**MỤC LỤC**

[**LỜI CẢM ƠN** 3](#_Toc481879133)

[**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN** 4](#_Toc481879134)

[**CHƯƠNG 1. TỔNG QUAN VỀ CÔNG TY DXC TECHNOLOGY VIỆT NAM** 5](#_Toc481879135)

[**1.1. Quá trình hình thành và phát triển của Công ty** 5](#_Toc481879136)

[**1.2. Đặc điểm hoạt động kinh doanh của công ty DXC Technology Việt Nam** 6](#_Toc481879137)

[**1.3. Đặc điểm tổ chức quản lý của công ty** 7](#_Toc481879138)

[**1.3.1. Đặc điểm phân cấp quản lý** 7](#_Toc481879139)

[**1.3.2. Số lượng nhân sự và chức năng từng phòng ban của DXC Technology Việt Nam** 7](#_Toc481879140)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ LẬP TRÌNH WEB JAVA (BACK-END)** 7](#_Toc481879141)

[**2.1. Tìm hiểu về Java** 7](#_Toc481879142)

[**2.1.1. Giới thiệu về ngôn ngữ lập trình Java** 7](#_Toc481879143)

[Hình 2.1.1a. Quá trình thông dịch của Java 9](#_Toc481879144)

[**2.1.2. Tại sao lại phát triển web bằng ngôn ngữ Java?** 10](#_Toc481879145)

[**2.2. Tìm hiểu về Web** 10](#_Toc481879146)

[**2.2.1. Web là gì?** 10](#_Toc481879147)

[**2.2.2. Tầm quan trọng của Web** 10](#_Toc481879148)

[**2.3. Tìm hiểu AngularJS** 10](#_Toc481879149)

[**2.3.1. AngularJS là gì?** 10](#_Toc481879150)

[**2.3.2. AngularJS hỗ trợ cho việc phát triển web Java như thế nào ?** 10](#_Toc481879151)

[**2.4. Lập trình Web với Java** 10](#_Toc481879152)

[**2.4.1. Tổng quan về J2EE** 10](#_Toc481879153)

[**2.4.1.1. Giới thiệu vê Servlet** 10](#_Toc481879154)

[**2.4.1.2. Giới thiệu về JSP** 10](#_Toc481879155)

[**2.4.1.3. JSTL là gì ?** 10](#_Toc481879156)

[**2.4.1.4. JPQL là gì ?** 10](#_Toc481879157)

[**2.4.1.5. Mô hình MVC với JSP/Servlet** 10](#_Toc481879158)

[**2.4.2. Tổng quan về EJB 2.0, JDBC** 10](#_Toc481879159)

[**2.4.3. Tổng quan về Java Persistence API, Hibernate** 10](#_Toc481879160)

[**2.4.4. Spring MVC** 11](#_Toc481879161)

[**CHƯƠNG 3. PHÂN TÍCH WEB TẠI CÔNG TY DXC TECHNOLOGY** 11](#_Toc481879162)

[**3.1. Quá trình hiện thực Web Java** 11](#_Toc481879163)

[**3.1.1. Phân tích hệ thống** 11](#_Toc481879164)

[**3.1.1.1. Thiết kế lược đồ Use Case** 11](#_Toc481879165)

[**3.1.1.2. Thiết kế cơ sở dữ liệu** 11](#_Toc481879166)

[**3.1.2. Thiết kế giao diện hệ thống** 11](#_Toc481879167)

[**3.1.3. Thiết kế phần back-end** 11](#_Toc481879168)

[**3.1.4. Viết Junit cho hệ thống** 11](#_Toc481879169)

[**3.2. Đánh giá kết quả đạt được và hạn chế** 11](#_Toc481879170)

[**TÀI LIỆU THAM KHẢO** 11](#_Toc481879171)

[**PHỤ LỤC** 11](#_Toc481879172)

# **LỜI CẢM ƠN**

Để hoàn thành được báo cáo thực tập tốt nghiệp với đề tài “PHÂN TÍCH WEB JAVA” tại công ty DXC Technology, tôi xin chân thành cảm ơn đến:

Giáo viên hướng dẫn – thầy Nguyễn Văn Thắng, trong quá trình nghiên cứu, mặc dù bận rộn với nhiều công việc nhưng Thầy vẫn nhiệt tình hướng dẫn tôi hoàn thành đề tài. Từ đề cương đến nội dung báo cáo, Thầy đều xem xét, chỉ ra những lỗi sai, đưa ý kiến giúp tôi hoàn thiện báo cáo của mình.

Tôi cũng chân thành cảm ơn các Thầy Cô khoa Công nghệ thông tin trong suốt những năm qua đã tận tình hướng dẫn, giảng dạy và giúp đỡ. Tôi xin gửi lời cảm ơn sâu sắc đến ngày hôm nay, tạo điều kiện cho tôi có thời gian thử sức với công việc thực tế.

Bên cạnh đó, tôi xin gửi lời cảm ơn đến Quý Công ty DXC Technology cùng các anh chị trong công ty và team đã nhiệt tình hướng dẫn và tạo điều kiện để tôi có thể đảm nhận công việc như một nhân viên thực sự.

Cuối cùng tôi xin kính chúc quý Thầy Cô trường Đại học Công nghiệp Thành phố Hồ Chí Minh, Quý Công ty cùng toàn thể các anh chị trong team Intergral luôn dồi dào sức khỏe và thành công trong công việc.

Tôi xin chân thành cảm ơn !

Tp Hồ Chí Minh, ngày 06 tháng 05 năm 2017

Sinh viên thực tập

Trương Thanh Đạm

# **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Tp.HCM, ngày … tháng … năm …

Giáo viên hướng dẫn

# **CHƯƠNG 1. TỔNG QUAN VỀ CÔNG TY DXC TECHNOLOGY VIỆT NAM**

## **1.1. Quá trình hình thành và phát triển của Công ty**

Lịch sử hình thành của công ty DXC Technology Việt Nam (tên cũ là CSC Việt Nam)

- Vào tháng 09 năm 1995, công ty Paragon Solutions (Chicago - Mỹ) hợp tác với công ty Tin Tiến (do 10 nhân viên từng làm việc tại trung tâm điện toán trường Đại học Bách Khoa Tp. Hồ Chí Minh) thành lập công ty Paragon Solutions Việt Nam (PSV).

- Tháng 01 năm 2002, PSV vinh dự được nhận Bằng Khen của Ủy Ban Nhân Dân TP HCM ghi nhận những thành tích đóng góp cho công nghiệp phát triển phần mềm của Thành phố và tiếp tục được nhận bằng khen trong các năm sau đó.

- Tháng 03 năm 2003, PSV sát nhập vào First Consulting Group (FCG) và đổi tên thành FCG Việt Nam.

- Tháng 03 năm 2004, công ty vinh dự được nhận giải thưởng “Sao Khuê” do VINASA trao tặng. Chúng tôi đã đoạt giải thưởng Sao Khuê ngay từ năm đầu tiên giải thưởng này được thành lập và tiếp tục đoạt giải trong các năm kế tiếp.

- Tháng 01 năm 2005, công ty tự hào là công ty đầu tiên tại Việt Nam đạt chuẩn CMMi Cấp 5- cấp cao nhất về quy trình quản lý chất lượng phát triển phần mềm của Viện Công Nghệ Phần Mềm Mỹ (SEI).).

- Tháng 01 năm 2006, công ty vinh dự nhận “Giải thưởng 40 doanh nghiệp có vốn đầu tư nước ngoài hàng đầu của Saigon Times” - giải thưởng dành cho các doanh nghiệp có thành tích đóng góp tích cực đối với cộng đồng.

- Tháng 12 năm 2006, Công ty trở thành công ty đầu tiên tại Việt Nam đạt được chứng chỉ TL 9000 phiên bản 4.0 (phiên bản mới nhất của Hệ Thống Quản Lý Chất Lượng Đặc Thù Cho Ngành Viễn Thông) và chứng chỉ về Bảo Mật Thông Tin ISO 27001,2005 (phiên bản mới nhất của Hệ Thống Quản Lý Bảo Mật Thông Tin).

- Tháng 03 năm 2007, công ty được bình chọn trong danh sách “50 nhà tuyển dụng hàng đầu Việt Nam” do Navigos Group, báo Thanh Niên và công ty ACNielsen phối hợp tổ chức.

- Tháng 07 năm 2007, công ty bắt đầu cung cấp và triển khai các Giải Pháp Toàn Diện của SAP cho thị trường trong nước.

- Tháng 05 năm 2008, FCG Việt Nam chính thức đổi tên thành CSC Việt Nam và hoàn toàn hòa nhập vào hệ thống của CSC toàn cầu.

- Tháng 07 năm 2009, FCG Việt Nam chính thức đổi tên thành CSC Việt Nam và hoàn toàn hòa nhập vào hệ thống của CSC toàn cầu.

- Tháng 01 năm 2011, công ty vinh dự nhận được bằng khen của Ủy Ban Nhân Dân TP HCM ghi nhận Những đóng góp tích cực của CSC cho các công tác trách nhiệm xã hội đối với cộng đồng.

- Tháng 12 năm 2011, công ty vinh dự đón nhận giải thưởng “Đối tác cung cấp dịch vụ SAP tốt nhất” do SAP Việt Nam trao tặng

- Tháng 12 năm 2012, công ty vinh dự đón nhận giải thưởng “Đối tác cung cấp dịch vụ SAP tốt nhất” tại Việt Nam do SAP Việt Nam trao tặng.

- Tháng 12 năm 2014, công ty được vinh danh “Doanh Nghiệp có chính sách nhân sự xuất sắc”.CSC Việt Nam chiến thắng ở 2 hạng mục trong 5 phân mục thuộc lĩnh vực quản lý nhân sự, “Quản Lý Hiệu Quả Công việc Xuất Sắc” và “Quản Trị Nhân Sự Hiệu Quả trong quá trình chuyển đổi” do hội đồng thẩm định Vietnam HR Awards 2014 bình chọn.

- Tháng 03 năm 2016, công ty vinh dự đón nhận giải thưởng “Ðối tác cung cấp dịch vụ SAP Sáng Tạo nhất"tại Việt Nam do SAP Việt Nam trao tặng.

- Tháng 04 năm 2016, CSC Việt Nam vinh dự đón nhận giải thưởng “Công Nghệ Xuất Sắc 2016” ở hàng mục “Ứng Dụng Di Động“ - do hội đồng thẩm định GSC Tetchcom của tập đoàn CSC tổ chức.

- Tháng 09 năm 2016, công ty được vinh danh “Doanh Nghiệp có chính sách nhân sự xuất sắc” 2016. CSC Việt Nam chiến thắng ở 3 hạng mục trong 5 phân mục thuộc lĩnh vực quản lý nhân sự, “Quản Lý Hiệu Quả Công việc Xuất Sắc” , “Hoạch định và Tuyển dụng nguồn nhân lực Xuất sắc”, và “Môi trường làm việc tốt” do Vietnam HR Awards 2016 bình chọn.

- Tháng 04 năm 2017, CSC Việt Nam đổi tên thành DXC Technology Việt Nam (“DXC Việt Nam”, là chi nhánh của tập đoàn DXC Technology - được hình thành vào tháng 4 năm 2017 từ sự sáp nhập giữa CSC và Hewlett Packard Enterprise (HPE).

## **1.2. Đặc điểm hoạt động kinh doanh của công ty DXC Technology Việt Nam**

DXC Technology là tập đoàn Công nghệ thông tin độc lập hàng đầu thế giới. DXC chuyên cung cấp các giải pháp IT toàn diện cho khách hàng toàn cầu bao gồm các dịch vụ Công nghệ thông tin trọn gói để giúp các doanh nghiệp phát triển vượt trội trong thời kỳ kỷ nguyên số. Được hình thành từ sự sáp nhập giữa CSC và mảng Dịch vụ Doanh nghiệp của Hewlett Packard Enterprise (HPE), DXC Technology phục vụ hơn 6.000 khách hàng bao gồm các doanh nghiệp thương mại và các tổ chức dịch vụ công tại hơn 70 quốc gia. Với sự độc lập về công nghệ, đội ngũ nhân viên tài năng toàn cầu, liên minh chặt chẽ với các nhà đối tác công nghệ hàng đầu thế giới và luôn ứng dụng các giải pháp công nghệ thế hệ mới, chúng tôi đảm bảo sẽ đem lại thành công mỹ mãn cho khách hàng.

Tại Việt Nam, DXC là một trong những công ty hàng đầu trong ngành Công nghệ thông tin , cung cấp các dịch vụ phần mềm và các dịch vụ Công nghệ thông tin cho khách hàng toàn cầu và các doanh nghiệp trong nước. Chúng tôi luôn tiên phong áp dụng những giải pháp thế hệ mới với các công nghệ hiện đại nhất trong các lĩnh vực như Ứng dụng Di động, Trí tuệ Nhân tạo, hiện đại hóa các ứng dụng, tư vấn và triển khai SAP ERP tại các ngành công nghiệp khác nhau như Thương mại, Bảo hiểm, Ngân hàng, Sản xuất, Giáo dục, Chăm sóc Sức khỏe và Khoa học Đời sống..

Với hơn hai thập kỷ hoạt động tại Việt nam, DXC Việt Nam có một đội ngũ kỹ sư hơn 1,100 nhân viên đầy kinh nghiệm với trình độ chuyên môn cao, năng động, liên tục cập nhật những kỹ thuật công nghệ mới, sẵn sàng đáp ứng mọi nhu cầu đa dạng của khách hàng. Cùng với sự kết hợp hoàn hảo các giải pháp kinh doanh sáng tạo và công nghệ hiện đại, quy trình phát triển chuẩn quốc tế và sự am hiểu sâu sắc về các lĩnh vực chuyên ngành, chúng tôi chắc chắn sẽ đem lại sự hài lòng và thành công cho khách hàng - đó là cung cấp những dịch vụ phát triển ứng dụng chất lượng cao với chi phí hợp lý.

Là một bộ phận quan trọng trong hệ thống phát triển toàn cầu của DXC, DXC tại Việt Nam đã xây dựng được một hình ảnh xuất sắc của một trung tâm chất lượng cao. Chúng tôi luôn áp dụng chặt chẽ những qui trình phát triển ứng dụng một cách nhất quán trong quá trình phát triển phần mềm để đảm bảo chất lượng sản phẩm, là chìa khóa cho dịch vụ tuyệt hảo của chúng tôi. DXC Việt Nam tự hào là công ty đầu tiên tại Việt Nam đạt chuẩn CMMI Cấp 5 - cấp cao nhất về quy trình quản lý chất lượng phát triển phần mềm của Viện Công Nghệ Phần Mềm Mỹ (SEI) và Chứng chỉ ISO 27001:2013 (phiên bản mới nhất của hệ thống quản lý an toàn thông tin).

## **1.3. Đặc điểm tổ chức quản lý của công ty**

### **1.3.1. Đặc điểm phân cấp quản lý**

### **1.3.2. Số lượng nhân sự và chức năng từng phòng ban của DXC Technology Việt Nam**

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ LẬP TRÌNH WEB JAVA (BACK-END)**

## **2.1. Tìm hiểu về Java**

### **2.1.1. Giới thiệu về ngôn ngữ lập trình Java**

Java vừa là một ngôn ngữ lập trình mà cũng vừa là một nền tảng , cú pháp ngôn ngữ Java có phần giống với ngôn ngữ C và C++.

Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun Microsystems năm 1991, thời điểm sơ khai thì ngôn ngữ này được gọi là Oak. Đến năm 1995 thì đổi tên thành Java 1.0. Khoảng năm 2009 thì tập đoàn Oracle đã mua lại Java bởi công ty Sun Microsystems và hiện vẫn đang tiếp tục xây dựng và phát triển các phiên bản tiếp theo.

Java được tạo ra với tiêu chí "Viết một lần, thực thi khắp mọi nơi" (Write Once, Run Anywhere). Những chương trình phần mềm được viết bằng ngôn ngữ lập trình Java có thể chạy trên mọi nền tảng khác nhau với điều kiện cần phải có môi trường máy ảo Java (JVM – Java Virtual Machine).

Một số nền tảng hiện nay hỗ trợ Java chẳng hạn như Sun Solaris, Linux, Mac OS, FreeBSD & Windows…

Ngày nay, Java được sử dụng rộng rãi như: ứng dụng dành cho máy tính để bàn (Desktop Applications), ứng dụng web (Web Applications), ứng dụng di động (Mobile).

Nói về Java thì không thể không nói đến JVM (Java Virtual Machine), JVM là phần mềm giả lập máy tính, nó tập hợp các lệnh logic để xác định hoạt động của máy, có thể xem nó là hệ điều hành nhỏ. JVM chuyển mã Byte Code thành Machine Code tùy theo môi trường tương ứng (gọi là khả năng khả chuyển). JVM cung cấp môi trường thực thi cho chương trình Java (gọi đó là khả năng độc lập về nền tảng). Sun MicroSystem chịu trách nhiệm thiết kế, phát triển các máy ảo Java chạy trên các hệ điều hành cũng như kiến trúc phần cứng khác nhau. JVM có sẵn cho nhiều nền tảng (Windows, Linux…).

Ngôn ngữ lập trình Java có 5 đặc điểm chính:

- Hướng đối tượng:

+ Trong quá trình tạo ra một ngôn ngữ mới phục vụ cho mục đích chạy được trên nhiều nền tảng, các kỹ sư của Sun MicroSystem muốn tạo ra một ngôn ngữ dễ học và quen thuộc với đa số người lập trình. Vì vậy họ đã sử dụng lại các cú pháp của C và C++.

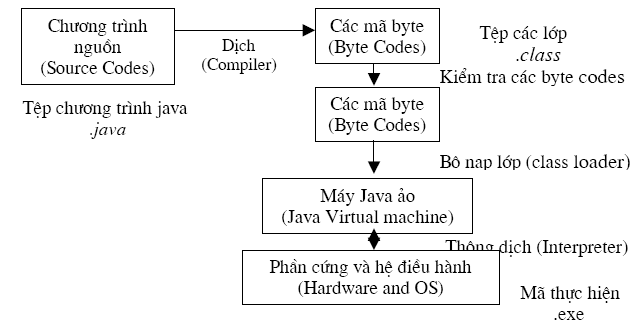
+ Tuy nhiên, trong Java thao tác với con trỏ bị lược bỏ nhằm đảo bảo tính an toàn và dễ sử dụng hơn. Các thao tác overload, goto hay các cấu trúc như struct và union cũng được loại bỏ khỏi Java.

- Độc lập nền tảng: Một chương trình viết bằng ngôn ngữ Java có thể chạy tốt ở nhiều môi trường khác nhau. Gọi là khả năng "cross-platform”. Khả năng độc lập phần cứng và hệ điều hành được thể hiện ở 2 cấp độ là cấp độ mã nguồn và cấp độ nhị phân.

+ Ở cấp độ mã nguồn: Kiểu dữ liệu trong Java nhất quán cho tất cả các hệ điều hành và phần cứng khác nhau. Java có riêng một bộ thư viện để hỗ trợ vấn đề này. Chương trình viết bằng ngôn ngữ Java có thể biên dịch trên nhiều loại máy khác nhau mà không gặp lỗi.

+ Ở cấp độ nhị phân: Một mã biên dịch có thể chạy trên nhiều nền tảng khác nhau mà không cần dịch lại mã nguồn. Tuy nhiên cần có Java Virtual Machine để thông dịch đoạn mã này.

- Ngôn ngữ thông dịch: Ngôn ngữ lập trình thường được chia ra làm 2 loại (tùy theo các hiện thực hóa ngôn ngữ đó) là ngôn ngữ thông dịch và ngôn ngữ biên dịch. Bạn có thể xem thêm tại đây - Phân Biệt Compile Và Interpret. Ngôn ngữ lập trình Java thuộc loại ngôn ngữ thông dịch. Chính xác hơn, Java là loại ngôn ngữ vừa biên dịch vừa thông dịch. Khi viết mã, hệ thống tạo ra một tệp .java. Khi biên dịch mã nguồn của chương trình sẽ được biên dịch ra mã byte code. Máy ảo Java (Java Virtual Machine) sẽ thông dịch mã byte code này thành machine code (hay native code) khi nhận được yêu cầu chạy chương trình.



Hình 1. Quá trình thông dịch của Java

+ Ưu điểm: Phương pháp này giúp các đoạn mã viết bằng Java có thể chạy được trên nhiều nền tảng khác nhau. Với điều kiện là JVM có hỗ trợ chạy trên nền tảng này.

+ Nhược điểm: Cũng như các ngôn ngữ thông dịch khác, quá trình chạy các đoạn mã Java là chậm hơn các ngôn ngữ biên dịch khác (tuy nhiên vẫn ở trong một mức chấp nhận được).

- Cơ chế cùa trình dọn rác (Garbage Collector) tự động:

+ Khi tạo ra các đối tượng trong Java, JRE (Java Runtime Environment) sẽ tự động cấp phát không gian bộ nhớ cho các đối tượng ở trên heap.

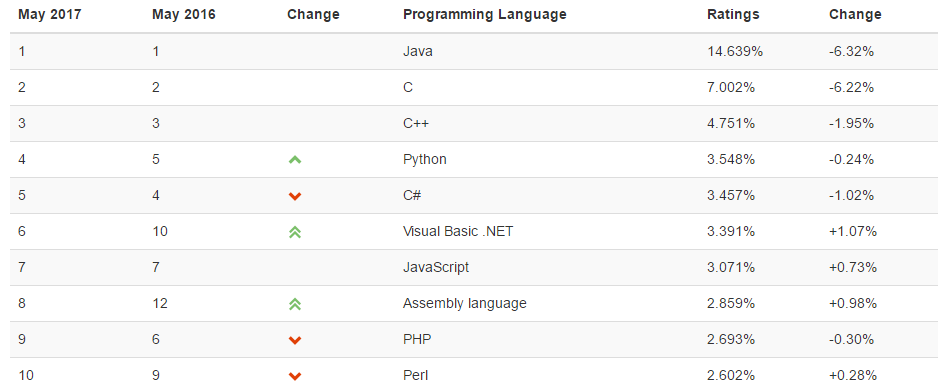
+ Với ngôn ngữ như C \ C++, bạn sẽ phải yêu cầu hủy vùng nhớ mà bạn đã cấp phát, để tránh việc thất thoát vùng nhớ. Tuy nhiên vì một lý do nào đó, bạn không hủy một vài vùng nhớ, dẫn đến việc thất thoát và làm giảm hiệu năng chương trình.

+ Ngôn ngữ lập trình Java hỗ trợ cho bạn điều đó, nghĩa là bạn không phải tự gọi hủy các vùng nhớ. Bộ thu dọn rác của Java sẽ theo vết các tài nguyên đã được cấp. Khi không có tham chiếu nào đến vùng nhớ, bộ thu dọn rác sẽ tiến hành thu hồi vùng nhớ đã được cấp phát.

- Đa luồng: Java hỗ trợ lập trình đa tiến trình (multithread) để thực thi các công việc đồng thời. Đồng thời cũng cung cấp giải pháp đồng bộ giữa các tiến trình (giải pháp sử dụng priority...).

### **2.1.2. Tại sao lại phát triển web bằng ngôn ngữ Java?**

Chỉ số TIOBE năm 2017 cho thấy, Java vẫn tiếp tục trở thành ngôn ngữ lập trình thông dụng thứ 1 trên thế giới. Trong dự báo về Top danh sách các ngôn ngữ lập trình dành cho người tìm việc (Job Seekers) năm 2014 của eWeek , Java đứng ở vị trí số 1. Jobs Tractor phân tích dựa trên 45 000 quảng cáo tìm việc trên Twitter trong suốt 12 tháng cho thấy Java cũng ở vị trí số 1 (không kể Java for Android ) và PHP ở vị trí số 2.



Hình 2. Bảng xếp hạng của TIOBE về độ phổ biến của ngôn ngữ lập trình

Một vài báo cáo khác về danh sách các ngôn ngữ lập trình thông dụng nhất cũng cho thấy Java luôn đứng ở vị trí số 1 trong danh sách các nền tảng lập trình. Tại Việt Nam, Java đang dần trở thành một nền tảng lập trình thông dụng do yêu cầu công việc từ outsourcing và các dự án lớn đòi hỏi tính bảo mật, khả năng đáp ứng xử lý lớn, nguồn mở,…

Từ Wikipedia cho thấy, Java là lựa chọn đầu, đảm đương phần back-end cho các websites. Các hãng lớn như Google, Facefook, Twitter, Amazon, Ebay,… cũng đều lựa chọn Java bên cạnh việc sử dụng các nền tảng lập trình khác.

Bên cạnh đó, Java là ngôn ngữ lập trình tốt nhất ở các khía cạnh cơ hội việc làm, phát triển các ứng dụng và hỗ trợ cộng đồng. Sau đây là những lý do giải thích cho việc Java luôn giữ vị trí ngôn ngữ lập trình tiên phong mặc dù nó đã xuất hiện cách đây hai thập kỷ:

- Lý do 1: Java là một ngôn ngữ lập trình hướng đối tượng

Một lý do khác khiến Java phổ biến bởi vì nó là một ngôn ngữ lập trình hướng đối tượng. Phát triển OOPS (Object-Oriented Programming System – hệ thống lập trình hướng đối tượng) dễ dàng hơn nhiều, và nó cũng duy trì module hệ thống, linh hoạt và mở rộng. Một khi bạn có kiến thức về định nghĩa OOP như sự trừu tượng, đóng gói, đa nhiệm và thừa kế, bạn có thể sử dụng chúng trong Java. Bản thân Java là hiện thân của nhiều phương pháp tư duy tốt nhất và các mẫu thiết kế trong các thư viện của nó. Java là một trong số ít ngôn ngữ lập trình đạt tới gần 100% OOPS. Java cũng thúc đẩy việc sử dụng các nguyên tắc SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion – một dạng thiết kế hướng đối tượng) và thiết kế hướng đối tượng theo hình thức dự án mã nguồn mở như Spring, mà chắc chắn rằng đối tượng phụ thuộc của bạn được quản lý tốt bằng việc sử dụng nguyên lý Injection (một lý thuyết trong thiết kế phần mềm).

- Lý do 2: Số lượng hàm dùng sẵn (API function) của Java hết sức phong phú

Một lý do khác mang lại thành công lớn cho ngôn ngữ lập trình Java là nó nhiều API. Quan trọng nhất là nó rất dễ nhìn, bởi vì nó xuất hiện cùng với việc cài đặt Java. Khi tôi bắt đầu lập trình Java, tôi viết mã cho các Applet và thời đó Applet là một giải pháp cho các hiệu ứng hoạt hình tuyệt vời, điều đó mang lại sự ngạc nhiên cho những lập trình viên mới như chúng tôi, những người đang sử dụng mã nguồn trong Turbo C++. Java cung cấp API cho I/O (giao tiếp dữ liệu), mạng, tiện ích, XML, phân tích cú pháp, kết nối cơ sở dữ liệu, và gần như tất cả mọi thứ. Những điều còn lại được chứa trong các thư viện mã nguồn mở như Apache, Google Guava và một số chương trình khác.

- Lý do 3: Các công cụ phát triển mạnh mẽ như Eclipse, Netbeans

Eclipse và Netbeans đã đóng vai trò rất lớn trong việc làm cho Java trở thành một trong những ngôn ngữ lập trình tốt nhất. Viết mã trong IDE (Integrated Development Environment)là niềm vui, đặc biệt nếu bạn đã từng viết mã trong hệ điều hành DOS Editor hoặc Notepad. Chúng không chỉ giúp hoàn thành mã mà còn cung cấp khả năng sửa lỗi mạnh mẽ, điều đó là cần thiết trong môi trường lập trình thực tế. Môi trường phát triển tích hợp (IDE) giúp cho việc phát triển Java dễ dàng hơn, nhanh và thuận tiện hơn. Tìm kiếm, tổ chức lại mã và đọc mã bằng IDE. Ngoài IDE, nền tảng Java cũng có một số công cụ khác như Maven và ANT để dịch và đóng gói ứng dụng Java, dịch ngược mã, JConsole, Visual VM để giám sát bộ nhớ Heap…

- Lý do 4: Bộ sưu tập thư viện mã nguồn mở phong phú

Thư viện mã nguồn mở đảm bảo rằng Java có thể được sử dụng ở khắp mọi nơi. Apache, Google, và các tổ chức khác đã đóng góp rất nhiều thư viện lớn, giúp Java phát triển dễ dàng hơn, nhanh hơn và tiết kiệm chi phí. Có những cấu trúc như Spring, Struts, Maven đảm bảo sự phát triển Java theo phương pháp xây dựng phần mềm tốt nhất, thúc đẩy sử dụng các mẫu thiết kế và hỗ trợ lập trình viên Java hoàn thành công việc.

- Lý do 5: Hỗ trợ cộng đồng tuyệt vời

Cộng đồng là sức mạnh lớn nhất của ngôn ngữ lập trình Java và nền tảng này. Một ngôn ngữ dù tốt thế nào đi nữa cũng sẽ không thể tồn tại nếu không có cộng đồng hỗ trợ, giúp đỡ và chia sẻ kiến thức. Java đã rất may mắn, nó có rất nhiều diễn đàn hoạt động, StackOverflow, tổ chức mã nguồn mở và một số nhóm người sử dụng Java giúp đỡ lẫn nhau. Cộng đồng các lập trình viên Java có thâm niên và thậm chí cả các chuyên gia sẽ giúp đỡ người mới bắt đầu. Java thực sự thúc đẩy việc thu nhận kiến thức và đóng góp hỗ trợ lại cộng đồng. Rất nhiều lập trình viên, những người sử dụng mã nguồn mở, tham gia vào xây dựng, nâng cấp mã mở, kiểm thử,… Các chuyên gia tư vấn miễn phí tại nhiều diễn đàn Java và StackOverflow. Điều tuyệt vời này đã mang lại tự tin cho những lập trình viên Java.

- Lý do 6: Java là miễn phí

Nếu một lập trình viên muốn học một ngôn ngữ lập trình, hoặc một tổ chức muốn sử dụng một công nghệ, chi phí là một yếu tố quan trọng. Vì Java là miễn phí ngay từ đầu, tức là bạn không cần phải trả bất cứ khoản chi phí nào để tạo ra các ứng dụng Java. Chính điều này cũng giúp Java trở thành kỹ năng thông dụng trong cộng đồng lập trình viên và các tổ chức lớn. Sự dồi dào lập trình viên Java là một lợi thế lớn, làm cho các tổ chức dễ dàng lựa chọn Java cho chiến lược phát triển.

- Lý do 7: Hỗ trợ tài liệu xuất sắc – Javadocs

Javadoc làm cho việc học dễ dàng và cung cấp một tài liệu tham khảo tuyệt vời trong khi viết mã Java. Với sự xuất hiện của IDE, bạn thậm chí không cần phải nhìn Javadoc một cách rõ ràng trong trình duyệt mà bạn đã có thể nhận được tất cả thông tin trong cửa sổ IDE của bạn.

- Lý do 8: Java là nền tảng độc lập

Trong năm 1990, đây là lý do chính khiến Java phổ biến. Ý tưởng về nền tảng độc lập là rất tuyệt, và slogan của Java “viết một lần chạy mọi nơi” đã đủ sức lôi kéo để thu hút rất nhiều sự phát triển mới trong Java. Điều này vẫn còn là một trong những lý do để Java là ngôn ngữ lập trình tốt nhất, hầu hết các ứng dụng Java được phát triển trong môi trường Windows và chạy trên nền tảng UNIX.

- Lý do 9: Java có ở khắp mọi nơi

Java hỗ trợ phát triển nhiều loại hình ứng dụng khác nhau chẳng hạn như J2SE (Java 2 Standard Edition) dành cho phát triển ứng dụng client-server, J2ME (Java 2 Micro Edition) dành cho phát triển ứng dụng trên thiết bị di động, J2EE (Java 2 Enterprise Edition) dành cho phát triển ứng dụng thương mại.

## **2.2. Tìm hiểu về Web**

### **2.2.1. Web là gì?**

World Wide Web đã được tạo ra vào năm 1990 của CERN bởi kỹ sư Tim Berners-Lee. Ngày 30 tháng tư năm 1993, CERN thông báo rằng World Wide Web sẽ được miễn phí để sử dụng cho bất cứ ai.

World Wide Web, gọi tắt là Web hoặc WWW, mạng lưới toàn cầu là một không gian thông tin toàn cầu mà mọi người có thể truy nhập (đọc và viết) qua các máy tính nối với mạng Internet. Thuật ngữ này thường được hiểu nhầm là từ đồng nghĩa với chính thuật ngữ Internet. Nhưng Web thực ra chỉ là một trong các dịch vụ chạy trên Internet, chẳng hạn như dịch vụ thư điện tử. Web được phát minh và đưa vào sử dụng vào khoảng năm 1990, 1991 bởi viện sĩ Viện Hàn lâm Anh Tim Berners-Lee và Robert Cailliau (Bỉ) tại CERN, Geneva, Switzerland.

Các tài liệu trên World Wide Web được lưu trữ trong một hệ thống siêu văn bản (hypertext), đặt tại các máy tính trong mạng Internet. Người dùng phải sử dụng một chương trình được gọi là trình duyệt web (web browser) để xem siêu văn bản. Chương trình này sẽ nhận thông tin (documents) tại ô địa chỉ (address) do người sử dụng yêu cầu (thông tin trong ô địa chỉ được gọi là tên miền (domain name)), rồi sau đó chương trình sẽ tự động gửi thông tin đến máy chủ (web server) và hiển thị trên màn hình máy tính của người xem. Người dùng có thể theo các liên kết siêu văn bản (hyperlink) trên mỗi trang web để nối với các tài liệu khác hoặc gửi thông tin phản hồi theo máy chủ trong một quá trình tương tác. Hoạt động truy tìm theo các siêu liên kết thường được gọi là duyệt Web.Quá trình này cho phép người dùng có thể lướt các trang web để lấy thông tin. Tuy nhiên độ chính xác và chứng thực của thông tin không được đảm bảo.

Tùy theo công việc của một cá nhân, một doanh nghiệp hoặc các tổ chức mà phân loại web. Bất kỳ trang web nào cũng có thể chứa một siêu liên kết vào bất kỳ trang web khác, do đó, phân biệt các trang web cá nhân, như cảm nhận của người sử dụng. Tạm thời phân loại như sau:

- Trang web cá nhân

- Trang web thương mại

- Trang web của chính phủ

- Trang web tổ chức phi lợi nhuận

Website là một tập hợp các trang web (web pages) bao gồm văn bản, hình ảnh, video, flash v.v… thường chỉ nằm trong một tên miền (domain name) hoặc tên miền phụ (subdomain). Trang web được lưu trữ (web hosting) trên máy chủ web (web server) có thể truy cập thông qua Internet.

Website đóng vai trò là một văn phòng hay một cửa hàng trên mạng Internet – nơi giới thiệu thông tin về doanh nghiệp, sản phẩm hoặc dịch vụ do doanh nghiệp cung cấp… Có thể coi website chính là bộ mặt của doanh nghiệp, là nơi để đón tiếp và giao dịch với các khách hàng, đối tác trên Internet.

Trình duyệt web là một phần mềm ứng dụng cho phép người sử dụng xem và tương tác với các văn bản, hình ảnh, đoạn phim, nhạc, trò chơi và các thông tin khác ở trên một trang web của một địa chỉ web trên mạng toàn cầu hoặc mạng nội bộ. Văn bản và hình ảnh trên một trang web có thể chứa siêu liên kết tới các trang web khác của cùng một địa chỉ web hoặc địa chỉ web khác. Trình duyệt web cho phép người sử dụng truy cập các thông tin trên các trang web một cách nhanh chóng và dễ dàng thông qua các liên kết đó. Trình duyệt web đọc định dạng HTML để hiển thị, do vậy một trang web có thể hiển thị khác nhau trên các trình duyệt khác nhau.

Một số trình duyệt web hiện nay cho máy tính cá nhân bao gồm Internet Explorer, Mozilla Firefox, Safari, Opera, Avant Browser, Konqueror, Lynx, Google Chrome, Flock, Arachne, Epiphany, K-Meleon và AOL Explorer.

### **2.2.2. Tầm quan trọng của Web**

Khi mà internet có một sức hút mạnh mẽ như thế thì tầm quan trọng nó mang lại cho sự phát triển của công ty, doanh nghiệp là rất lớn. Việc thiết kế website riêng cho doanh nghiệp gần như đã trở thành một điều bắt buộc nếu doanh nghiệp muốn thúc đẩy sự phát triển cũng như quảng bá sản phẩm, thương hiệu của mình với người tiêu dùng. Website đã trở thành bộ mặt, một kênh quan trọng để doanh nghiệp tiếp thị sản phẩm của mình tới mọi nơi trên thế giới.

Ngoài ra, tâm quan trọng của web còn thể hiện ở các lợi ích mà nó mang lại như:

- Chi phí thấp so với các ấn phẩm quảng cáo thông thường:

+ Web rất khác các ấn phẩm quảng cáo thông thường với giá rẻ. Web của bạn có thể truy cập trong một thời gian dài, nội dung có thể thay đổi mà không cần phải yêu cầu một ai đó để làm điều đó cho bạn (nếu bạn sử dụng một hệ thống quản lý nội dung – CMS) và bạn có thể có khả năng tiếp cận với một đối tượng rộng lớn hơn.

+ Tuy nhiên, điều đó không có nghĩa là bạn không nên sử dụng các hình thức khác của quảng cáo. Bạn có thể sử dụng tất cả hình thức quảng cáo để thu hút người truy cập vào trang web của bạn và tìm hiểu về công ty của bạn. Ngoài ra website còn có khả năng mở ra cách thức giao tiếp giữa các khách hàng tiềm năng và người bán hàng.

- Thị trường mở rộng: Web cho phép các doanh nghiệp ở bất cứ vị trí địa lý nào đều có thể bỏ qua các rào cản và trở nên dễ tiếp cận. Từ bất cứ quốc gia nào trên thế giới của một khách hàng tiềm năng có thể truy cập Internet cũng có thể biết đến trang web của bạn.

- Đa dạng hóa doanh thu: Một trang web không chỉ là một phương tiện truyền thông đại diện cho công ty của bạn, nó còn là một hình thức của phương tiện thông tin mà từ đó mọi người có thể thu được thông tin. Bạn có thể sử dụng phương tiện truyền thông này để bán không gian quảng cáo cho các doanh nghiệp khác.

- Phục vụ 24/7 và 365 ngày: Khi bạn đóng cửa hàng hoặc nghỉ, website như một công cụ hữu ích để cung cấp thông tin mọi nơi mọi lúc.

- Thuận tiện:

+ Nó có rất nhiều thuận lợi cho khách hàng muốn mua một sản phẩm trên Internet và cũng là nơi yêu cầu một ai đó để có được thông tin của sản phẩm.

+ Các khách hàng tiềm năng có thể truy cập vào trang web của bạn bất cứ khi nào họ muốn một cách riêng tư và thoải mái, không có những căng thẳng tồn tại trong thế giới thực.

- Thêm giá trị gia tăng và hài lòng: Trang web của bạn có thể tăng thêm giá trị trong các mặt khác, bởi có lời khuyên, tư vấn và nội dung quan tâm chung mà bạn có thể cung cấp cho khách hàng. Điều này cũng sẽ giúp khách hàng nhớ bạn tốt hơn.

- Cải thiện tin cậy: Một trang web sẽ cho bạn cơ hội để thông báo cho các khách hàng tiềm năng về những gì bạn đang có và tại sao bạn xứng đáng được họ tin tưởng. Trong thực tế, nhiều người sử dụng Internet để nghiên cứu trước khi mua hàng, họ có thể xác định cho mình cái mà họ cần mua. Internet cũng cho phép các khách hàng của bạn làm tiếp thị cho bạn.

- Cơ hội tăng trưởng: Một trang web là một nơi tuyệt vời để giới thiệu tiềm năng cho các khách hàng.

- Dễ dành nhận thông tin phản hồi: Khách hàng có thể nhanh chóng và dễ dàng đưa ra ý kiến phản hồi về sản phẩm của bạn hoặc các phương pháp tiếp cận thị trường.

- Nghiên cứu thị trường giá rẻ:

+ Bạn có thể sử dụng các tính năng trên trang web của bạn như là phiếu thăm dò ý kiến người truy cập, các cuộc điều tra trực tuyến và các số liệu thống kê của trang web của bạn để tìm hiểu những gì khách hàng muốn để xác định xem bạn có thể cải thiện sản phẩm của bạn và cách thức kinh doanh của bạn.

+ Số liệu thống kê trang web hiển thị cho bạn có bao nhiêu lưu lượng truy cập trang web của bạn nhận được, làm thế nào người truy cập được vào trang web của bạn và ở đâu, người truy cập là từ đâu.

## **2.3. Tìm hiểu AngularJS**

### **2.3.1. AngularJS là gì?**

AngularJS là một dự án mã nguồn mở được phát triển đầu tiên bởi Misko Hevery, một nhân viên của Google. Hevery bắt đầu nghiên cứu và phát triển dự án vào năm 2009 và phiên bản 1.0 được cho ra mắt vào năm 2012. Do sự hữu ích của dự án này nên Google quyết định là công ty chính thức đứng đằng sau hỗ trợ sự phát triển của AngularJS.

Angular là một bộ Javascript Framework rất mạnh và thường được sử dụng để xây dựng project Single Page Application (SPA). Nó hoạt động dựa trên các thuộc tính mở rộng HTML (các atributes theo quy tắc của Angular). Đây là một Framework mã nguồn mở hoàn toàn miễn phí và được hàng ngàn các lập trình viên trên thế giới ưa chuộng và sử dụng.

Định nghĩa chính thức được đưa ra như sau :

AngularJS là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép bạn sử dụng HTML như là ngôn ngữ mẫu và cho phép bạn mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng của bạn một cách rõ ràng và súc tích. Hai tính năng cốt lõi: Data binding và Dependency Injection của AngularJS loại bỏ phần lớn code mà bạn thường phải viết. Nó xảy ra trong tất cả các trình duyệt, làm cho nó trở thành đối tác lý tưởng của bất kỳ công nghệ Server nào.

Các tính năng cốt lõi của AngularJS:

- Data-binding: (liên kết dữ liệu) tự động đồng bộ dữ liệu giữa model và view

- Scope: Đây là những đối tượng kết nối giữa Controller và View

- Controller: Đây là những hàm javascript xử lý kết hợp với bộ điều khiển Scope

- Service: Như tôi đề cập ở trên, AngularJS sử dụng các API được xây dựng từ các web service (PHP, ASP) để thao tác với DB.

- Filters: Bộ lọc lọc ra các thành phẩn của một mảng và trả về mảng mới

- Directives: đánh dấu vào các yếu tố của DOM, nghĩa là sẽ tạo ra các thẻ HTML tùy chỉnh

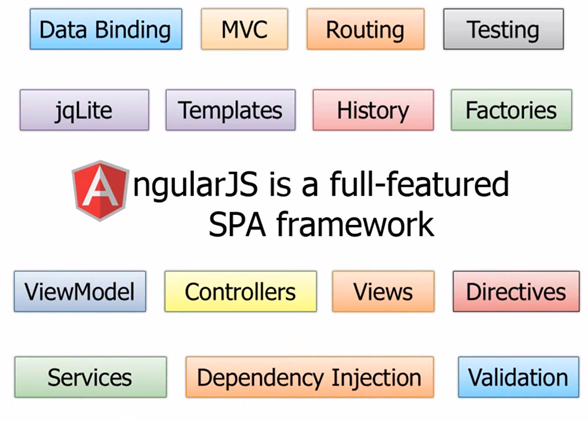
- Templates: hiển thị thông tin từ controller, đây là một thành phần của views

- Routing: chuyển đổi giữa các action trong controller

- MVC: Mô hình chia thành phần riêng biệt thành Model, View, Controller. Đây là một mô hình khá hay nhưng trong Angular thì nó được chế biến lại một chút gần giốn với MVVM (Model View View Model)

- Deep Linking: Liên kết sâu, cho phép bạn mã hóa trạng thái của ứng dụng trong các URL để nó có thể đánh dấu được với công cụ tìm kiếm.

- Dependency Injection: Angular giúp các nhà phát triển tạo ứng dụng dễ dàng hơn để phát triển, hiểu và thử nghiệm dễ dàng.



Hình 3. Các components của Angular

Ưu điểm của AngularJS:

- Angular cho phép tạo ra các ứng dụng một cách đơn giản, code sạch

- Angular sử dụng data bind giống .NET với tính năng liên kết với HTML nên giúp người dùng cảm thấy dễ chịu.

- Angular đang ở giai đoạn thử nghiệm

- Angular có thể chạy trên hầu hết các trình duyệt điện thoại thông minh.

Mặc dù AngularJS có nhiều lợi thế ưu điểm nhưng nó cũng có mặt trái ngược lại:

- Không an toàn: Được phát triển từ javascript nên nó không an toàn, phía máy chủ phải thường xuyên xác nhận quyền để hệ thống chạy trơn tru.

- Phụ thuộc: Nếu người dùng vô hiệu hóa javascript thì AngularJS cũng bị vô hiệu hóa theo.

Các components chính trong AngularJS:

- ng-app: định nghĩa này chỉ thị một kết nối ứng dụng Angular JS tới HTML.

- ng-model: chỉ thị này liên kết với dữ liệu của ứng dụng Angular.

- ng-bind: chỉ thị này dùng đưa dữ liệu vào HTML tags.

### **2.3.2. AngularJS hỗ trợ cho việc phát triển web Java như thế nào ?**

## **2.4. Lập trình Web với Java**

### **2.4.1. Tổng quan về J2EE**

J2EE là thuật ngữ viết tắt của Java 2 Platform, Enterprise Edition , đây là một distributed computing framework của Sun Microsystems. Java EE (Enterprise Edition) là một nền tảng được sử dụng rộng rãi, chứa một tập hợp các công nghệ được phối hợp vào nhau, làm giảm đáng kể chi phí và độ phức tạp của việc phát triển, triển khai và quản lý các tầng làm việc, các ứng dụng máy chủ trung tâm. Java EE được xây dựng dựa trên nền tảng Java SE và cũng cấp thêm một tập các API (giao diện lập trình ứng dụng) để phát triển và hoạt động các ứng dụng phía máy chủ (Server-Side Applications) một cách mạnh mẽ, có khả năng mở rộng, đáng tin cậy, khả chuyển (portable) và bảo mật.

Một số thành phần cơ bản bao gồm:

- Enterprise Java Beans (EJB): một thành phần kiến trúc của các ứng dụng server được quản lý, sử dụng để bao gói (encapsulate) các business logic của các ứng dụng. Công nghệ EJB cho phép phát triển nhanh chóng và đơn giản hóa các ứng dụng phân tán, các giao dịch an toàn và di động dựa trên công nghệ Java.

- Java Persistence API (JPA): một framework cho pháp nhà phát triển quản lý dữ liệu bằng cách sử dụng ánh xạ đối tượng quan hệ (Object Relational Mapping - ORM) trong các ứng dụng được xây dựng trên nền tảng Java.

Những công nghệ được sử dụng trong J2EE bao gồm:

- Java 2 Standard Edition (J2SE) – trước kia còn gọi là JDK

- Java Database Connectivity (JDBC) – Một API chuẩn dùng để kết nối Java Platform với cơ sở dữ liệu.

- RMI-JRMP – Remote Method Invocation (RMI) , một chuẩn dùng để gọi hàm từ xa dựa trên Java Remote Message Protocol (JRMP)

- Java Interface Definition Language (Java IDL) – một dịch vụ kết hợp Common Object Request Broker Architecture (Corba) vào trong Java Platform để cung cấp tính interoperability thông qua chuẩn IDL .

- Remote Method Invocation – Internet Inter- ORB Protocol (RMI-IIOP) – Một giao thức cho phép lập trình viên RMI kết hợp những lợi ích của RMI API và giao thức CORBA IIOP để giao tiếp với CORBA-compliant client được phát triển bởi bất cứ ngôn ngữ nào tuân theo CORBA.

- Enterprise JavaBean (EJB) – một kiến trúc component để phát triển và triển khai các ứng dụng kiểu component-based distributed.

- Servlet – tương tác với Web client thông qua cơ chế request – response. Cơ chế này dựa trên giao thức HTTP.

- Java Server Page (JSP) – Dùng để xây dựng những ứng dụng có chứa nội dung web động như là HTML, DHTML, XML

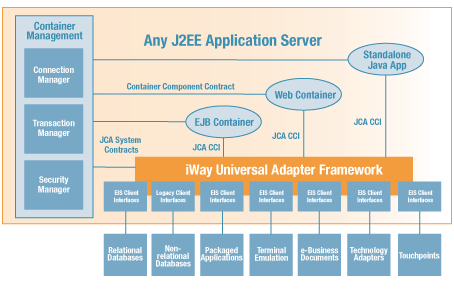
- Java Message Service (JMS) – một API để giao tiếp với Message Oriented Middleware (MOM) cho phép truyền nhận message theo cơ chế point-to-point và publish/subcribre.

- Java Naming and Directory Interface (JNDI) – một interface duy nhất dùng để truy cập những kiểu dịch vụ khác nhau về naming và directory.

- Java Transaction API (JTA) – một tập API cho phép quản lý transaction. Ứng dụng có thể Sử dụng JTA để start, commint, abord transaction.

- JavaMail – một API cung cấp 1 framework platform-independent và protocol-independent (không phụ thuộc giao thức về platform) để xây dựng ứng dụng về mail và messaging

- JavaBean Activation Framework (JAF) – tập các API được sử dụng bởi các package khác như là JavaMail,…Bạn có thể dùng JAF để xác định kiểu dữ liệu, đóng gói truy cập đến dữ liệu đó, mở rộng những tác vụ dựa trên dữ liệu đó và khởi tạo một bean tương ứng để thực hiện những tác vụ đó. Ví dụ JavaMail dùng JAF để xác định object nào sẽ được khởi tạo dựa trên kiểu MIME của object.



Hình 4. Các thành phần của J2EE

J2EE được xây dựng trên một mô hình container component. Bốn container component cốt lõi cung cấp môi trường cho các component khác của J2EE thông qua các API. Những component cốt lõi này liên quan đến bốn kiểu container được hỗ trợ trong J2EE bao gồm, Application Client, Applet, Web và EJB:

- Java Application – component này là 1 chương trình standalone chạy bên trong Application Client container. Application Client container cung cấp những API hỗ trợ cho messaging, remote invocation, database connectivity và lookup service. Application Client container đòi hỏi những API sau: J2SE, JMS, JNDI, RIM-IIOP và JDBC. Container này được cung cấp bởi nhà cung cấp application server

- Applet – Applet component là java applet chạy bên trong Applet container, chính là web browser có hỗ trợ công nghệ Java. Applet phải hỗ trợ J2SE API.

- Servlet và JSP – đây là Web-based component chạy ở bên trong Web container, được hỗ trợ bởi Web Server. Web container là một môi trường run-time cho servlet và jsp. Web Container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC. Serlet và JSP cung cấp một cơ chế cho việc chuẩn bị, xử lý, định dạng nội dung động

- Enterprise JavaBean (EJB) – EJB component là business component chạy bên trong EJB container. EJB component là phần nhân, cốt lõi của ứng dụng J2EE. EJB container cung cấp các dịch vự quản lý transaction, bảo mật, quản lý trạng thái, quay vòng tài nguyên (resource pooling). EJB container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC.

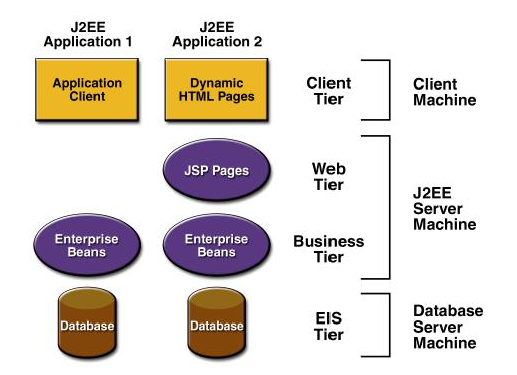
Trong Java EE còn có kiến trúc ứng dụng đa tầng, kiến trúc này có các thành phần: Java EE clients, Web components, Java EE components, Database components, Security components.

- Java EE clients tham chiếu cho end-user hoặc thực thể mà các ứng dụng được thiết kế như: Web clients, Application clients, Applets, Java Bean components.

- Cả hai thành phần Web và các thành phần Java EE thực hiện logic kinh doanh cho các ứng dụng web và các ứng dụng kinh doanh tương ứng: Servlets, JSP, và JSFs là những thành phần Web trong một ứng dụng Web. Các trang tĩnh HTML cũng là những thành phần web. Enterprise beans thực hiện logic trong trường hợp ứng dụng doanh nghiệp.

- Các tầng thông tin doanh nghiệp bao gồm các cơ sở dữ liệu của ứng dụng.

- Thành phần bảo mật liên quan đến việc thực hiện các phương pháp bảo mật liên quan đến các ứng dụng dữ liệu.



Hình 5. Kiến trúc đa tầng và các thành phần tương ứng trong J2EE

#### **2.4.1.1. Giới thiệu về Servlet**

Servlet là thành phần chính được sử dụng để phát triển các chương trình dịch vụ Java ở phía máy chủ. Các Servlet là các chương trình Java thực hiện ở các ứng dụng Server (tên gọi “Servlet” cũng gần giống như “Applet” ở phía máy Client) để trả lời cho các yêu cầu của Client. Các Servlet không bị ràng buộc chặt với một giao thức Client-Server cụ thể nào cả, nhưng giao thức thường được sử dụng là HTTP, do vậy, khi nói tới Servlet nghĩa là nói tới HTTP Servlet. Servlet là sự phát triển mở rộng của Common Gateway Interface (CGI) để đảm bảo Server thực hiện được các chức năng của mình. Ta có thể sử dụng Servlet của Java để tuỳ chỉnh lại một dịch vụ bất kỳ, như Web Server, Mail Server, v.v.

Web Server hiển thị các tư liệu được viết trong HTML và hồi đáp cho yêu cầu của người sử dụng qua HTTP. Các tư liệu HTML chứa các văn bản được đánh dấu (định dạng) để các trình duyệt như IE, Netscape đọc được.

Một trình duyệt chấp nhận đầu vào ở dạng HTML, khi người sử dụng nhấn một nút để yêu cầu một số thông tin nào đó, một Servlet đơn giản được gọi để xử lý các yêu cầu đó. Các công việc chính của Servlet được mô tả khái quát trong hình, bao gồm:

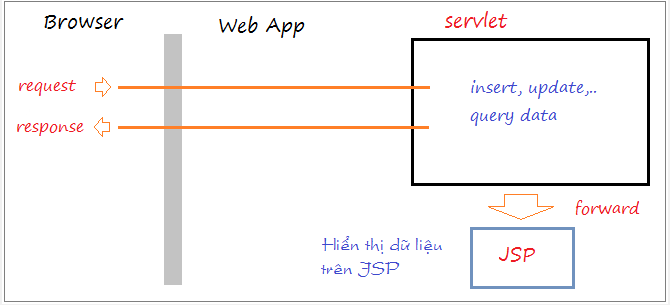
- Đọc dữ liệu hiển thị (explicit) được gửi bởi Client (hoặc trình duyệt) bao gồm một HTML Form trên một trang web hoặc nó cũng có thể từ một Applet hoặc một chương trình Custom từ HTTP Client.

- Đọc dữ liệu yêu cầu HTTP ẩn (implicit) được gửi bởi Client (hoặc trình duyệt) bao gồm cookie, các loại media.

- Xử lý dữ liệu và cho ra kết quả. Tiến trình này có thể yêu cầu Database, đang thực thi một triệu hồi tới RMI hoặc CORBA, triệu hồi một Web Service, hoặc tính toán phản hồi một cách trực tiếp.

- Gửi dữ liệu hiển thị (ví dụ: tài liệu) tới các Client (hoặc trình duyệt). Tài liệu này có thể được gửi theo nhiều định dạng khác nhau, gồm text (HTML hoặc XML), nhị phân (hình ảnh GIF), Excel, .v.v.

- Gửi phản hồi HTTP ẩn tới các Client (hoặc trình duyệt), thông báo cho trình duyệt hoặc Client về kiểu của tài liệu được trả về (ví dụ: HTML), thiết lập các Cookie và Caching các tham số, cùng các tác vụ khác.



Hình 6. Vai trò của Servlet

CGI: Theo cách thông thường, để bổ sung các chức năng vào cho một Web Server người ta hay sử dụng Common Gateway Interface (CGI), một giao diện độc lập với ngôn ngữ cho phép một Server khởi động một tiến trình ngoại để nhận thông tin được yêu cầu thông qua các biến môi trường. Mỗi yêu cầu được trả lời bởi một tiến trình riêng thông qua một đại diện riêng của một chương trình CGI hoặc bởi một kịch bản CGI (thường được viết bằng ngôn ngữ thông dịch như Perl). Java Servlets thường có chung mục đích là các chương trình độc lập bởi việc sử dụng Common Gateway Interface (CGI). Nhưng so với CGI, thì Servlets có các lợi thế sau:

- Hiệu năng tốt hơn đáng kể. Một khi được tải về, nó sẽ được lưu lại trong bộ nhớ của máy chủ. Servlet duy trì các trạng thái của nó, do vậy những tài nguyên bên ngoài như việc kết nối với CSDL cũng sẽ được lưu giữ lại.

- Servlets thực thi bên trong không gian địa chỉ của một Web server, không cần thiết phải tạo một tiến trình riêng biệt để xử lý mỗi yêu cầu từ Client.

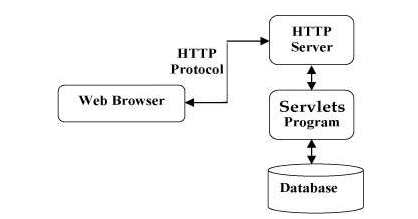
- Servlets là độc lập trên nền tảng bởi vì chúng được viết bằng Java.

- Bởi vì viết bằng Java (mà Java cung cấp tình bảo mật cao trên Server) nên Servlets là đáng tin cậy. Cơ chế tự động dọn rác và việc không sử dụng con trỏ của Java giúp cho Servlet thoát khỏi nhiều công việc quản lý bộ nhớ. Đồng thời nó xử lý các lỗi rất an toàn theo cơ chế xử lý ngoại lệ của Java.

- Tính năng đầy đủ của thư viện của các lớp trong Java là luôn luôn có sẵn cho Servlets. Nó có thể giao tiếp với Applet, Database hoặc phần mềm khác thông qua các Socket và kỹ thuật RMI mà bạn đã từng thấy.

Cấu trúc của Servlets

Sơ đồ dưới đây minh họa cấu trúc của Servlets trong một ứng dụng web:



Hình 7. Cấu trúc Servlet

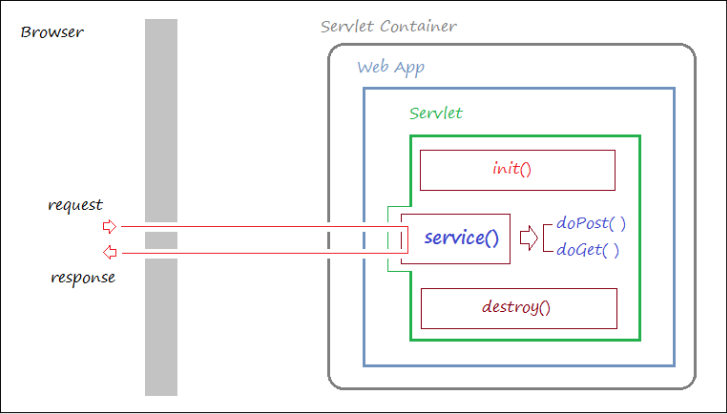
Các lớp Servlet của Java có thể được nạp tự động để mở rộng các chức năng của Server. Các Servlet của Java thực hiện bên trong JVM. Chúng được đảm bảo an toàn và chuyển đổi tương thích giữa các hệ điều hành và giữa các Server với nhau. Điều này khác với các Applet, Servlet chỉ thao tác được trong miền của một Server.

Servlet API được phát triển dựa trên những điểm mạnh của Java platform nhằm giải quyết vấn đề của CGI và Server API. Nó là một API đơn giản, hỗ trợ tất cả các Web server và thậm chí cho phép các ứng dụng máy chủ dùng để kiểm tra và quản lý các công việc trên Server. Nó giải quyết vấn đề thực thi bằng việc thực hiện tất cả các yêu cầu như các luồng Thread trong quá trình xử lý, hoặc việc cân bằng tải trên một Server trong các cụm máy tính Cluster. Các Servlet dễ dàng chia sẻ tài nguyên với nhau.

Trong định nghĩa Servlet, vấn đề bảo mật được cải tiến theo nhiều cách. Trước hết, bạn hiếm khi thực thi được các câu lệnh trên Shell với dữ liệu cung cấp bởi người dùng mà Java API đã cung cấp với những khả năng truy cập đến tất cả các hàm thông dụng. Bạn có thể sử dụng Java Mail để đọc và gửi mail, kết nối vào các CSDL (thông qua JDBC), tệp lớp (.class) và những lớp liên quan để truy cập hệ thống tệp, CSDL, RMI, CORBA, Enterprise Java Beans (EJB), …

Giao diện Servlet là một khái niệm trừu tượng trung tâm trong Servlet API. Tất cả các Servlet đều cài đặt trực tiếp hoặc gián tiếp giao diện này hoặc mở rộng (kế thừa) những lớp đã cài đặt nó.

Giao diện này khai báo ba phương thức định nghĩa vòng đời của Servlet.



Hình 8. Vòng đời của Servlet

*public void init(ServletConfig config) throws ServletException*

Phương thức này được gọi một lần khi Servlet được tải vào trong Servlet Engine, trước khi Servlet được yêu cầu để xử lý một yêu cầu nào đó. Phương thức init() có một thuộc tính là đối tượng của ServletConfig, và Servlet có thể đọc các đối số khởi tạo của nó thông qua đối tượng ServletConfig. Chúng thường được định nghĩa trong một tệp cấu hình. Một ví dụ thông thường của một đối số khởi tạo là định danh database cho một CSDL.

*public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException*

Phương thức này được gọi để xử lý các yêu cầu. Nó có thể không được gọi, gọi một lần hay nhiều lần cho đến khi Servlet được ngưng tải. Nhiều Thread (mỗi Thread cho một yêu cầu) có thể thực thi phương thức này song song, vì thế nó trở nên an toàn và hiệu quả hơn.

*public void destroy()*

Phương thức này chỉ được gọi một lần trước khi Servlet được ngưng tải và sau khi đã kết thúc các dịch vụ.

#### **2.4.1.2. Giới thiệu về JSP**

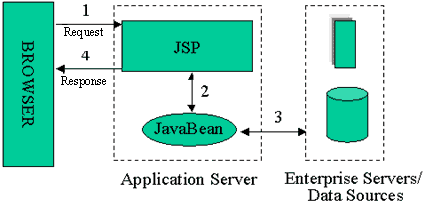
JSP (viết tắt của tiếng Anh JavaServer Pages) còn được biết đến với một cái tên khác là Java Scripting Preprocessor - tạm dịch là "Bộ tiền xử lý văn lệnh Java" - là một công nghệ Java cho phép các nhà phát triển tạo nội dung HTML, XML hay một số định dạng khác của trang web một cách năng động, trong khi hồi âm yêu cầu của trình khách. Công nghệ này cho phép người ta nhúng mã Java và một số hành động xử lý đã được định trước (pre-defined actions) vào trong nội dung tĩnh của trang.

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

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

JSP là cách đơn giản hóa hơn cho Servlet. Nếu như đối với Servlet phải viết mã Java và biên dịch bằng tay trước khi sử dụng với trình chủ Web server thì JSP không cần điều này. JSP viết mã Java tương tự Servlet nhưng cho phép trộn lẫn Java với các thẻ HTML để tạo ra Servlet xử lý các yêu cầu mà trình khác gửi đến trang.

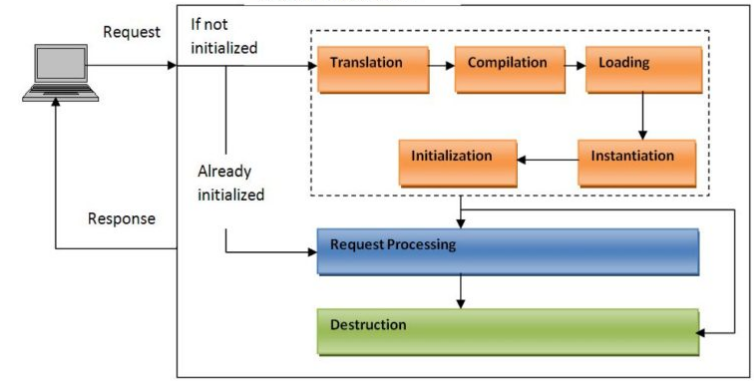
JSP đơn giản và dễ sử dụng hơn Servlet mặc dù sau khi biên dịch và thực thi chúng chỉ là một.



Hình 9. Kiến trúc JSP trong ứng dụng web

Đây là cách thực nạp, dịch và thực thi trang của một trang JSP khi được triệu gọi trên Web Server.

Chu trình sống của JSP



Hình 10. Vòng đời của JSP

Trang JSP có chu trình sống xác định tính từ khi hệ thống đọc biên dịch trang JSP, gọi thực thi và loại bỏ trang ra khỏi bộ nhớ. Chu trình sống của trang JSP gồm có 5 giai đoạn sau:

- Biên dịch trang: Khi trình duyệt yêu cầu trang JSP, Web server sẽ kiểm tra xem trang JSP đã được biên dịch hay chưa. Nếu chưa biên dịch hoặc đã biên dịch nhưng trang JSP mới vừa thay đổi mã nguồn thì Web Server sẽ thực hiện biên dịch trang JSP. Quá trình biên dịch JSP thực tế là chuyển trang JSP thành servlet. File biên dịch .class của trang chỉ diễn ra một lần. Nếu trang đã biên dịch và sau đó không bị thay đổi trong mã nguồn thì quá trình biên dịch sẽ không xảy ra nữa, do đó mà tốc độ thực thi sẽ nhanh hơn. Sau khi biên dịch, mã trang sẽ được nạp vào bộ nhớ để thực thi.

Quá trình biên dịch trang JSP sẽ được diễn ra như sau:

Bước 1: Kiểm tra xem trang đã được dịch thành mã nguồn tương đương servlet hay chưa.

Bước 2: Nếu chưa được biên dịch thì trang JSP sẽ được biên dịch thành file nguồn .java theo cấu trúc của servlet. Gọi trình biên dịch javac biên dịch file nguồn .java thành file thực thi của servlet .class.

Bước 3: Nạp servlet đã biên dịch ở bước 2, thực thi trả kết quả về cho trình khách.

Bước 4: Nếu file JSP đã được biên dịch trước đó : thực hiện kiểm tra xem nội dung file .jsp có thay đổi không, Nếu có thì quay lại bước 2 biên dịch lại trang, nếu không thì quay lại bước 3.

- Nạp trang: Kể từ giai đoạn này, quá trình nạp trang tương tự như servlet (trang JSP sau khi biên dịch có thể coi như một servlet). Chỉ có một điểm khác là servlet chỉ được nạp một lần trong khi mã trang JSP mặc dù đã biên dịch nhưng phải nạp lại nhiều lần mỗi khi web server nhận được yêu cầu trang từ trình duyệt.

- Khởi tạo: Khi nạp mã trang thành công, Web server sẽ gọi đến phương thức khởi tạo trang. Và mặc dù JSP được biên dịch ra servlet nhưng phương thức khởi tạo cho trang JSP lại mang tên là jspInit() chứ không phải là init() như servlet.

- Thực thi: Sau quá trình khởi tạo, Web server sẽ gọi đến phương thức \_jspService (khác với servlet gọi đến doPost(), doGet() hoặc service()). Phương thức \_jspService sẽ chuyển đến hai lớp đối tượng HttpServletRequest và HttpServletResponse để đọc và ghi kết xuất trả về trình khách.

- Dọn dẹp: Khi trang JSP đã thực thi xong, trình chủ Web Server sẽ gọi phương thức jspDestroy() để giải phóng mã trang khỏi bộ nhớ. Tương tự như trong Servlet, có thể cài đặt phương thức jspDestroy() thực hiện giải phóng vùng nhớ hoặc đóng kết nối trả về tài nguyên cho hệ thống.

Lý do sử dụng JSP:

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

- Hiệu năng tốt hơn đáng kể bởi vì JSP cho phép nhúng các phần tử động trong chính các HTML page thay vì phải có một CGI file riêng biệt.

- JSP luôn luôn được biên dịch trước khi nó được xử lý bởi Server, không giống như CGI/Perl mà yêu cầu Server tải một trình thông dịch (Interpreter) và Target Script mỗi khi trang được yêu cầu.

- JavaServer Pages (JSP) được xây dựng ở trên cùng của Java Servlet API, vì thế, giống Servlet, JSP cũng có quyền truy cập tới tất cả Enterprise Java APIs mạnh mẽ, bao gồm JDBC, EJB, JAXP, …

- Các JSP page có thể được sử dụng để kết nối với Servlet mà xử lý Business logic, mô hình được hỗ trợ bởi Java Servlet.

- Cuối cùng, JSP là một phần toàn bộ của Java EE, một Platform đầy đủ cho các ứng dụng lớp Enterprise. Nghĩa là, JSP có thể hoạt động như là một phần trong các ứng dụng đơn giản nhất cũng như các ứng dụng phức tạp.

Lợi thế của việc sử dụng JSP khi so sánh với các công nghệ khác:

- So với Active Server Pages (ASP): Lợi thế của JSP có thể coi là gấp đôi. Đầu tiên, các phần động được viết bằng Java, không phải bằng Visual Basic hoặc ngôn ngữ MS khác, vì thế nó mạnh mẽ hơn và dễ dàng để sử dụng hơn. Thứ hai, nó thích hợp cho các Hệ điều hành khác, không chỉ là Microsoft Web Server.

- So với Pure Servlets: Nó tiện lợi hơn khi viết (và sửa đổi) HTML, vì có nhiều lệnh printIn hơn.

- So với Server-Side Includes (SSI): SSI chỉ dành cho các thể đơn giản, không dành cho các chương trình "thực" mà sử dụng Form Data, tạo kết nối Database.

- So với JavaScript: JavaScript có thể tạo HTML động trên Client nhưng lại tương tác khó khan với Web Server để thực hiện các tác vụ phức tạp như truy cập Database và xử lý hình ảnh, …

- So với Static HTML: Tất nhiên, HTML thông thường không thể chứa thông tin động.

#### **2.4.1.3. JSTL là gì ?**

JSTL (JSP Standard Tag Library) là thư viện thẻ chuẩn dựa trên JSP cái mà đưa ra các thẻ kiểm soát luồng trong trang JSP, định dạng ngày tháng/số, quốc tế hóa và nhiều hàm EL (ngôn ngữ biểu thức -Express Language).

Chú ý JSTL cũng đưa ra thư viện thẻ XML và SQL giúp thực thi truy vấn SQL và phân tích XML trong trang JSP. Tuy nhiên điều này không được khuyến khích, nó chỉ sử dụng cho các ứng dụng đơn giản. Trong thực tế các nhiệm vụ này phải được thực hiện bởi các lớp Java thực sự.

JSTL là một phần của Java EE API và có mặt trong hầu hết (nhưng không phải tất cả) các servlet container. JSTL là những thư viện thẻ mà nhập vào từ: http://java.sun.som/jsp/jstl/\* .

JSTL hỗ trợ tới các tác vụ phổ biến và có tính cấu trúc, ví dụ như các tính lặp và điều kiện, các thẻ để thao tác tài liệu XML, các thẻ đa ngôn ngữ, và các thẻ SQL. Nó cũng cung cấp Framework để tích hợp các Custom Tags với các thẻ JSTL.

Các thẻ JSTL có thể được phân loại, theo tính năng của nó, thành các nhóm thư viện thẻ JSTL sau, mà có thể được sử dụng khi tạo một JSP page:

- Core Tags: Nhóm thẻ cơ bản: là các thẻ JSTL được sử dụng phổ biến nhất

Ví dụ: <%@ taglib prefix="c"

uri="http://java.sun.com/jsp/html/core" %>

- Formatting tags: Nhóm thẻ định dạng: được sử dụng để định dạng và hiển thị text, date, time và số ngôn ngữ trong Website.

Ví dụ: <%@ taglib prefix="fmt"

uri="http://java.sun.com/jsp/html/fmt" %>

- SQL tags: Nhóm thẻ SQL: cung cấp các thẻ để tương tác với các Relational Database (RDBMSs), ví dụ như Oracle, MySQL, hoặc Microsoft SQL Server.

Ví dụ: <%@ taglib prefix="sql"

uri="http://java.sun.com/jsp/html/sql" %>

- XML tags: Nhóm thẻ XML: Thư viện thẻ XML Tags trong JSTL có các custom tags để tương tác với XML data. Điều này bao gồm việc parse XML, truyền tải XML data, và điều khiển dòng dữ liệu dựa trên các XPath Expression.

Ví dụ: <%@ taglib prefix="x"

uri="http://java.sun.com/jsp/html/xml" %>

- JSTL Functions: Nhóm hàm JSTL: JSTL bao gồm một số hàm chuẩn, mà hầu hết là các hàm thao tác chuỗi phổ biến.

Ví dụ: <%@ taglib prefix="fn"

uri="http://java.sun.com/jsp/html/functions" %>

JSTL có các phiên bản:

\* 1.0: bao gồm 2 file Jar jstl.jar (API) và standard.jar (phần triển khai). URI của thư viện thẻ không có /jsp trong đường dẫn giống như http://java.sun.com/jstl/core và phiên bản kiểu có tên thư viện thẻ đi kèm với hậu tố \_rt giống như http://java.sun.com/jstl/core\_rt. Yêu cầu tối thiểu Servlet 2.3/ JSP 1.2. Nó kết thúc cuộc sống và không còn sử dụng nữa.

\* 1.1: bao gồm các file JAR như 1.0. URI của thư viện thẻ bao gồm /jsp trong đường dẫn như http://java.sun.com/jsp/jstl/core. Yêu cầu Servlet 2.4/ JSP 2.0.

\* 1.2: bao gồm một file Jar jstl-1.2.jar (đóng gói cả phần API + phần triển khai). Phần URI giống như 1.1 Yêu cầu Servlet 2.5/ JSP 2.1 nhưng cũng làm việc với Servlet 2.4/JSP 2.0.

\*1.2.1: bao gồm 2 file Jar javax.servlet.jsp.jstl-api-1.2.1 (API) và javax.servlet.jsp.jstl-1.2.1.jar (phần triển khai) và có URI giống như 1.1. Yêu cầu servlet 3.0/ jsp 2.2 nhưng làm việc servlet 2.5/jsp 2.1 và servlet 2.4/ jsp 2.0.

Hầu hết nhưng không phải tất cả các servlet container có JST sẵn. Ví dụ Apache Tomcat không được phát hành với JSTL sẵn có nhưng Oracle GlassFish thì lại có.

#### **2.4.1.4. JPQL là gì ?**

#### **2.4.1.5. Mô hình MVC với JSP/Servlet**

Đây là mô hình thực tế áp dụng tiếp cận theo đúng khái niệm đã nêu ra ở trên, tách biệt riêng biệt thành từng thành phần, tạo nên sự uyển chuyển khi vận dụng và cài đặt, đặc biệt là bảo trì nâng cấp.

Cơ chế thực hiện:

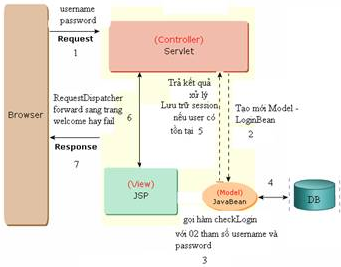
- Web Browser gửi request đến server thông qua các control trên form HTML hay JSP, hay query string Url hay qua cookies.

- Servlet – Controller đón nhận request và xác định Model tương ứng để tạo ra instance của JavaBean để đón nhận các giá trị nhập từ request để lưu trữ và xử lý.

- Model thực hiện xử lý, kết nối dữ liệu vật lý dưới DBMS (nếu có) và trả kết quả trả về cho Controller.

- Kết quả xử lý được chuyển vào Servlet – Controller, Servlet Controller thực hiện tạo hay lựa chọn View để từ đó đưa kết quả xử lý hay dữ liệu lấy từ Model để cập nhật lại trang kết quả View.

- Controller gửi View qua response cho người dùng để browser có thể trình bày dữ liệu trong Web Browser.



Hình 11. Cơ chế thực hiện của Servlet, JSP trong MVC

Ưu điểm của cơ chế này:

- Tính tin cậy (reliability): việc chia từng phần riêng biệt giúp chúng ta sửa đổi từng thành phần riêng biệt, không ảnh hưởng, có thể thay thế thành từng phần tương đương, có thể chia công việc theo nhóm, biên dịch độc lập, tăng cường khả năng tích hợp với khả năng đúng đắn cao.

- Tính tương thích cao (compatibility): có khả năng sử dụng các loại công nghệ khác nhau không lệ thuộc vì chúng ta đã được tách biệt và khái niệm cho từng loại thành phần riêng biệt.

- Tính tái sử dụng (reusable): chúng ta có thể sử dụng các thành phần chia cắt lại trong các ứng dụng khác hay sử dụng lại nhiều lần trong cùng một ứng dụng, tăng tính hiệu quả trong lập trình.

- Khả năng triền khai nhanh chóng và bảo trì nhanh chóng (quick deploy and easy maintenance): vì các thành phần độc lập với nhau.

**2.4.1.5.1. Vai trò của Servlet trong MVC**

Servlet là đoạn chương trình java thực thi trên Web Server hỗ trợ người lập trình Java xây dựng trang web động mà không cần học ngôn ngữ lập trình web mới

Servlets nhận request – yêu cầu từ client, sao đó thực hiện các yêu cầu xử lý để gửi response – phản hồi đến người dùng sử dụng HTTP

Servlet được load sẵn ở Web Server duy nhất lần đầu tiên khi ứng dụng được deploy và đáp ứng tức thời yêu cầu của người dùng thông qua Web Container. Người dùng không lo lắng đến chuyện khởi tạo servlet (như cách chúng ta phải dùng lệnh new đối với việc tạo ra một instance mới cho một object).

Servlet được server hỗ trợ cơ chế multithread giúp giảm tài nguyên và quá tải trong việc xử lý của server hay container

Khuyết điểm của Servlet trong mô hình này:

- Không thích hợp cho việc trình bày và xử lý giao diện vì code html được viết trong chuỗi String của các câu lệnh Java, rất khó trong việc checking và kiểm tra lỗi về tính đúng đắn của văn bản xml (well-form)

- Không hỗ trợ đầy đủ các thành phần liên quan đến session.

- Về phía người dùng có thể nói tương tác chỉ là single thread vì người dùng không thể xác định instance servlet phục vụ cho mình mà container và server tự động xác định instance tương ứng và yêu cầu nó xử lý. Chúng ta chỉ biết được xử lý khi thấy kết quả được hiển thị ở browser. Nghĩa là mọi thứ xử lý phải lệ thuộc container.

Cơ chế hoạt động của Servlet trong MVC cũng tương tự như cơ chế hoạt động bình thường của Servlet:

- Khi có request từ client gửi đến Server hay Web Container.

- Container sẽ lựa chọn một instance Servlet tương ứng để đáp ứng request đó (người dùng sẽ không bao giờ biết instance nào được lựa chọn, nó lựa chọn khi nào, servlet xử lý khi nào).

- Servlet lựa chọn sẽ thực hiện xử lý và kết nối DB nếu cần.

- Sau khi servlet thực hiện xong, sẽ gửi kết quả ra container để gửi response về cho người dùng.

- Browser đón nhận kết quả và trình bày ra màn hình dữ liệu

- Khi một ứng dụng chưa được deploy vào trong server thì servlet chưa được khởi tạo (uninstantiated).

- Khi ứng dụng được deploy vào server thì container sẽ thực hiện khởi tạo instance cho servlet, trong lúc khới tạo kích hoạt hàm init. Lưu ý, hàm init chỉ được kích hoạt lần duy nhất khi deploy.

- Khi có một request đến servlet của người dùng, container đón nhận request và chọn instance bean bất kỳ tương ứng với yêu cầu để đáp ứng.

- Sau khi chọn được bean, container sẽ kích hoạt hàm service tương ứng. Lưu ý ở đây, ở lần yêu cầu thứ 2 thì container cũng sẽ chọn bean và kích hoạt hàm service tương ứng, do vậy, hàm service sẽ được gọi rất nhiều lần.

- Khi servlet được cập nhật mới hay server bị crash hay undeploy ứng dụng ra khỏi server – container, thì hàm destroy của servlet được kích hoạt. Nghĩa là servlet chỉ kích hoạt phương thức destroy lần cuối cùng duy nhất.

Để tạo một servlet chúng ta phải implements HttpServlet. HttpServlet được kế thừa từ GenericServlet. GenericServlet được kế thừa từ phần chính yếu là Servlet, đây là interface cho tất cả servlet và định nghĩa một trong ba phương thức đã được định nghĩa trong chu kỳ sống (init, services, destroy). GenericServlet ngoài được kế thức từ Servlet, nó còn được kế thừa từ ServletConfig, java.io.Serializable (tất cả các lớp đều thuộc package javax.servlet)

- Khi servlet chấp nhận lời gọi từ client, nó sẽ đón nhận 02 tham số là ServletRequest (đối tượng chứa đựng dữ liệu được truyền từ client đến server) và ServletResponse (đối tượng chứa đựng dữ liệu được truyền từ server đến client)

- Khi servlet áp dụng protocol HTTP để giao tiếp thì các thành phần mở rộng từ 02 lớp trên tương ứng được cung cấp đó là HttpServletRequest và HttpServletResponse

- Servlet định nghĩa 3 tầm vực thao tác: request, session, ServletContext. Đây là vùng không gian bộ nhớ (memory segment) được cung cấp cho mỗi ứng dụng web dùng để chứa các thông tin để giao tiếp với các thành phần khác trong server. Mỗi vùng không gian này tồn tại trong một khoảng thời gian nhất định tùy theo qui định:

+ request: tồn tại từ lúc gửi request cho đến khi response

+ session: một khoảng thời gian từ lúc mở trình duyệt đến đóng trình duyệt, hết thời gian session, session bị hủy, …

+ ServletContext: có thể gọi là application tồn tại từ lúc bắt đầu ứng dụng đến khi ứng dụng bị undeploy ra khỏi server hay server bị crash

- Servlet cung cấp thêm một interface RequestDispatcher để hỗ trợ việc giao tiếp và xác định view tương ứng trong xử lý. RequestDispatcher hỗ trợ container chuyển request object từ đối tượng của server từ thành phần này sang thành phần khác (đây là ưu điểm vượt trội so với response.sendRedirect hay click một link trên trang web vì 02 đối tượng này không truyền object request đi).

- Đối tượng cuối cùng là đối tượng sẽ reponse kết quả trả về hay cho phép nhúng đối tượng này sang đối tượng khác.

Cơ chế này còn giúp che dấu thông tin xử lý của các đối tượng xử lý trên thành url của trình duyệt – đảm bảo tính bảo mật cao.

**2.4.1.5.2. JavaBean trong MVC**

Một thành phần cấu thành object và chứa đầy đủ đặc tính của object đó là một object bao gồm state – trạng thái và behaviors – các hành vi, đảm bảo 4 tính chất:

- abstraction (mang tính chất chung nhất của object).

- encapsulation (cho phép người dùng truy cập các trạng thái của object thông qua các behavior).

- hierarchy (có tính kế thừa).

- modularity (phân chia module theo từng nhóm chức năng và tách biệt các thành phần theo dạng component để có thể dễ dàng cài đặt, maintenance – bảo trì, và tái sử dụng – reusable).

JavaBeans là một đối tượng đại diện cho object và được sử dụng như Model bởi vì nó chứa đầy đủ các yêu cầu đã nêu trên.

Đặc điểm của JavaBeans:

- JavaBeans là một java class được implements từ Serializable vì đây là một object sử dụng qua protocol và để giao tiếp với các thành phần trong và ngoài server, do vậy nó phải được chuyển đổi từ thành dạng byte stram để dễ dàng truyền đi

- JavaBeans bắt buộc phải được cài đặt có package để có thể tái sử dụng thông qua lệnh import.

- Các thuộc tính properties trong JavaBeans bắt buộc phải được khai báo là private và các việc khai thác các thuộc tính này phải được thông qua các hàm getTênThuộcTính hay setTênThuộcTính (encapsulate)

- Hàm get sẽ được đổi thành hàm is nếu kiểu dữ liệu là kiểu boolean

- Bắt buộc phải có một contrucstor không tham số để có thể khởi tạo object mà không cần khởi tạo giá trị ban đầu cho object luôn luôn ở trạng thái đảm bảo thao tác không bị lỗi kể cả khi giá trị thuộc tính của các object chưa cập nhật gì cả

- Cài đặt đầy đủ các phương thức hay hành vi mà JavaBeans cần giao tiếp với thế giới bên ngoài. Lưu ý: hàm thiết kế phải tuân theo chuẩn đó là hàm chỉ truyền tham số khi các giá trị này không thể tồn tại trong các thuộc tính của chính instance JavaBeans mà chúng ta đang thao tác – thiết kế.

**2.4.1.5.3. JSP trong MVC**

JSP tích hợp bao gồm HTML, XML, Java Code, và kể cả Servlet nghĩa là nó tạo thuận lợi cho người dùng trong việc xây dựng giao diện – khắc phục nhược điểm của servlet về giao diện.

Ngoài ra, nó cho người dùng mở rộng khả năng sử dụng JSP qua việc định nghĩa các tag mới như XML – khắc phục nhược điểm của HTML.

Đặc biệt cho phép người dùng sử dụng nhúng trực tiếp code Java vào trong JSP thông qua Declaration – khai báo biến và hàm, Scriptlets – chứa code trực tiếp của Java và Expression – tính toán biểu thức và in kết quả ra màn hình.

Bản chất của JSP là Servlet, do vậy các thành phần của Servlet sẽ có tồn tại hết trên JSP.

Ngoài ra, JSP không cần phải biên dịch mà nó được biên dịch khi có request lần đầu tiên yêu cầu đến server, do vậy JSP khắc phục nhược điểm chỉnh sửa phải cần có source code và biên dịch lại khi deploy sau khi chỉnh sửa của Servlet.

JSP cung cấp các thành phần implicit Object để người dùng có thể sử dụng các thành phần tương tác trên server mà không cần khai báo và khởi tạo.

Chu kỳ sống của JSP trong mô hình MVC:

- Khi có một yêu cầu từ client đến server, container xác định trang jsp được yêu cầu.

- Trang JSP được đưa qua JSP Engine để xử lý. JSP Engine thực hiện các bước sau.

+ Đọc cấu trúc file của JSP File từ trên xuống dưới, từ trái qua phải để chuyên đổi (Parsing) sang Java code tương ứng

+ Phát sinh Servlet từ nội dung parsing ở bước trên để cấu tạo thành servlet

+ Thực hiện biên dịch code Servlet

+ Sau khi biên dịch thành công thì quá trình hoạt động sẽ thực hiện đúng theo chu kỳ sống của Servlet như đã nêu trong các phần trên.

+ Tương tự 03 phương thức init, services, destroy của Servlet thì JSP sẽ có 03 phương thức tương ứng là jspInit, jspService và jspDestroy.

- JSP định nghĩa ra 4 tầm vực thao tác đó là page – tồn tại trong page, request – tồn tại từ lúc bắt đầu gửi request cho đến khi response thành cồng, session, application (từ khi deploy đến khi kết thúc ứng dụng) – ServletContext.

Để trang JSP mang tính chất đặc thù của View thì Java Sun đưa ra các thành phần cải tiến để tránh việc trộn lẫn code và xử lý giao diện và tạo tính đặc thù của giao diện. Chúng đó là:

- EL expression language

+ Cách viết tắt ngắn gọn trên trang JSP và che dấu một phần code java được xử lý truy cập

+ Định nghĩa các tầm vực truy cập là requestScope, pageScope, sessionScope, applicationScope

+ Cú pháp: ${trị hay biểu thức hay hằng số}

- JSTL: JSP Standard Tag Library định nghĩa ra các tag hỗ trợ chức năng xử lý trên trang JSP một cách đơn giản và rõ ràng. Chúng gồm các tag như core, sql, fmt – format, xml, function.

### **2.4.2. Tổng quan về EJB 2.0, JDBC**

**2.4.2.1. Tổng quan về EJB**

Enterprise JavaBeans(EJB) – là một thành phần trong kiến trúc J2EE (Java 2 platform Enterprise Edition). EJB là một nền tảng cho phép xây dựng các thành phần phần mềm có tính di động, có khả năng sử dụng lại cao. EJB giúp chúng ta có thể xây dựng, triển khai các ứng dụng phân tán trở nên dễ dàng hơn.

Đặc tả EJB hướng tới việc xây dựng các ứng dụng thương mại (enterprise), lớn, phân tán. EJB đưa ra kiến trúc và các đặc tả cho việc phát triển và triển khai các thành phần (component) phía server của các ứng dụng phân tán. Các thành phần này có thể được xây dựng bởi chính tổ chức phát triển ứng dụng hoặc sử dụng hay mua lại từ một bên thứ ba.

Các EJB được chứa trong một trình chứa EJB (EJB container). Các EJB container này sẽ lưu trữ, lập kế hoạch và quản lý các EJB. EJB container quản lý mọi mặt của một ejb khi nó chạy, bao gồm cả việc truy cập tới các bean, bảo mật, duy trì dữ liệu, trạng thái, quản lý giao dịch, xử lý tương tranh, đồng thời và quản lý truy cập tới tài nguyên của hệ thống.

Đặc tả EJB đã được triển khai bởi nhiều hãng phát triển phần mềm hàng đầu như IBM, Sun, BEA System… Tất cả đều tuân theo những đặc tả mà Sun đưa ra nên các thành phần được xây dựng trên nền tàng của hãng này hoàn toàn có thể tương thích và chạy được trên nền tảng của một hãng khác.

Một EJB được chia làm 3 loại:

- Entity beans: tương tự với các đối tượng thực thể (entity object) trong đặc tả UML [Jacobson/UML]. Entity bean chứa các thông tin tác vụ và các phương thức hoạt động của nó. Một entity beans có thể được lưu trữ và khôi phục lại một các tự động bởi trình quản lý ejb, với việc sử dụng cơ sở dữ liệu quan hệ hay một hệ thống lưu trữ nào đó do bạn chọn.

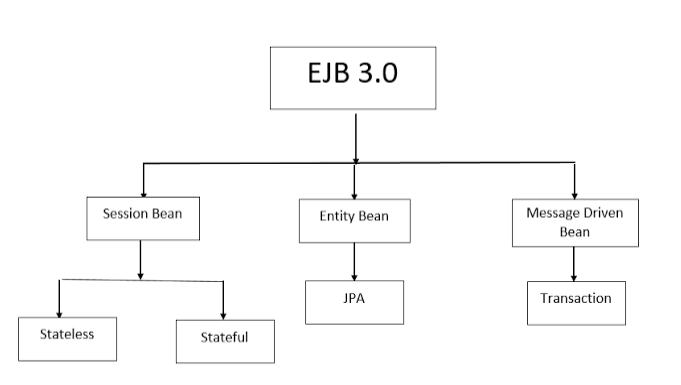
- Session beans: quản lý các nhiệm vụ tác nghiệp của các client của ejb. Client gửi một message tới session beans thông qua môi trường mạng. Session bean có thể triệu gọi tới một entity bean hay một session bean khác để hoàn thành nhiệm vụ mà client yêu cầu. Session bean lại được chia làm hai loại:

+Stateless: trạng thái các giao dịch của client sẽ không được lưu lại cho những lần giao dịch sau. Những lần giao dịch khác nhau sẽ hoàn toàn động lập với nhau.

+ Statefull: trạng thái các giao tác của client sẽ được lưu trữ lại phục vụ cho những lần giao dịch sau.

- Message-driven beans: chịu trách nhiệm điều khiển các message giữa client và server. Message này sẽ được truyền theo hình thức P2P hay broadcast (quảng bá)….

Mô hình EJB



**2.4.2.2 Tổng quan về JDBC**

JDBC, là viết tắt của Java Database Connectivity, là một Java API chuẩn để kết nối giữa ngôn ngữ lập trình Java và các cơ sở dữ liệu đa dạng. Sử dụng JDBC, bạn có thể thực hiện nhiều tác vụ đa dạng khi làm việc với cơ sở dữ liệu như tạo, xóa cơ sở dữ liệu; tạo và thực thi các lệnh SQL hoặc MySQL; tạo, xóa các bản ghi; …

**2.4.2.2.1. Tại sao sử dụng JDBC?**

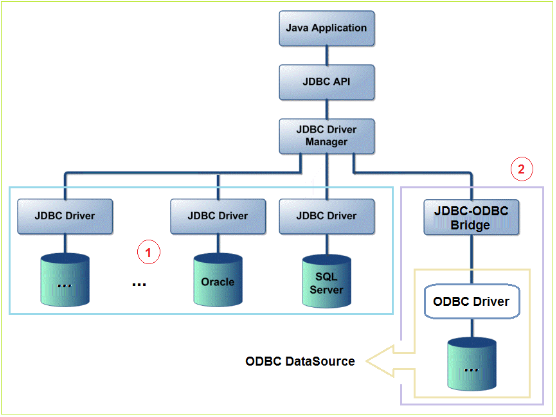
- Trước khi có JDBC, ODBC API được sử dụng là API để kết nối và thực thi truy vấn với cơ sở dữ liệu. Nhưng API này có một hạn chế rõ ràng là nó sử dụng ODBC driver được viết bằng ngôn ngữ C (là phụ thuộc nền tảng và không an toàn).

- Do đó, Java đã định nghĩa riêng API (là JDBC API) để thực hiện các thao tác khi làm việc với cơ sở dữ liệu, và tất nhiên các JDBC driver này được viết bằng ngôn ngữ Java để cung cấp các lợi thế rõ rệt của ngôn ngữ này.

- API, là viết tắt của Application Programming Interface, là một tài liệu gồm sự mô tả về tất cả các đặc điểm của một sản phẩm hoặc phần mềm. Nó biểu diễn các Class và Interface mà các chương trình phần mềm có thể theo sau để giao tiếp với nhau. Một API có thể được tạo cho các ứng dụng, các thư viện, các hệ điều hành, …

**2.4.2.2.2. Các thành phần của JDBC:**

JDBC API cung cấp các Class và Interface sau:



Hình 12. Các thành phần của JDBC

- DriverManager: Lớp này quản lý các Database Driver. Ánh xạ các yêu cầu kết nối từ ứng dụng Java với Data driver thích hợp bởi sử dụng giao thức kết nối phụ.

- Driver: Interface này xử lý các kết nối với Database Server. Hiếm khi, bạn tương tác trực tiếp với các đối tượng Driver này. Thay vào đó, bạn sử dụng các đối tượng DriverManager để quản lý các đối tượng kiểu này.

- Connection: Đối tượng Connection biểu diễn ngữ cảnh giao tiếp. Interface này chứa nhiều phương thức đa dạng để tạo kết nối với một Database.

- Statement: Bạn sử dụng các đối tượng được tạo từ Interface này để đệ trình các lệnh SQL tới Database. Ngoài ra, một số Interface kết thừa từ nó cung chấp nhận thêm các tham số để thực thi các thủ tục đã được lưu trữ.

- ResultSet: Các đối tượng này giữ dữ liệu được thu nhận từ một Database sau khi bạn thực thi một truy vấn SQL. Nó nóng vai trò như một Iterator để cho phép bạn vọc qua dữ liệu của nó.

- SQLException: Lớp này xử lý bất cứ lỗi nào xuất hiện trong khi làm việc với Database.

**2.4.2..2.3. Các loại JDBC Driver**

Có 4 loại JDBC driver. Thông dụng nhất và cũng là hiệu quả nhất là loại 4. Sau đây là mô tả:

- JDBC Driver loại 1- Chúng là các trình điều khiển cầu nối JDBC-ODBC. Chúng ủy nhiệm công việc truy cập dữ liệu cho ODBC API. Chúng là trình điều khiển chậm nhất trong số còn lại. SUN cung cấp một phần mềm trình điều khiển JDBC/ODBC.

- JDBC Driver loại 2 – Chúng chủ yếu sử dụng API mã nền để truy cập dữ liệu và cung cấp các lớp bao Java để có thể được gọi ra bằng cách dùng các JDBC driver.

- JDBC Driver loại 3 – Chúng được viết thuần bằng Java và sử dụng giao thức Net độc lập nhà sản xuất để truy cập đến trình theo dõi từ xa độc lập nhà sản xuất. Trình theo dõi này đến lượt nó lại ánh xạ các lời gọi độc lập nhà sản xuất này vào các lời gọi phụ thuộc nhà sản xuất. Bước đặc biệt này đã làm tăng độ phức tạp và giảm tính hiệu quả trong truy cập cơ sở dữ liệu.

- JDBC Driver loại 4 – Chúng được viết thuần túy bằng Java và là loại hiệu quả nhất. Chúng cho phép kết nối trực tiếp vào cơ sở dữ liệu, cung cấp kết quả tối ưu và cho phép lập trình viên thực hiện các chức năng tùy thuộc vào cơ sở dữ liệu cụ thể. Điều này đã tạo ra tính cơ động cao nhất là khi bạn cần thay đổi cơ sở dữ liệu bên dưới một ứng dụng. Loại driver này thường được dùng cho các ứng dụng phân tán cao.

### **2.4.3. Tổng quan về Java Persistence API, Hibernate**

**2.4.3.1. Tổng quan về Java Persistence API**

Java Persistence API hay JPA là một đăc tả Java cho việc ánh xạ giữa các đối tượng Java tới cơ sở dữ liệu quan hệ sử dụng cộng nghệ phổ biến là ORM( Object Relational Mapping). JPA API cung cấp đầy đủ các công cụ cho phép người lập trình có thể tạo cơ sở dữ liệu một cách đơn giản và nhanh chóng. JPA API có thể dùng để persist một đối tượng business (POJO) vào trong cơ sở dữ liệu hoặc lấy dữ liệu từ cơ sở dữ liệu và ánh xạ ra các đối tượng business một cách đơn giản.

Bất kỳ một ứng dụng doanh nghiệp (Enterprise) nào cũng đều cần phải thực hiện các tháo tác (CRUD) đến database bằng việc lưư trữ, trích rút lượng lớn dữ liệu. Các nhà phát triển (developers) cần phải viết nhiều code để thực hiện các việc này.

Với công nghệ JDBC trước đây, developers sẽ phải chủ động thực hiện các việc: từ mở kết nối vào Database, tạo các Statement, ResultSet cho đến đóng tất cả các thứ đó lại, và code sẽ trở nên cồng kềnh. Developers cần phải làm việc với cả java code và SQL. Thực tế, từng Database khác nhau thì câu SQL có một số phần khác nhau, nên đòi hỏi developers phải nắm được sự khác biệt này để viết code cho phù hợp. Khi ứng dụng muốn chuyển từ database sang database khác (ví dụ từ Oracle sang MS SQL Server) thì chắc chắn sẽ có một số phần của câu SQL cần phải đổi. Công việc sửa code lại đòi hỏi developers phải test lại ứng dụng. Điều này sẽ tốn thời gian phát triển. Để khắc phục nhược điểm này, đã có rất nhiều frameworks ra đời với mục đích giúp xóa đi vấn để về tương thích giữa các Database, giúp developers tập trung vào phần xử lý nghiệp vụ.

**2.4.3.1.1. JPA là gì?**

JPA là một đặc tả của SunMicrosystem, ra đời cùng với bản đặc tả J2EE 5. JPA không phải là một sản phẩm và không thể nó như thành phần persistence. Nó cần phải có một cài đặt ORM để hoạt động và persist các đối tượng Java. Các framework ORM có thể sử dụng cho JPA như Hibernate, Toplink, Open JPA,…

Ngày nay đa số các nhà cung cấp đều hỗ trợ cài đặt JPA cho framework persistence của họ. Vì vây, người lập trình có thể lựa chọn nhà cung cấp tốt nhất tùy thuộc yêu cầu ứng dụng của mình.

**2.4.3.1.2. Một số lợi ích của JPA.**

- Đơn giản hóa công nghệ cho tầng persistence (tầng dữ liệu)

- Không phụ thuộc vào các framework ORM

- Có nhiều nhà cung cấp hỗ trợ cài đặt JPA

- Dữ liệu có thể được lưu trữ thông qua việc ORM

**2.4.3.1.3. Một số ORM framework hỗ trợ JPA**

– Hibernate

– EclipseLink

– Open JPA.

**2.4.3.1.4. Tại sao chúng ta nên dùng JPA**

- Viết ít code hơn

- Performance tốt

- Độc lập về database

- Không phải làm việc với SQL

- JPA là một đặc tả đã được chuẩn hóa và là một thành phần trong đặc tả EJB 3.0.

- Có nhiều framework ORM miễn phí hỗ trợ có thể dùng để phát triển nhiều loại ứng dụng khác nhau.

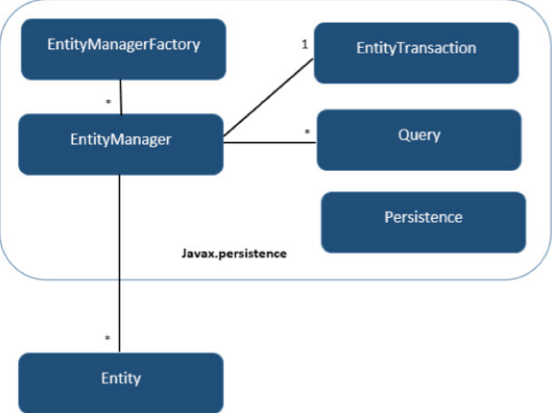
- Ứng dụng xây dựng trên JPA mang tính di động cao

- Có thể sử dụng cho cả ứng dụng J2EE và J2SE

- Hỗ trợ cấu hình triển khai bằng annotation và xml.

**2.4.3.1.5. Kiến trúc JPA.**

JPA sử dụng metadata để ánh xạ các đối tượng persistence với các bảng trong cơ sở dữ liệu. JPA hỗ trợ SQL như là một ngôn ngữ truy vấn để dễ dàng xử lý các truy vấn cơ sở dữ liệu. Ngôn ngữ truy vấn JPA có thể dùng thực thi cả truy vấn tĩnh và truy vấn động.



Hình 13. Kiến trúc JPA

Khái niệm JPA bao gồm ba thành phần chính là Entity, EntityManager, và EntityManagerFactory:

- Entity: là các đối tượng persistence thể hiện một mẫu tin trong cơ sở dữ liệu. Entity chỉ là các lớp POJO đơn giản, dễ dàng tạo. Dưới đây là một số đặc điểm của một Entity:

+ Entity có thể tương tác với cơ sở dữ liệu quan hệ.

+ Entity được xác định thông qua một định danh( tương đương với khóa chình trong table của cơ sở dữ liệu quan hệ)

+ Entity hỗ trợ giao tác (transaction)

+ Entity hỗ trợ kế thừa giống như những lớp thường khác.

- EntityManager: là một giao diện (interface) cung cấp các API cho việc tương tác với các Entity. Một số chức năng cơ bản của EntityManager như:

+ Persist: phương thức này dùng để lưu một thực thể mới tạo vào cơ sở dữ liệu

+ Merge: dùng để cập nhật trạng thái của entity vào cơ sở dữ liệu.

+ Remove: xóa một thể hiện của entity.

- EntityManagerFactory:được dùng để tạo ra một thể hiện của EntityManager.

Các tính năng của JPA: Việc giới thiệu JPA vào trong đặc tả J2EE 5 là một bước tiến lớn trong việc đơn giản hóa các quy trình phát triển ứng dụng. JPA đơn giản hóa mô hình thực thể dữ liệu và cộng thểm một số tính năng mới mà phiên bản EJB trước (EJB 2.0) không có. Giờ đây người lập trình có thể ánh xạ trực tiếp các đối tượng persistence với cơ sở dữ liệu quan hệ. JPA có thể sử dụng bên ngoài container, điều này không dễ thực hiện trong EJB 2.1. Bạn cũng có thể sử dụng JPA trong các ứng dụng swing.

Một số tính năng của JPA:

- JPA hỗ trợ pluggable, tức là có thể sử dụng nhiều nhà hãng cung cấp thứ ba như Hibernate hay Toplink.

- Hỗ trợ annotation

- Giảm bớt số lớp yêu cầu cho việc phát triển persistence.

- Không cần phải viết các mô tả triển khai trong xml. Annotation dựa trên metadata đã hỗ trợ trong các ứng dụng JPA.

- Đã chuẩn hóa ORM và dễ dàng phát triển hơn

JPA hỗ trợ truy vấn động và tĩnh.

- Nhiều IDE hỗ trợ phát triển ứng dụng JPA và có thể tự động sinh code ánh xạ từ cơ sở dữ liệu thành các entity và ngược lại.

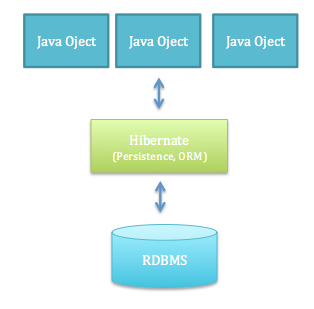
**2.4.3.2. Tổng quan về Hibernate**

Một ứng dụng có thể chia làm 3 phần như sau: phần giao diện người dùng (UI layer), phần xử lý nghiệp vụ (business layer) và phần chứa dữ liệu (data layer). Cụ thể ra, business layer sẽ có thể chia nhỏ thành 2 layer con là business logic layer (chỉ quan tâm đến ý nghĩa của các nghiệp vụ, các tính toán mang nhằm thoả mãn yêu cầu của người dùng) và persitence layer. Persistence layer chịu trách nhiệm giao tiếp với data layer (thường là một hệ quản trị cơ sở dữ liệu quan hệ – Relational DBMS). Persistence layer sẽ đảm nhiệm các nhiệm vụ mở kết nối, truy xuất và lưu trữ dữ liệu vào các Relational DBMS.

Hibernate là một giải pháp ORM cho Java , một dự án open source chuyên nghiệp, một framework cho pesistence layer, và là một thành phần cốt tuỷ của bộ sản phẩm JBoss Enterprise Middleware System (JEMS). JBoss, như chúng ta đã biết là một đơn vị của Red Hat, chuyên cung cấp các dịch vụ 24/7 về hỗ trợ chuyên nghiệp, tư vấn và huấn luyện sẵn sàng hỗ trợ bạn trong việc dùng Hibernate…

Hibernate ánh xạ các lớp Java với các bảng trong cơ sở dữ liệu và ánh xạ giữa các kiểu dữ liệu trong Java với các kiểu dữ liệu SQL. Hibernate giúp giảm thiểu các công việc liên quan đến nhiệm vụ xử lý dữ liệu thông thường trong phát triển ứng dụng.

Hibernate nằm giữa các đối tượng Java truyền thống và cơ sở dữ liệu để giải quyết tất cả các công việc trong lớp persistence dựa trên mô hình kỹ thuật ORM.



Hình 14. Mô hình ORM/Hibernate

Ưu điểm của Hibernate:

- Hibernate chịu trách nhiệm ánh xạ giữa các lớp Java đến các bảng trong CSDL dùng các file XML mà không cần phải viết bất kỳ dòng code nào.

- Cung cấp các API đơn giản để lưu trữ, truy xuất trực tiếp các đối tượng Java và CSDL.

- Nếu có bất kỳ thay đổi nào trong CSDL thì chỉ cần thay đổi file XML.

- Cung cấp đầy đủ các tiện ích, tính năng truy vấn dữ liệu đơn giản, hiệu quả

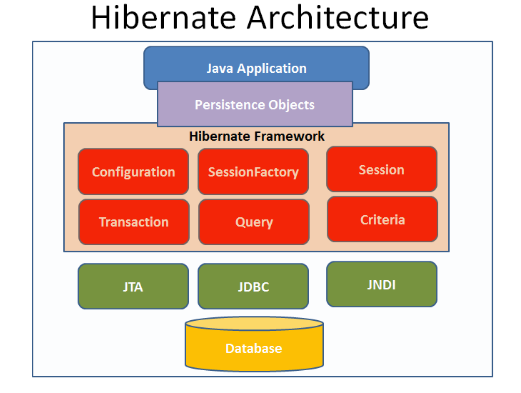
- Thao tác, xử lý được các quan hệ phức tạp của các đối tượng trong CSDL.

- Giảm thiểu sự truy cập đến CSDL đến mức thấp nhất nhờ có chiến lược tìm, nạp thông minh.

Hibernate hỗ trợ hầu hết các RDBMS quan trọng như: HSQL Database Engine, DB2/NT, MySQL, PostgreSQL, Oracle, Microsoft SQL Server Database, …

Hibernate hỗ trợ một số công nghệ sau: XDoclet Spring, J2EE, Eclipse plug-ins, Maven.

Hibernate sử dụng các file cấu hình để cung cấp các dịch vụ và đối tượng persistence cho ứng dụng.



Hình 15. Kiến trúc Hibernate

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

Phần dưới đây đưa ra mô tả ngắn gọn của từng lớp đối tượng tham gia trong kiến trúc ứng dụng Hibernate:

- Configuration Object: Các đối tượng cấu hình là đối tượng Hibernate đầu tiên bạn tạo trong bất kỳ ứng dụng Hibernate và thường được tạo ra chỉ một lần trong thời gian khởi ứng dụng. Nó đại diện cho một tập tin cấu hình, tài sản theo yêu cầu của Hibernate. Các đối tượng cấu hình cung cấp hai khóa các thành phần:

+ Connection: được xử lý thông qua một hoặc nhiều tập tin cấu hình được hỗ trợ bởi Hibernate. Những tập tin này là hibernate.properties và hibernate.cfg.xml .

+ Config Mapping: thành phần này tạo ra sự kết nối giữa các lớp Java và các bảng cơ sở dữ liệu …

- Object SessionFactory: đối tượng cấu hình được sử dụng để tạo ra một đối tượng SessionFactory mà inturn cấu hình Hibernate cho các ứng dụng sử dụng các tập tin cấu hình cung cấp và cho phép cho một đối tượng Session được khởi tạo. SessionFactory là đối tượng quan trọng nên nó thường được tạo ra trong suốt quá trình khởi động và giữ để sử dụng sau. Bạn sẽ cần một đối tượng SessionFactory mỗi khi cơ sở dữ liệu sử dụng một tập tin cấu hình riêng biệt. Vì vậy, nếu bạn đang sử dụng nhiều cơ sở dữ liệu sau đó bạn sẽ phải tạo ra nhiều đối tượng SessionFactory.

- Session Object: Một Session được sử dụng để được kết nối vật lý với một cơ sở dữ liệu. Các đối tượng Session là trọng lượng nhẹ và được thiết kế để khởi tạo mỗi lần tương tác với cơ sở dữ liệu. Đối tượng liên tục được lưu lại và được lấy thông qua một đối tượng Session.Các đối tượng này không nên được giữ mở trong một thời gian dài bởi vì chúng thường không an toàn và chúng cần được phá hủy khi cần thiết.

- Transaction object: Một giao dịch đại diện cho một đơn vị làm việc với cơ sở dữ liệu và hầu hết các RDBMS hỗ trợ chức năng giao dịch. Giao dịch trong Hibernate được xử lý bởi một người quản lý giao dịch cơ bản và giao dịch (từ JDBC hoặc JTA). Đây là một đối tượng tùy chọn và ứng dụng Hibernate có thể chọn không sử dụng giao diện này, thay vì quản lý giao dịch trong mã ứng dụng của riêng mình.

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

- Criteria object: Các tiêu chí đối tượng được sử dụng để tạo ra và thực hiện theo định hướng đối tượng tiêu chí truy vấn để lấy đối tượng.

### **2.4.4. Spring MVC**

Như đã phân tích ở các phần trên, ta đã biết mô hình MVC là gì nên phần này chỉ tập trung vào việc phân tích mô hình MVC trong Spring.

Spring là một framework giúp các nhà phát triển xây dựng những hệ thống và ứng dụng chạy trên JVM một cách đơn giản, tiện gọn, nhanh chóng và mềm dẻo.

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

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

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

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

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

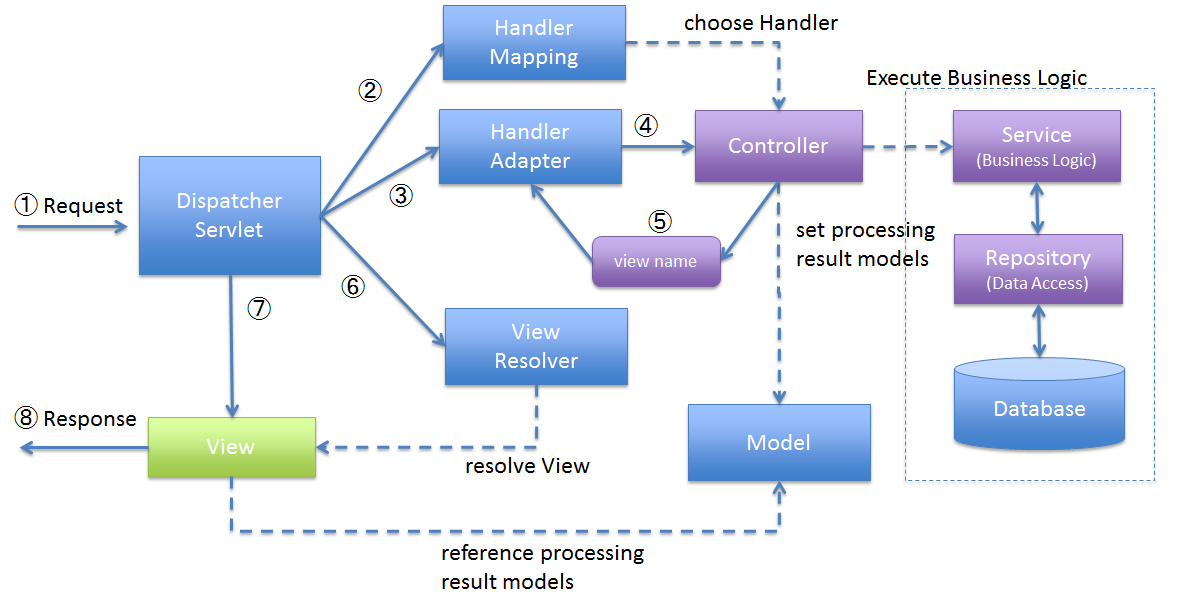
Để ngăn chặn sự phức tạp trong phát triển các ứng dụng, Spring Framework thường dựa trên các quan điểm như sau:

- Đơn giản hóa công việc phát triển thông qua việc sử dụng các đối tượng Java đơn giản hay còn được gọi là POJO (Plain Old Java Object).

- Nới lỏng ràng buộc giữa các thành phần thông qua việc sử dụng Dependency Injection và viết các interface.

- Tiếp cận lập trình khai báo bằng cách sử dụng các quy tắc (convention) và các khía cạnh (aspect) chung.

- Giảm thiểu các mã nghi thức và soạn sẵn (boilerplate) thông qua việc sử dụng các khuôn mẫu (template) và các khía cạnh.



Hình 16. Kiến trúc Spring MVC

Kiến trúc của SpringMVC:

- DispatcherServlet nhận request.

- DispatcherServlet dispatches chọn controller phù hợp đưa vào HandlerMapping. HandlerMapping chọn controller liên kết với yêu cầu đầu vào đường dẫn URL và trả về controller của DispatcherServlet.

- DispatcherServlet dispatches hiện thực business logic của Controller tới HandlerAdapter.

- HandlerAdapter gọi các business logic của Controller.

- Controller hiện thực business logic, thiết lập các kết quả trong Model và trả về view trong HandlerAdapter.

- DispatcherServlet dispatches giải quyết View tương ứng với tên View của ViewResolver. ViewResolver trả về View thông qua View name.

- DispatcherServlet dispatches trả về tiến trình bằng cách trả về View.

- View trả về dữ liệu Model và trả về response.

# **CHƯƠNG 3. PHÂN TÍCH WEB TẠI CÔNG TY DXC TECHNOLOGY**

## **3.1. Quá trình hiện thực Web Java**

### **3.1.1. Phân tích hệ thống**

Mô tả hệ thống:

Xây dựng một hệ thống quản lý hồ sơ bệnh nhân trong bệnh viện, mỗi hồ sơ sẽ được chia thành 2 phần:

- Thông tin cá nhân: Họ và Tên, Địa chỉ, Ngày sinh, Giới tính, Nơi sinh ...

- Hồ sơ điều trị y tế: Lịch sử bệnh, đơn thuốc, thuốc đang sử dụng hoặc mới sử dụng, kết quả xét nghiệm y khoa ...

Các chức năng của hệ thống:

- Chức năng của y tá:

+ Nhận thông tin bệnh nhân ban đầu.

+ Chỉ có thể cập nhật thông tin cá nhân trong hồ sơ bệnh nhân.

+ Tìm kiếm hồ sơ bệnh nhân (tìm kiếm theo thông tin cụ thể như ).

- Các chức năng của bác sĩ:

+ Tìm kiếm và lấy hồ sơ bệnh nhân: Lọc theo tên, bệnh, thuốc.

+ Tạo / Cập nhật hồ sơ điều trị y tế trong hồ sơ bệnh nhân: Thông tin cơ bản, tải kết quả xét nghiệm y khoa (quét hoặc sao chụp đơn thuốc, hình X-Ray ...)

+ Kiểm tra xem một bệnh nhân có dị ứng với một loại thuốc cụ thể.

- Chức năng của admin:

+ CRUD tài khoản của y tá và bác sĩ.

+ Quản lý danh sách thuốc (CRUD)

+ Quàn lý danh sách bệnh nhân.

Ràng buộc của hệ thống:

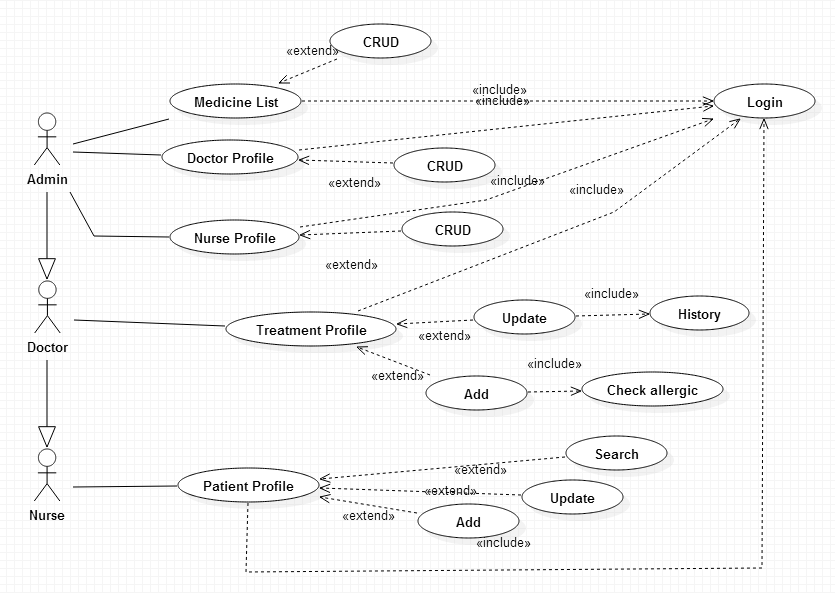
- Mỗi hồ sơ bệnh nhân có thể có một hoặc nhiều hồ sơ điều trị y tế. Kích thước file kết quả kiểm tra sức khỏe giới hạn ở 10 MB. (cụ thể ở đây là hình ảnh chụp X-quang,...)

- Theo dõi lịch sử thay đổi trong hồ sơ điều trị y tế: Bác sĩ nào làm thay đổi, Ngày và thời gian của mỗi thay đổi.

Hệ thống được xây dựng bằng ngôn ngữ Java, theo mô hình SpringMVC,

Hệ thống sử dụng IDE Eclipse và cơ sở dữ liệu là SQL.

#### **3.1.1.1. Thiết kế lược đồ Use Case**

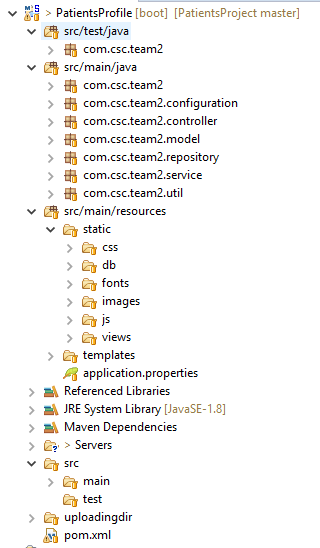


Hình 17. Use Case Diagram

#### **3.1.1.2. Thiết kế cơ sở dữ liệu**

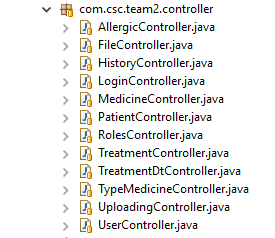
### **3.1.2. Thiết kế giao diện hệ thống**

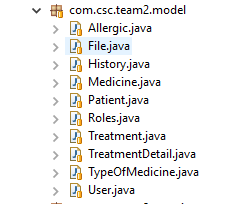
### **3.1.3. Thiết kế phần back-end**

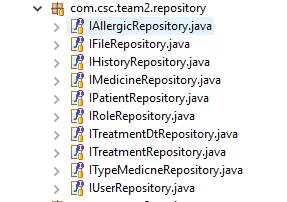


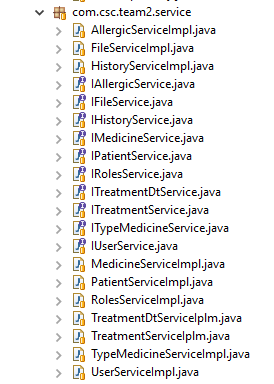
Hình 18. Cấu trúc của hệ thống



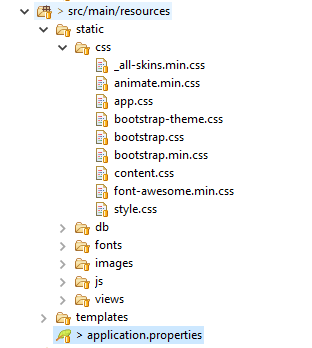


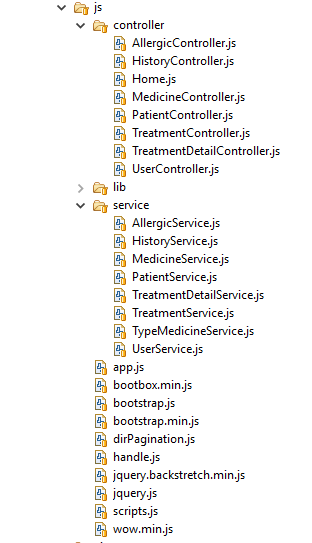


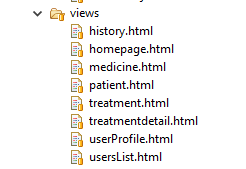


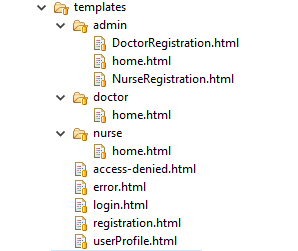


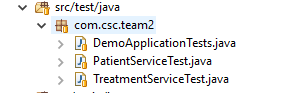












### **3.1.4. Viết Junit cho hệ thống**

## **3.2. Đánh giá kết quả đạt được và hạn chế**

# **TÀI LIỆU THAM KHẢO**

[1] <https://techtalk.vn/10-ly-do-de-hoc-ngon-ngu-lap-trinh-java-va-tai-sao-java-la-tot.html>

[2] <http://blog.vietsol.net/thiet-ke-website/website-la-gi-nhung-khai-niem-co-ban-ve-website.html>

[3] <http://vuongth990.blogspot.co.id/2016/12/java-ee-i.html>

[4] <http://khanhspring.com/tong-quan-ve-hibernate/>

# **PHỤ LỤC**