**Tìm hiểu về ShedLock**

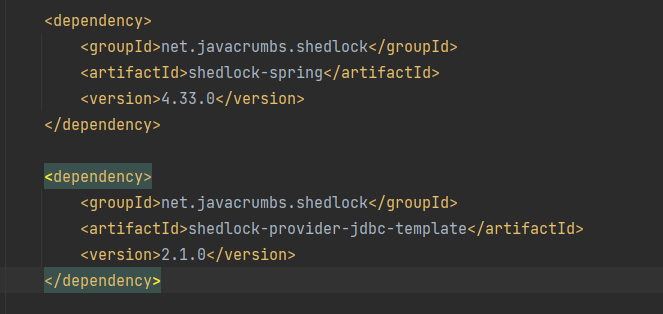
1. **ShedLock là gì?**

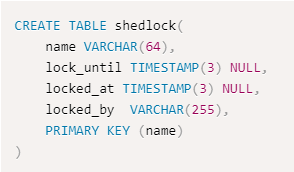
* Shedlock là công nghệ cho phép lên lịch trình tự động cho các công việc trong Spring boot.

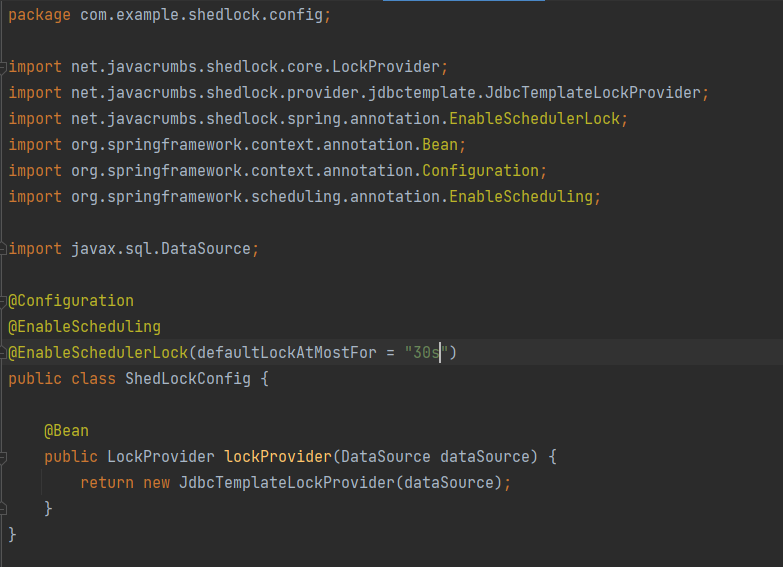
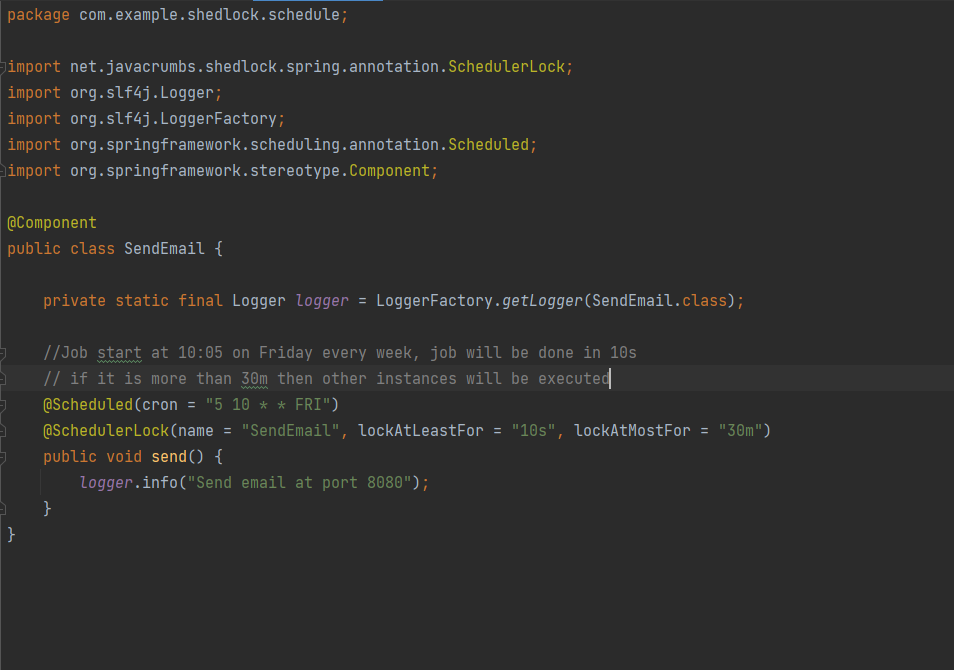
1. **Tại sao lại sử dụng ShedLock?**

* Đôi khi viết chương trình chúng ta sẽ gặp những tình huống thực hiện chức năng chạy theo 1 lịch cố định.
* VD:
  + Cứ 5 phút gửi request 1 lần.
  + Cứ 23h đêm hàng ngày thì thực hiện chạy chức năng backup data.
  + Cứ 7h sáng hàng hàng tuần thì tính toán và gửi báo cáo.
* Có nhiều giải pháp để giải quyết vấn đề trên:
  + Làm bằng tay (không thể thực hiện khi thời gian giữa các lần là quá ngắn và có thể bị quên).
  + Tạo Thread chạy không ngừng nghỉ và kiểm tra thời gian nếu khớp thì chạy method.
  + Sử dụng các TimeTask của Java, các thư viện như Quartz,…
* Như vậy có thể thấy cách thứ 3 là tối ưu và hiệu quả nhất.
* Có nhiều ứng dụng Spring đã sử dụng @Scheduled để thực thi task thường xuyên. Từ việc báo cáo mỗi tối, các công việc dọn dẹp nặng cho đến cơ chế đồng bộ hóa,… và sự đa dạng của các trường hợp sử dụng là rất lớn.
* Khi sử dụng nhiều phiên bản của cùng một ứng dụng Spring Boot đang chạy song song, các công việc đã lên lịch sẽ được thực thi song song.
* Nhưng trong trường hợp báo cáo hoặc đồng bộ hóa, lúc đó ta chỉ muốn thực hiện điều này một lần cho toàn bộ ứng dụng. Theo mặc định, mọi phiên bản sẽ thực thi các tác vụ đã lên lịch, bất kể có phiên bản nào khác đang chạy nó hay không. Điều này có thể dẫn đến dữ liệu không nhất quán hoặc các hành động trùng lặp.
* Spring không cung cấp giải pháp để chạy các tác vụ @Scheduled chỉ trên một phiên bản tại một thời điểm không cần thiết.
* Để giải quyết vấn đề trên thì chúng ta sẽ sử dụng ShedLock.

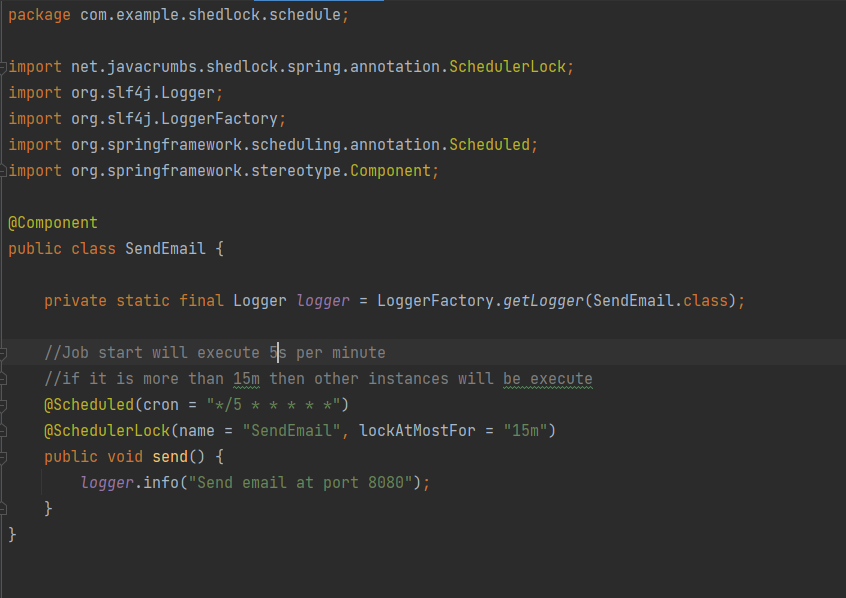
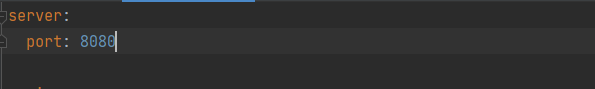
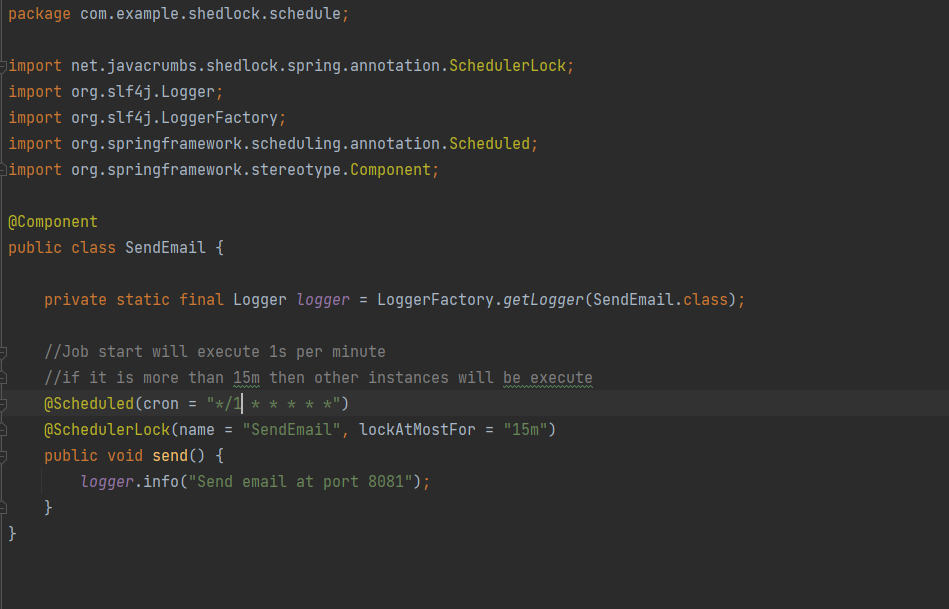
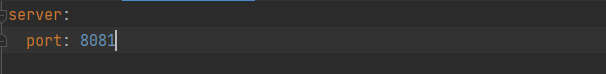
1. **Tích hợp ShedLock trong ứng dụng Spring boot**

* Khởi tạo dự án Spring boot và thêm các thư viện sau: 
* Trong database chúng ta sẽ tạo bảng shedlock với các thuộc tính và kiểu dữ liệu như sau:



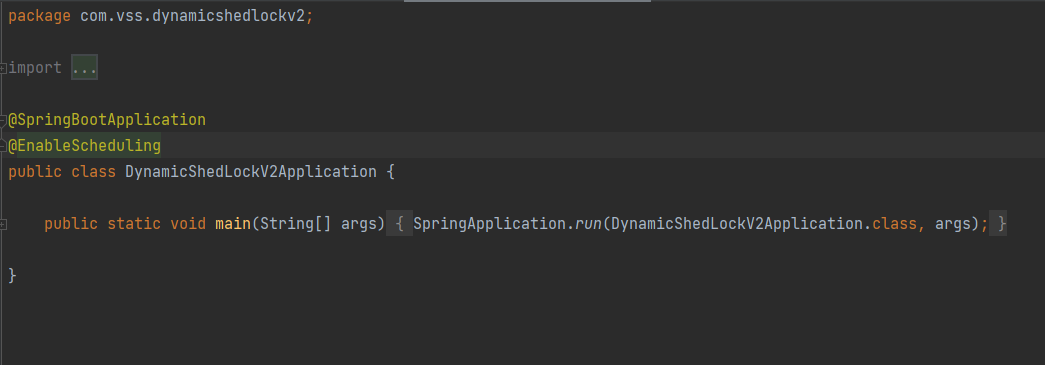
* Tại project Spring boot, tạo package config -> Tạo class ShedLockConfig: 
  + **@EnableSchedulerLock** cho phép chúng ta có thể sử dụng ShedLock trong công việc.
    - Thuộc tính **defaultLockAtMostFor** hoạt động như cấu hình dự phòng khi không chỉ định **lockAtMostFor** một cách rõ ràng.
  + **@EnableScheduling** được sử dụng để đảm bảo rằng background task excutor được khởi tạo. Nếu ứng dụng khởi chạy không được chú thích với @EnableScheduling chúng ta sẽ không thể lập lịch chạy task định kỳ được chú thích bởi @Scheduled.
* Tiếp theo tạo 1 class SendEmail: 
* Tất cả các jobs chúng ta sẽ thêm annotation @SchedulerLock, nếu không các jobs sẽ chạy song song và các thuộc tính như sau:
  + **name**: là duy nhất trong các jobs và là tên của job
  + **lockAtLeastFor:** Thuộc tính này để đảm bảo công việc hiện tại sẽ giữ khóa ít nhất trong khoảng thời gian nhất định. Ví dụ nếu ta cấu hình nó thành 20s thì khóa sẽ không được giải phóng mặc dù job đã hoàn thành trước 20s (ít nhất bao lâu nó sẽ bị khóa).
  + **lockAtMostFor:** Thuộc tính này để đảm bảo khóa được phát hành trong trường hợp instance chết. Nếu một tác vụ mất nhiều thời gian hơn giá trị đã được cài đặt thì những hành vi không mong muốn có thể xảy ra và những instance khác cũng sẽ chạy job vì khóa sẽ được giải phóng (nó sẽ bị khóa trong bao lâu).

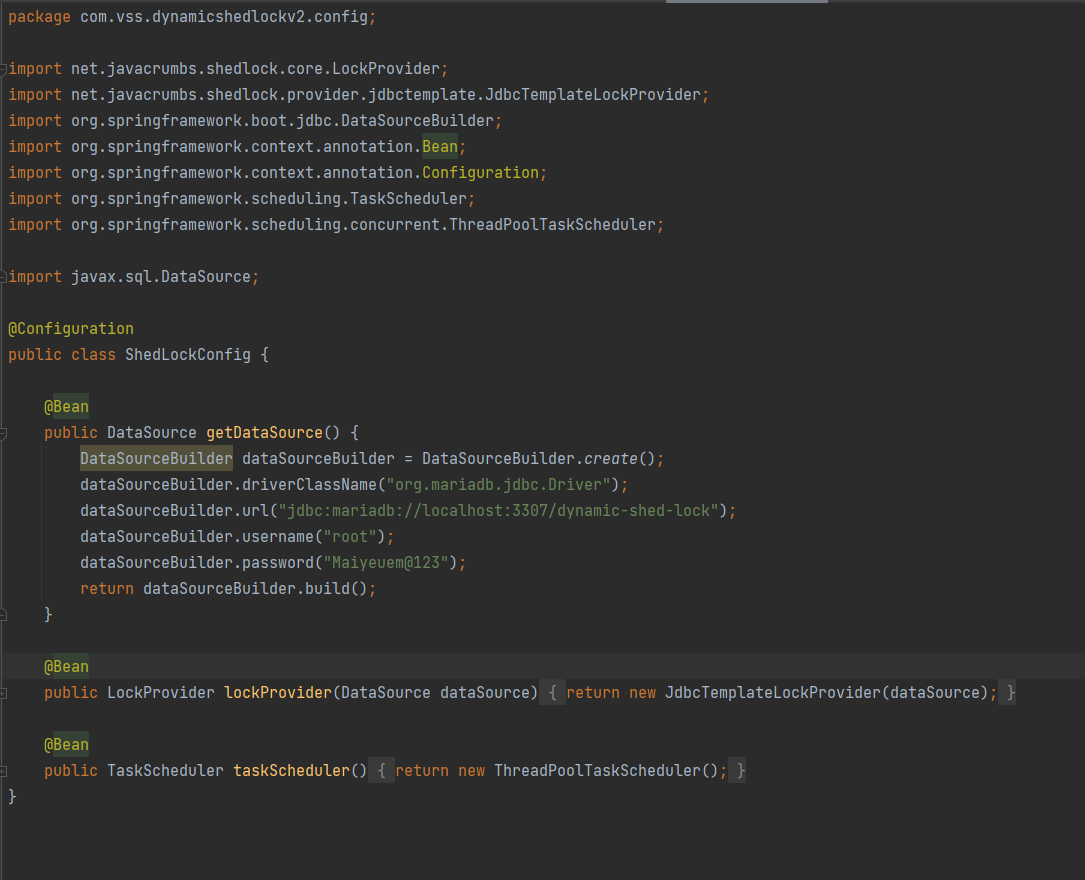
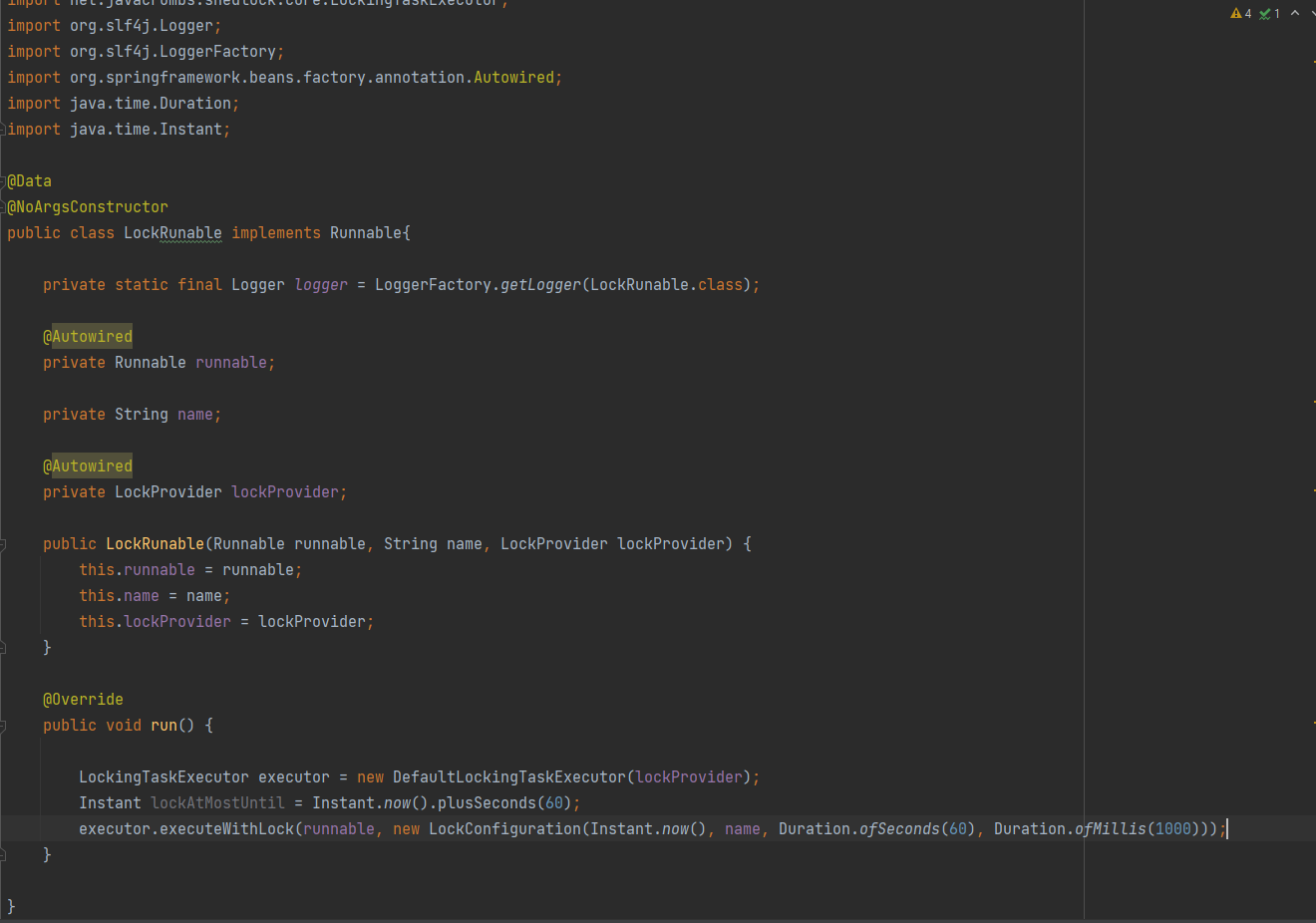
1. **Thực hiện Test:**

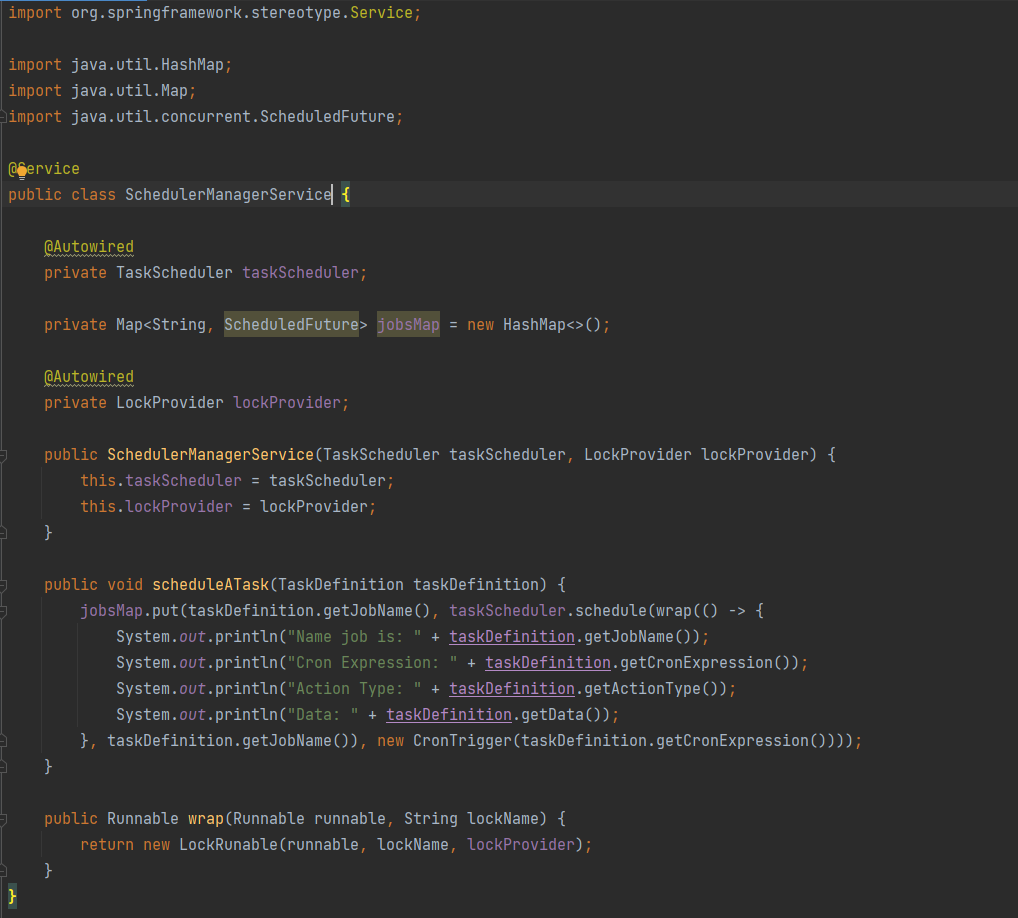
* Đầu tiên chúng ta sẽ build ra file jar với port 8080: 
* Chọn maven bên phải màn hình và chọn install để build ra file jar
* Tiếp theo chúng ta sẽ build ra file jar với port 8081: 
* Chọn maven bên phải màn hình và chọn install để build ra file jar.
* Để chạy file jar vừa build chúng ta di chuyển đến thư mục chứa file và chạy lệnh sau trong màn hình cmd:
  + java -jar shed-lock-0.1.1-SNAPSHOT-8080.jar

1. **Dynamic Shed Lock:**

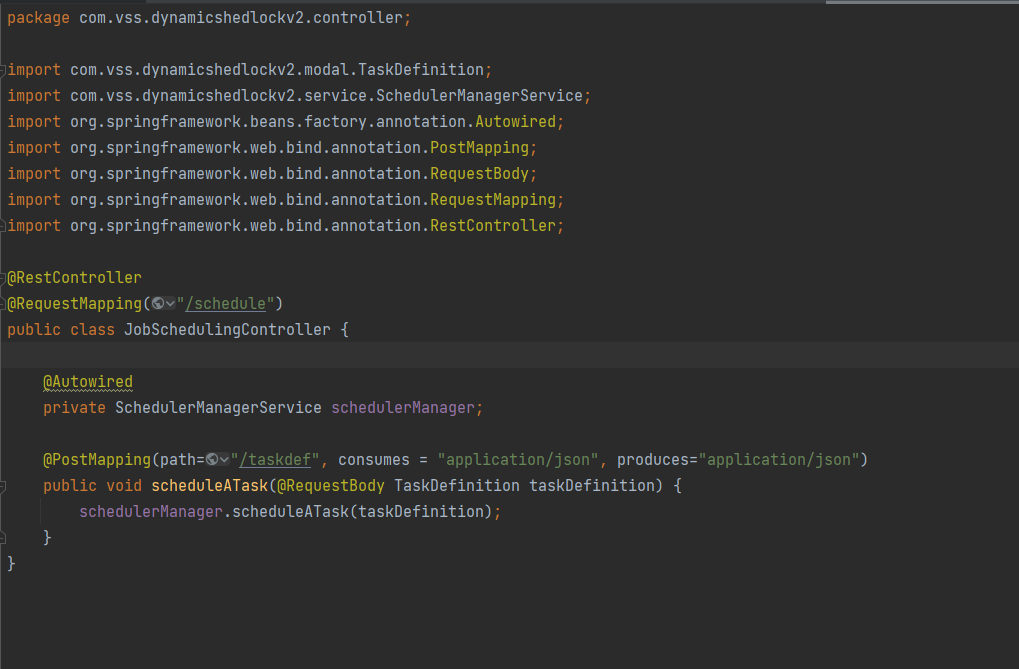
* Ở trên là chúng ta fix cứng code để cho Shed Lock tự động lập lịch. Ngoài ra ta còn có thể lập lịch theo cách mà ta tự định nghĩa.
* Tại dự án ta thêm annotation **@EnableScheduling**

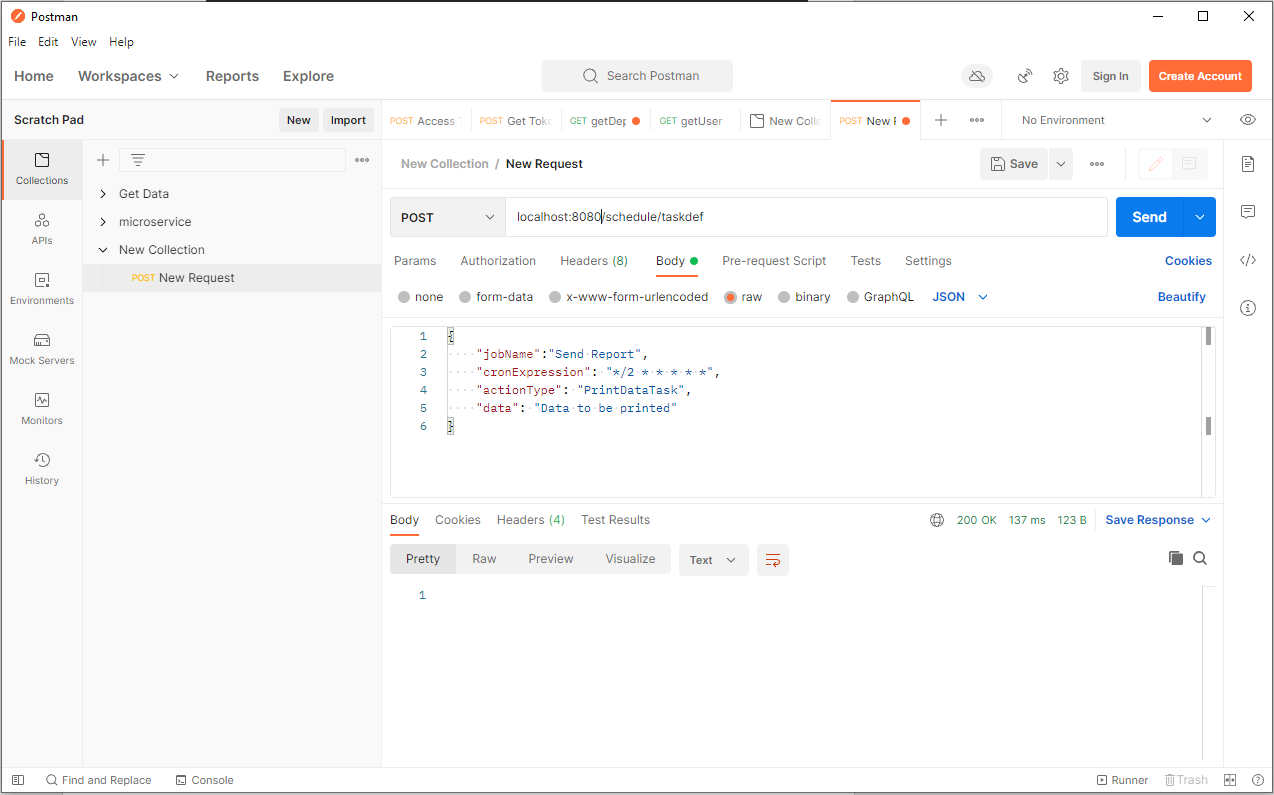


* Tạo class config **ShedLockConfig:** 
* Tạo class **LockRunable:** 
* Tạo class **SchedulerManagerService:**



* Tạo class **JobSchedulingController:**
* Tại Controller ta sẽ nhận vào một object taskDefenition từ body và sẽ có các trường thuộc tính như sau:
  + jobName
  + cronExpression
  + actionType
  + data



* Test trên Postman: 

1. **Các thuộc tính trong annotation @Scheduled**
   1. **@Scheduled(fixedDelay = 1000)**
   * Cứ sau khoảng thời gian fixedDelay thì nó lại chạy một lần.
   * Ở ví dụ trên thì cứ sau 1000ms (1 giây) thì task sẽ chạy lại method 1 lần.
   * Với fixedDelay thì chỉ khi nào task trước đó thực hiện xong thì nó mới chạy tiếp task đó lại lần nữa. Ví dụ sau 1 giây mà method đó chưa chạy xong thì nó sẽ chờ cho tới khi nào xong mới chạy lại lần tiếp theo.
   1. **@Scheduled(fixedRate = 1000)**
   * fixedRate giống với fixedDelay, tuy nhiên sau khoảng thời gìn fixedRate thì nó chạy tiếp một lần nữa mà không cần quan tâm lần chạy trước đã hoàn thành hay chưa.
   * Ví dụ sau 1s mà method chưa thực hiện xong thì nó vẫn chạy lần tiếp theo.
   1. **@Schedule(cron = “15 \* \* \* \* ?”)**
   * Hẹn một khoảng thời gian nào đó thì task sẽ thực hiện
   * VD: bạn muốn gửi vào 12h thứ 6 hàng tuần hoặc 23h59 ngày cuối tháng,…
   * VD trên thì sẽ thực hiện task vào giây thứ 15 của mỗi phút.
2. **Tài liệu tham khảo**
3. <https://www.baeldung.com/shedlock-spring>
4. [https://crontab.guru/#5\_4\_\*\_\*](https://crontab.guru/#5_4_*_*)
5. <https://stackjava.com/spring/huong-dan-tao-lich-task-scheduler-voi-schedule-trong-spring.html>
6. <https://viblo.asia/p/scheduling-task-trong-spring-boot-yMnKMy2QK7P>
7. <https://rieckpil.de/lock-scheduled-tasks-with-shedlock-and-spring-boot/>
8. <http://buraktas.com/spring-shedlock-tutorial/>
9. <https://www.baeldung.com/shedlock-spring>
10. <https://erkanerkisi.github.io/Dynamic-Scheduler-shedlock>