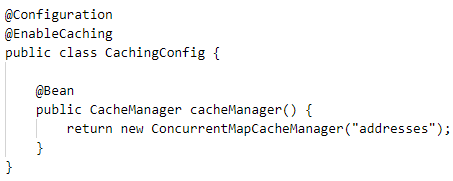
**CACHE TRONG SPRING BOOT**

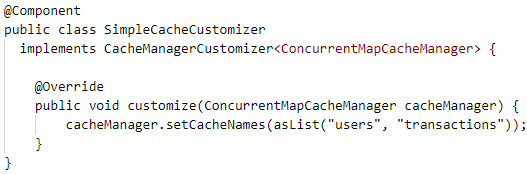
* Cache dữ liệu là một kỹ thuật quan trọng trong lập trình để cải thiện hiệu suất hệ thống bằng cách lưu trữ các dữ liệu phổ biến và thường xuyên truy cập trong bộ nhớ tạm (cache). Điều này giúp giảm thời gian truy cập dữ liệu từ nguồn lưu trữ chính, như cơ sở dữ liệu hoặc các hệ thống bên ngoài. Trong ngữ cảnh của Spring Boot, bạn có thể sử dụng Memory và Redis làm các giải pháp cache dữ liệu

**Cache dữ liệu với Memory trong Spring Boot**

* Memory caching sử dụng bộ nhớ tạm trong ứng dụng (JMV) để lưu trữ dữ liệu tạm thời. Spring Boot cung cấp một số annotation như **@Cacheable**, **@CachePut**, và **@CacheEvict** để đơn giản hóa việc quản lý cache
* Enable CacheManager:

****

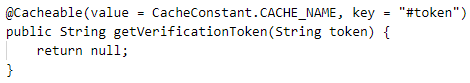
* Customize CacheManager:

****

* Use Caching With Annotations:

**@Cacheable**

**@Cacheable** là một annotation trong Spring Framework được sử dụng để kiểm tra cache trước khi thực hiện một phương thức. Nếu giá trị đã tồn tại trong cache với key tương ứng, phương thức sẽ không được gọi và giá trị sẽ được trả về từ cache. Nếu giá trị không tồn tại trong cache, phương thức sẽ được gọi, và giá trị trả về sẽ được đặt vào cache để lần sau có thể sử dụng



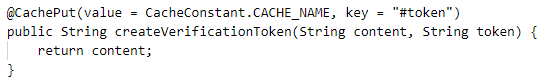
**@CacheEvict**

Là một annotation trong Spring Framework được sử dụng để xóa các mục khỏi cache. Mục đích chính của **@CacheEvict** là làm mới dữ liệu trong cache khi dữ liệu trong hệ thống có sự thay đổi hoặc cần được cập nhật

****

**@CachePut**

**@CachePut** là một annotation trong Spring Framework được sử dụng để cập nhật (hoặc đặt mới) một giá trị trong cache bằng giá trị trả về từ một phương thức. Annotation này thường được sử dụng khi bạn muốn đảm bảo rằng cache được cập nhật sau khi một phương thức được gọi. Nếu key không tồn tại trong cache, **@CachePut** sẽ tạo mới một cache entry với key và giá trị tương ứng



**Cache dữ liệu với Redis trong Spring Boot**