**What is a batch? Why do we need batch?**  
  
Spring Batch là một framework xử lý hàng loạt (batch processing) nhẹ và toàn diện được thiết kế để cho phép phát triển các ứng dụng batch mạnh mẽ, quan trọng cho hoạt động hàng ngày của các hệ thống doanh nghiệp. Nó được xây dựng dựa trên các đặc điểm của Spring Framework, giúp các nhà phát triển dễ dàng truy cập và tận dụng các dịch vụ doanh nghiệp nâng cao của nó khi cần thiết.

**Tại sao chúng ta cần Spring Batch?**

Spring Batch giải quyết các thách thức của việc phát triển ứng dụng batch bằng cách cung cấp:

* **Phát triển đơn giản:** Nó cung cấp một tập hợp các thành phần có thể tái sử dụng và một mô hình lập trình nhất quán, giảm nhu cầu viết mã boilerplate và đơn giản hóa quá trình phát triển.
* **Tính mạnh mẽ:** Nó xử lý các mối quan tâm chung về xử lý batch như quản lý giao dịch, khởi động lại job, xử lý lỗi và quản lý tài nguyên, đảm bảo thực thi đáng tin cậy và hiệu quả của các job batch.
* **Khả năng mở rộng:** Nó hỗ trợ nhiều kỹ thuật tối ưu hóa khác nhau như xử lý song song và phân vùng, cho phép xử lý hiệu quả một lượng lớn dữ liệu.
* **Tính linh hoạt:** Nó có thể dễ dàng tích hợp với các thành phần và công nghệ Spring khác như Spring Boot, Spring Data và Spring Integration, cung cấp một giải pháp toàn diện cho nhu cầu xử lý batch của doanh nghiệp.
* **Cộng đồng và hỗ trợ:** Nó là một dự án mã nguồn mở với một cộng đồng lớn các nhà phát triển và người dùng, cung cấp tài liệu, hướng dẫn và hỗ trợ rộng rãi.

**Xử lý tự động, phức tạp một lượng lớn thông tin** hiệu quả nhất khi không có tương tác người dùng. Các hoạt động này thường bao gồm các sự kiện theo thời gian (như tính toán cuối tháng, thông báo, thư tín)

**Tích hợp thông tin nhận được từ hệ thống nội bộ và bên ngoài**, thường yêu cầu định dạng, xác thực và xử lý theo giao dịch vào hệ thống lưu trữ. Xử lý hàng loạt được sử dụng để xử lý hàng tỷ giao dịch mỗi ngày cho các doanh nghiệp

**Spring Batch cung cấp các chức năng có thể tái sử dụng** cần thiết trong xử lý một lượng lớn bản ghi, bao gồm:

* Ghi log và theo dõi
* Quản lý giao dịch
* Thống kê xử lý công việc
* Khởi động lại công việc
* Bỏ qua lỗi
* Quản lý tài nguyên

**Principles and guidelines that we should follow when implementing a batch?**

**Nguyên tắc chung:**

* **Đơn giản hóa:** Tránh xây dựng cấu trúc logic phức tạp trong một ứng dụng batch.
* **Gần gũi dữ liệu:** Xử lý dữ liệu càng gần nơi dữ liệu được lưu trữ càng tốt, hoặc ngược lại (tức là giữ dữ liệu của bạn ở nơi xử lý).
* **Tối ưu hóa tài nguyên hệ thống:** Đặc biệt là hoạt động I/O. Thực hiện càng nhiều thao tác càng tốt trong bộ nhớ trong.
* **Tránh trùng lặp:** Không thực hiện một việc hai lần trong một lần chạy batch. Ví dụ: nếu cần tổng hợp dữ liệu để báo cáo, hãy tăng tổng số được lưu trữ nếu có thể khi dữ liệu được xử lý ban đầu, để ứng dụng báo cáo không phải xử lý lại cùng một dữ liệu.
* **Cấp phát bộ nhớ đủ:** Ngay từ đầu để tránh việc cấp phát lại tốn thời gian trong quá trình xử lý.
* **Đảm bảo tính toàn vẹn dữ liệu:** Luôn giả định trường hợp xấu nhất về tính toàn vẹn dữ liệu. Chèn các kiểm tra và xác thực bản ghi đầy đủ. Triển khai checksum để xác thực nội bộ nếu có thể.
* **Kiểm tra ứng dụng I/O:** (phân tích các câu lệnh SQL) để đảm bảo tránh I/O vật lý không cần thiết. Đặc biệt chú ý đến 4 lỗi phổ biến:
  + Đọc dữ liệu cho mỗi giao dịch khi dữ liệu có thể được đọc một lần và giữ trong bộ nhớ đệm hoặc bộ nhớ làm việc.
  + Đọc lại dữ liệu cho một giao dịch trong khi dữ liệu đã được đọc trước đó trong cùng một giao dịch.
  + Gây ra quét bảng hoặc chỉ mục không cần thiết.
  + Không chỉ định giá trị khóa trong mệnh đề WHERE của câu lệnh SQL.

**Hướng dẫn bổ sung:**

* **Xử lý lỗi và phục hồi:** Triển khai các cơ chế xử lý lỗi mạnh mẽ và giao thức khôi phục để đảm bảo độ tin cậy và khả năng phục hồi của quy trình batch.
* **Ưu tiên các tác vụ quan trọng:** Không làm quá tải hệ thống xử lý batch bằng các tác vụ quá mức.
* **Giám sát thường xuyên:** Triển khai giám sát thường xuyên các quy trình batch để phát hiện và giải quyết các vấn đề tiềm ẩn.

**§ Job là gì? JobInstance, JobParameters, JobExecution và mối quan hệ giữa chúng là gì?**

* **Job:** Một đơn vị công việc hoàn chỉnh, bao gồm một hoặc nhiều bước (step) xử lý dữ liệu. Mỗi Job có một tên duy nhất.
* **JobInstance:** Một lần chạy cụ thể của một Job. Nhiều JobInstance có thể được tạo ra từ cùng một Job, mỗi lần chạy sẽ có các tham số (JobParameters) khác nhau.
* **JobParameters:** Các tham số đầu vào cho một JobInstance. Ví dụ: bạn có thể truyền đường dẫn đến tệp dữ liệu đầu vào dưới dạng JobParameters.
* **JobExecution:** Một lần thực thi cụ thể của một JobInstance. Mỗi JobInstance có thể có nhiều JobExecution, tương ứng với các lần thử lại nếu có lỗi xảy ra.

**Mối quan hệ:**

1. **Job** định nghĩa công việc cần làm.
2. **JobInstance** là một lần chạy của Job.
3. **JobParameters** là các tham số cho JobInstance đó.
4. **JobExecution** là một lần thực thi cụ thể của JobInstance.

**§ Step là gì? StepExecution là gì?**

* **Step:** Một bước xử lý dữ liệu trong Job. Mỗi Step có thể đọc dữ liệu, xử lý dữ liệu và ghi dữ liệu.
* **StepExecution:** Một lần thực thi cụ thể của một Step trong JobExecution.

**§ ExecutionContext là gì?**

* ExecutionContext là một kho lưu trữ dữ liệu (key-value) dùng để chia sẻ dữ liệu giữa các Step trong cùng một JobExecution.

**§ JobRepository là gì?**

* **JobRepository:** Nơi lưu trữ thông tin về trạng thái của Job (JobInstance, JobExecution, StepExecution) và ExecutionContext. Thường được cấu hình để sử dụng cơ sở dữ liệu.

**§ JobLauncher là gì?**

* **JobLauncher:** Đối tượng chịu trách nhiệm khởi chạy một JobInstance với các tham số cụ thể (JobParameters).

**§ ItemReader, ItemWriter và ItemProcessor là gì?**

* **ItemReader:** Đối tượng đọc dữ liệu từ nguồn (ví dụ: tệp CSV, cơ sở dữ liệu) và cung cấp dữ liệu dưới dạng các mục (Item).
* **ItemProcessor:** Đối tượng xử lý từng Item nhận được từ ItemReader.
* **ItemWriter:** Đối tượng ghi các Item đã qua xử lý vào đích (ví dụ: tệp mới, cơ sở dữ liệu).

**§ Mô tả bức tranh toàn cảnh với tất cả những thứ trên.**

1. **JobLauncher** khởi chạy một **JobInstance** với **JobParameters**.
2. **JobRepository** lưu trữ thông tin trạng thái.
3. **JobExecution** bắt đầu.
4. Các **Step** được thực thi tuần tự.
5. Mỗi **Step** sử dụng **ItemReader** để đọc dữ liệu, **ItemProcessor** để xử lý, và **ItemWriter** để ghi dữ liệu.
6. **ExecutionContext** được sử dụng để chia sẻ dữ liệu giữa các Step.
7. **JobExecution** kết thúc, trạng thái được cập nhật vào **JobRepository**.

**§ Làm thế nào để cấu hình một Job? Làm thế nào để chạy một Job?**

* **Cấu hình:** Thường sử dụng các annotation của Spring Batch hoặc tệp cấu hình XML để định nghĩa các bean cho Job, Step, ItemReader, ItemProcessor, ItemWriter, và JobRepository.
* **Chạy:** Sử dụng JobLauncher để khởi chạy Job với các JobParameters cần thiết