# **CON ĐƯỜNG PHÁT TRIỂN SỰ NGHIỆP NGÀNH IT**

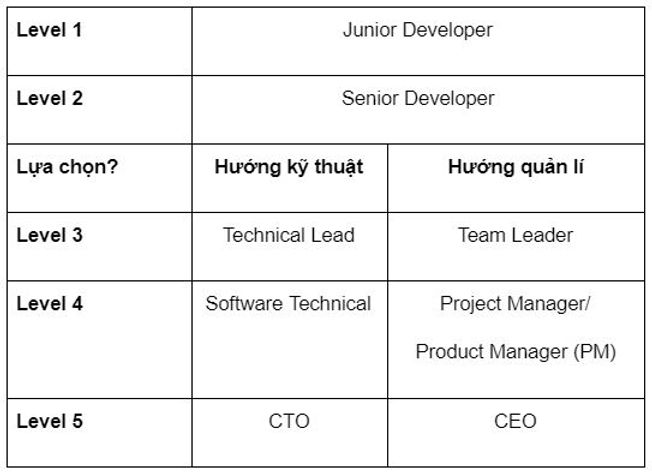
Con đường sự nghiệp (career path) là việc nên tìm hiểu và xác định khi bạn chọn lựa dấn thân vào ngành IT. Để xác định con đường đúng đắn cho mình, hãy chắc chắn về sở trường và khả năng của bạn. Nhìn chung, con đường sự nghiệp của lập trình viên (dev) có thể chia làm 3 nhánh chính:

* Fulltime developer
* Freelancer.
* Entrepreneur.



1. Fulltime Developer

Đây là hướng đi mà khá nhiều developer lựa chọn cho sự nghiệp của mình. Họ ưu tiên chọn những công ty lớn, nhỏ hoặc vừa để làm lấy kinh nghiệm và ăn lương. Một số lựa chọn gắn bó lâu dài để tiến lên các cấp bậc, vị trí cao hơn trong nghề. Số khác, sẽ nhảy việc sang công ty khác hoặc đi theo hai hướng còn lại nếu cảm thấy môi trường làm việc hoặc lương bổng chưa phù hợp với họ. Và dù thế nào, thì việc làm ở công ty giai đoạn đầu trong sự nghiệp vẫn là một hướng đi an toàn nhằm giúp bạn làm quen với môi trường công nghệ hơn. Sau đây là các cấp bậc cụ thể bạn có thể đạt được ở công ty.



Các cấp bậc cụ thể của Full-time Dev

## 1.1. Junior Developer:

Junior Developer hay Fresher là những sinh viên vừa mới ra trường. Họ đang thực hiện quá trình thực tập tại công ty để nạp thêm kinh nghiệm làm việc. Sau khi hoàn thành các khóa học đại học, họ đã được trang bị các kiến thức nền về lập trình, cơ sở dữ liệu, ngôn ngữ lập trình, v.v. Ở công ty, họ sẽ được ứng dụng những lý thuyết đã học này vào thực tế.

Kinh nghiệm cho giai đoạn này:

* Level này thường kéo dài trong khoảng 1 năm.
* Khoan hãy nghĩ đến chính sách lương bổng.
* Vì kinh nghiệm làm việc của bạn còn sơ khai. Vậy nên, môi trường làm việc tốt là yếu tố hàng đầu bạn nên cân nhắc để chọn công ty thực tập.
* Không ngừng học code, ngôn ngữ lập trình và trau dồi một số kỹ năng nghề nghiệp mới.

## 1.2. Senior Developer:

Ở cấp bậc này, bạn đã sở hữu cho mình 1-3 năm kinh nghiệm dày dặn. Bạn có thể xử lý code cho những module phức tạp và tham gia những project công nghệ lớn nhỏ. Đây là giai đoạn thể hiện năng lực để có thể vươn lên những vị trí cao hơn sau này.

Tuy nhiên, công nghệ luôn có xu hướng phát triển rất nhanh. Do đó, bạn cần luôn giữ mình trong tâm thế học hỏi liên tục để cập nhật những công nghệ mới nhất.

Kinh nghiệm cho giai đoạn này:

* Liên tục trau dồi những kỹ năng cứng và mềm để phát triển con đường sự nghiệp developer tốt hơn.
* Tìm hiểu các loại hình công ty khác nhau để chọn ra loại hình phù hợp với bản thân (Product hoặc Outsourcing).

## 1.3. Lựa chọn hướng kỹ thuật:

Đây là hướng sự nghiệp mà bạn phải đảm nhận trách nhiệm về chuyên môn kỹ thuật của công ty. Ở mỗi cấp độ sẽ yêu cầu mức chuyên môn cũng như trách nhiệm khác nhau, tôi sẽ trình bày rõ ràng sau đây:

* **Technical Lead:** Đây là vị trí có thể đạt được sau 3-5 năm làm việc, chịu trách nhiệm kỹ thuật cho toàn team. Yêu cầu của một Technical Lead là cần có kiến thức sâu rộng về công nghệ và hệ thống. Nhằm giúp đưa ra các quyết định như chọn công nghệ hay thiết kế hệ thống thế nào. Đặc biệt, ở một vài công ty, bạn có thể kiêm luôn vị trí Team Leader. Do đó, kỹ năng quản lí con người là một yếu tố quan trọng để vươn lên vị trí này.
* **Software Architect:** Vị trí này có khả năng đạt được khi bạn có từ 5 năm kinh nghiệm trở lên. Tuy nhiên, nó cũng phụ thuộc vào nhiều yếu tố bởi yêu cầu cao của công việc này. Bạn phải làm việc với nhiều bên như khách hàng, PM, Technical Lead. Điều này nhằm đảm bảo hiểu rõ về sản phẩm khách hàng cần từ những yêu cầu của họ đưa ra. Sau đó, tìm ra hướng giải quyết cho sản phẩm, hệ thống từ các cấp bậc khác trong công ty.
* **CTO:** Đây là vị trí quản lí cấp cao về mảng kỹ thuật của công ty. Họ theo sát quy trình sản phẩm để tìm ra những công nghệ mới phù hợp và ứng dụng nó. Làm được điều này, CTO không chỉ có thế mạnh về code mà còn có khả năng quản lí con người để xây dựng một đội ngũ làm việc có năng lực.

## 1.4. Lựa chọn hướng quản lí:

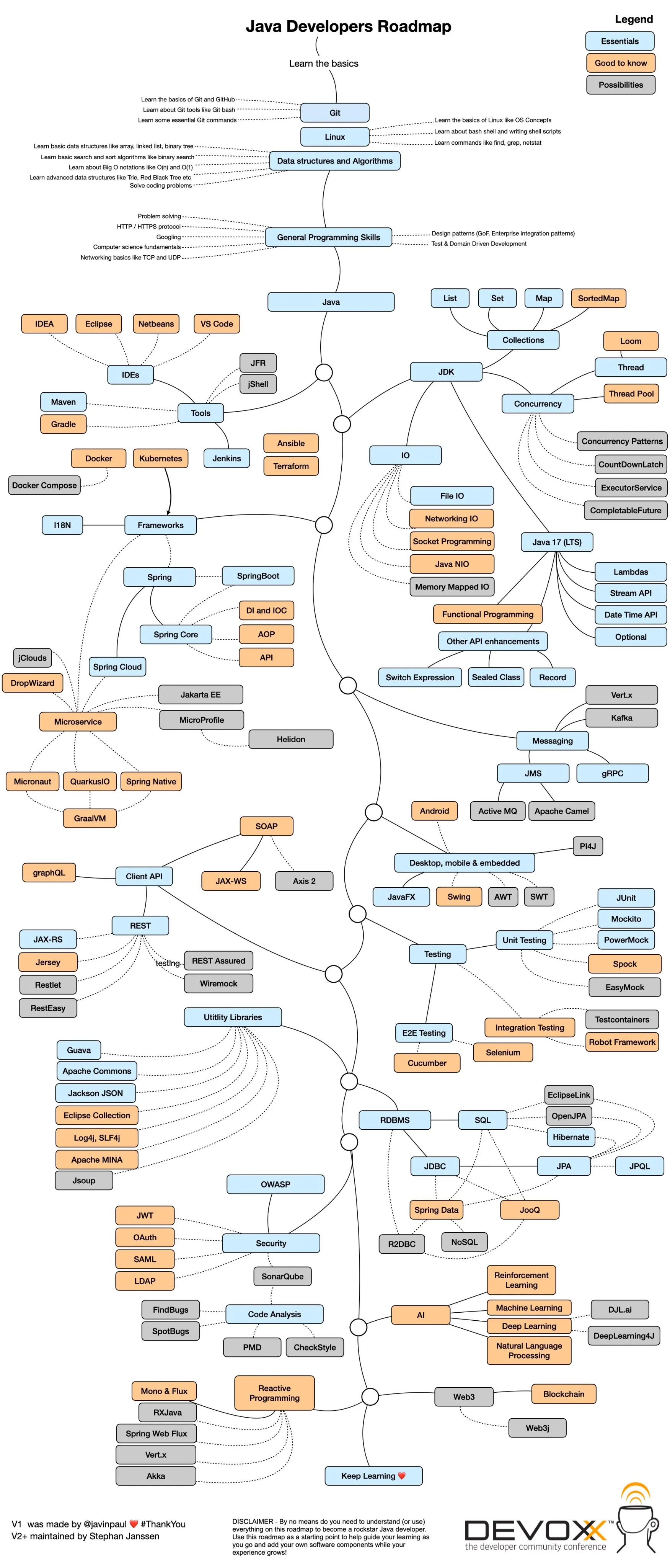
Về hướng quản lí, bạn sẽ đảm nhận nhiệm vụ chính liên quan đến dự án ở mảng con người. Do vậy, nó đòi hỏi những kĩ năng sau:

* **Quản lí dự án:** Lên kế hoạch, quản lí và giám sát kế hoạch nhằm cho thực hiện được những mục tiêu mà dự án đã đề ra.
* **Quản lí thời gian:** Follow các dòng công việc nhằm thực hiện đúng tiến độ và ổn định nhất.
* **Quản lí con người:** Dựa trên tiêu chí “đúng người đúng việc” để điều hướng công việc cũng như khối lượng công việc cho mỗi thành viên một cách hiệu quả. Từ trên những mục tiêu chung, bạn cần biết cách hướng những nỗ lực của mỗi thành viên vào dự án để đạt những mục tiêu đã đề ra đó.
* **Kỹ năng giao tiếp:** Ở những vị trí quản lí, bạn không chỉ giao tiếp để tìm hiểu nhu cầu của khách hàng về sản phẩm mà còn giao tiếp với các members trong team để đôn thúc công việc hoặc tìm giải pháp cho một số vấn đề xảy ra.

Ba cấp bậc bạn sẽ được thăng cấp ở hướng quản lí này là:

* **Team Leader:** Đây là một vị trí như bước đệm để bạn gặt hái được nhiều kinh nghiệm nhằm hướng đến một người quản lí chuyên nghiệp hơn. Bạn sẽ đảm nhận công việc đào tạo cũng như quản lí các thành viên trong một nhóm nhỏ.
* **PM:** Đây là vị trí có nhiệm vụ lên kế hoạch, thực thi cũng như giám sát toàn dự án hoàn thành như mục tiêu và tiến độ đã đề ra.
* **CEO:** Đây là vị trí quản lí cấp cao. Người giám đốc điều hành này có nhiệm vụ lên kế hoạch, tổ chức, giám sát ở cấp độ chiến lược cho tổ chức nhằm đạt được mục tiêu tài chính. Trong công ty, quyền quyết định cuối cùng sẽ được đưa ra bởi CEO.

# **Java programmer roadmap**



# **Kỹ năng bắt buộc đối với bất kỳ nhà phát triển Java nào**

## **Java**

Đây là một số kiến thức cơ bản về Java mà bạn nên biết:

* Cú pháp cơ bản: Java sử dụng cú pháp giống C++ và C#. Bạn cần biết về khai báo biến, kiểu dữ liệu, câu lệnh điều khiển (if, for, while), hàm, lớp và ghi đè (override), cũng như quản lý ngoại lệ (exception handling).
* Hướng đối tượng: Java là một ngôn ngữ lập trình hướng đối tượng. Bạn cần hiểu về các khái niệm như lớp (class), đối tượng (object), kế thừa (inheritance), đa hình (polymorphism) và đóng gói (encapsulation).
* Giao diện (Interface) và lớp trừu tượng (Abstract class): Java hỗ trợ cả giao diện và lớp trừu tượng. Bạn cần hiểu cách sử dụng và khác nhau giữa hai khái niệm này.
* Xử lý ngoại lệ: Java có cơ chế xử lý ngoại lệ mạnh mẽ. Bạn cần biết cách sử dụng câu lệnh try-catch-finally và cách tạo và quản lý các ngoại lệ (exceptions).
* Bộ sưu tập (Collection framework): Java cung cấp một bộ sưu tập mạnh mẽ để làm việc với dữ liệu như danh sách (List), tập hợp (Set) và bản đồ (Map). Bạn nên hiểu về các loại cấu trúc dữ liệu này và cách sử dụng chúng.
* Luồng (Thread): Java hỗ trợ lập trình đa luồng. Bạn cần biết cách tạo và quản lý các luồng (threads) trong Java để xử lý đồng thời (concurrency) và đồng bộ hóa (synchronization).
* Xử lý tệp tin (File I/O): Java cung cấp các lớp và phương thức để đọc và ghi dữ liệu từ và vào tệp tin. Bạn cần hiểu cách sử dụng các lớp như FileReader, FileWriter, BufferedReader và BufferedWriter để làm việc với tệp tin.
* JDBC (Java Database Connectivity): Nếu bạn làm việc với cơ sở dữ liệu, bạn nên biết về JDBC để kết nối và thao tác với cơ sở dữ liệu từ Java.

Đây chỉ là một số kiến thức cơ bản về Java. Java là một ngôn ngữ lập trình rất rộng và có nhiều khía cạnh khác nhau để khám phá.

* Các bạn nên có kiến thức cơ bản về Java, tốt nhất nên học và lấy chứng chỉ Java Oracle Certified Professional: Java SE 11 Developer ( được nhà tuyển dụng đánh giá cao và mức lương cao hơn )

Tài liệu:

* <https://github.com/1995mars/OCP-Java-SE-11>
* Phần Document gồm có OCP.dox là những kiến thức cơ bản giành cho các bạn và OCP\_Oracle\_Certified\_Professional\_Java\_SE\_11\_Developer\_Complete.pdf là sách giúp các bạn học nâng cao hơn và đề chi chứng chỉ.
* Thư mục Dump là chương trình và license để giúp các bạn có thể ôn luyện Java

## **SQL**

SQL (Structured Query Language) là một ngôn ngữ dùng để truy vấn và quản lý cơ sở dữ liệu quan hệ. Nó cho phép bạn tương tác với cơ sở dữ liệu để thực hiện các thao tác như truy vấn dữ liệu, chèn dữ liệu mới, cập nhật dữ liệu hiện có và xóa dữ liệu.

Java web thường sử dụng cơ sở dữ liệu để lưu trữ và quản lý dữ liệu. SQL là ngôn ngữ phổ biến nhất để làm việc với cơ sở dữ liệu quan hệ, và nó được sử dụng rộng rãi trong các ứng dụng web Java để truy vấn và tương tác với cơ sở dữ liệu.

Dưới đây là một số lý do tại sao Java web cần học SQL:

* Truy vấn dữ liệu: SQL cho phép bạn viết các truy vấn để truy xuất dữ liệu từ cơ sở dữ liệu. Trong Java web, bạn cần hiểu cách viết các truy vấn SQL để lấy dữ liệu từ cơ sở dữ liệu và trả về cho ứng dụng web.
* Thao tác dữ liệu: SQL cũng cho phép bạn thực hiện các thao tác khác như chèn (insert), cập nhật (update) và xóa (delete) dữ liệu trong cơ sở dữ liệu. Điều này rất hữu ích khi bạn cần thêm, sửa đổi hoặc xóa dữ liệu từ ứng dụng Java web của bạn.
* Tối ưu hóa truy vấn: SQL cung cấp các cú pháp và chức năng để tối ưu hóa hiệu suất truy vấn. Bằng cách hiểu và sử dụng các cú pháp và chỉ thị tối ưu hóa, bạn có thể cải thiện hiệu suất của ứng dụng Java web của mình.
* Kết nối và tương tác với cơ sở dữ liệu: Java web thường sử dụng JDBC (Java Database Connectivity) để kết nối và tương tác với cơ sở dữ liệu. JDBC sử dụng SQL để thực hiện các truy vấn và thao tác dữ liệu. Bằng cách hiểu SQL, bạn có thể sử dụng JDBC để kết nối và tương tác với cơ sở dữ liệu từ ứng dụng Java web của bạn.

Tài liệu:

* <https://www.oracletutorial.com/oracle-basics/>
* File SQL.pdf mình đã đính kèm

## **Frameworks**

### **Spring Framework**

Đây là một trong những Java Framework phổ biến nhất và theo nghĩa đen, hầu hết mọi ứng dụng Java mà mình đã làm việc trong 5 năm qua đều sử dụng framework này.

Spring Framework khuyến khích viết clean code, dễ kiểm tra và bảo trì hơn bằng cách cung cấp cho bạn các tính năng như Dependency Insert và Inversion of Control. Nó cũng có API phong phú cho hầu hết các công việc hàng ngày và đó là lý do tại sao mọi nhà phát triển Java nên tìm hiểu Spring framework.

### **Spring Boot là gì?**

Spring Boot là một trong số các module của Spring framework chuyên cung cấp các tính năng RAD (Rapid Application Development) cho phép tạo ra và phát triển các ứng dụng độc lập dựa trên Spring một cách nhanh chóng.

Spring Boot ra đời với mục đích loại bỏ những cấu hình phức tạp của Spring, nó không yêu cầu cấu hình XML và nâng cao năng suất cho các nhà phát triển. Với sự góp mặt của Spring Boot, hệ sinh thái Spring đã trở nên mạnh mẽ, phổ biến và hiệu quả hơn bao giờ hết.

### **Spring Data JPA**

JPA đơn giản hơn, gọn gàng hơn và ít tốn công sức hơn là JDBC, SQL hay ánh xạ viết tay. JPA thích hợp cho các ứng dụng phức tạp không tuân theo hiệu suất. Ưu điểm chính của JPA so với JDBC là: trong JPA, dữ liệu được biểu thị bằng các object và class trong khi trong JDBC dữ liệu được biểu diễn bằng các bảng và bản ghi. Nó sử dụng POJO để biểu diễn dữ liệu liên tục giúp đơn giản hóa việc lập trình database. Có một số ưu điểm khác của JPA:

* JPA tránh viết Ngôn Ngữ Định Nghĩa Dữ Liệu (DDL) bằng phương ngữ SQL dành riêng cho database. Thay vào đó nó cho phép ánh xạ trong XML hoặc sử dụng các annotation Java.
* JPA cho phép chúng ta tránh viết Ngôn Ngữ Thao Tác Dữ Liệu (DML) bằng phương ngữ SQL dành riêng cho database.
* JPA cho phép chúng ta save và load các object và đồ thị Java mà không cần bất kỳ ngôn ngữ DML nào.
* Khi chúng ta cần thực hiện các truy vấn JPQL, nó cho phép chúng ta viết các truy vấn dưới dạng các entity Java mà không phải là bảng và cột SQL.

Tài liệu: (bao gồm cả Spring – Spring Boot – Spring Data JPA)

* <https://github.com/1995mars/Spring-Framework>

## **Git & Github**

Git là một hệ thống quản lý phiên bản phân tán (distributed version control system) được phát triển bởi Linus Torvalds, người cũng là người sáng lập hệ điều hành Linux. Git cho phép bạn theo dõi các thay đổi trong các tập tin và dự án phát triển phần mềm, quản lý lịch sử thay đổi, và làm việc cùng nhau trong các nhóm phát triển.

Github là một dịch vụ lưu trữ mã nguồn và quản lý dự án phần mềm trên nền tảng Git. Nó cung cấp một giao diện web cho phép người dùng tạo, sao chép, quản lý và chia sẻ kho lưu trữ (repository) Git. Github cung cấp các tính năng bổ sung như quản lý lỗi (issue tracking), hệ thống wiki và tích hợp dự án trong một môi trường phát triển phần mềm toàn diện.

Java web cần học Git và Github vì những lợi ích sau:

* Quản lý phiên bản: Git cho phép bạn quản lý lịch sử thay đổi của mã nguồn trong dự án Java web. Bằng cách sử dụng Git, bạn có thể theo dõi các thay đổi, so sánh phiên bản, khôi phục phiên bản trước đó và quản lý các nhánh (branches) khác nhau của dự án. Điều này giúp đảm bảo an toàn và dễ dàng trong việc phát triển và triển khai ứng dụng web Java.
* Làm việc nhóm: Git và Github hỗ trợ làm việc nhóm hiệu quả trên dự án Java web. Bạn có thể chia sẻ mã nguồn với các thành viên khác trong nhóm, làm việc đồng thời trên cùng một dự án và hợp nhất (merge) các thay đổi từ các nhánh khác nhau. Điều này giúp tăng tính hiệu quả và giảm xung đột trong quá trình phát triển.
* Theo dõi và giải quyết lỗi: Github cung cấp tính năng quản lý lỗi (issue tracking) cho dự án Java web. Bạn có thể tạo các vấn đề (issues) để theo dõi các lỗi, yêu cầu tính năng và các tác vụ khác. Nhóm phát triển có thể làm việc cùng nhau để giải quyết các vấn đề này. Điều này giúp quản lý và theo dõi quá trình phát triển một cách có tổ chức và tiện lợi.

Tài liệu:

* <https://viblo.asia/p/nhung-lenh-git-co-ban-can-nho-V3m5W1OyZO7>

## **Testing**

**Testing** là một kỹ năng cần thiết đối với bất kỳ nhà phát triển Java nào, đặc biệt unit testing, integration testing, và automation testing. Ở mức tối thiểu, mọi nhà phát triển Java đều phải làm quen với JUnit và Mockito , hai trong số thư viện Unit testing và Mock library phổ biến nhất.

Nếu bạn biết hai điều này và biết cách sử dụng chúng để tạo một bài kiểm thử đơn vị một cách hiệu quả, bạn sẽ trở thành nhà phát triển Java giỏi hơn nhiều so với việc không có chúng.

Tài liệu: Java Test.pdf