạt được trong quá trình thực hiện đồ án ngành. Đồng thời nêu ra những định hướng để phát triển thêm hệ thống.

5.1. Kết luận

Nhờ sự giúp đỡ của các thầy cô trong khoa, đặc biệt là sự hướng dẫn nhiệt tình của thầy Th.S Dương Hữu Thành, cũng như quá trình tìm hiểu thêm kiến thức từ nhiều nguồn thông tin khác nhau, em đã đạt được các kết quả sau:

- Nắm bắt được lý thuyết cơ bản về Spring Framework đặc biệt là Spring MVC.
- Vận dụng được các đặc trưng, tính năng cơ bản của Spring MVC.
- Xây dựng được một hệ thống tour du lịch với những chức năng cơ bản của Spring MVC kết hợp với hệ thống cơ sở dữ liệu MySQL thông qua Hibernate.

5.2. Hướng phát triển

Từ hệ thống tour du lịch cơ bản trên, chúng ta có thể phát triển hệ thống để hoàn chỉnh hơn nữa với nhiều chức năng hơn như:

- Phát triển tích hợp dịch vụ bán vé máy bay.
- Phát triển tích hợp dịch vụ bán bảo hiểm du lịch.
- Thực hiện chức năng load dữ liệu khi người dùng có nhu cầu kéo đến để xem nhằm cải thiện tốc độ của hệ thống.
- Tích hợp thêm nhiều công thanh toán phù hợp với nhu cầu thanh toán của người dùng.

TÀI LIỆU THAM KHẢO

- [1] Dương Hữu Thành. *Lập trình Java*. NXB Thông tin & Truyền thông. 2019.
- [2] Amuthan Ganeshan. *Spring MVC Beginner's Guide*. Packt Publishing Ltd. 2014.
- [3] Spring Tutorial. [Trực tuyến]. Địa chỉ: <https://www.tutorialspoint.com/spring/index.htm> [Truy cập 10/2021]
- [4] Spring Tutorial – Spring Core Framework Tutorials. [Trực tuyến]. Địa chỉ: <https://www.journaldev.com/2888/spring-tutorial-spring-core-tutorial> [Truy cập 10/2021]
- [5] Web MVC framework. [Trực tuyến]. Địa chỉ: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html> [Truy cập 10/2021]
- [6] Front Controller. [Trực tuyến]. Địa chỉ: https://en.wikipedia.org/wiki/Front_controller [Truy cập 10/2021]
- [7] Spring Security. [Trực tuyến]. Địa chỉ: <https://spring.io/projects/spring-security> [Truy cập 10/2021]
- [8] What is REST. [Trực tuyến]. Địa chỉ: <https://restfulapi.net/> [Truy cập 10/2021]
- [9] Java Bean Validation Basics. [Trực tuyến]. Địa chỉ: <https://www.baeldung.com/javax-validation> [Truy cập 10/2021]
- [10] Views và View Resolvers. [Trực tuyến]. Địa chỉ: <https://hocspringmvc.net/bai-doc-views-va-view-resolvers/> [Truy cập 10/2021]