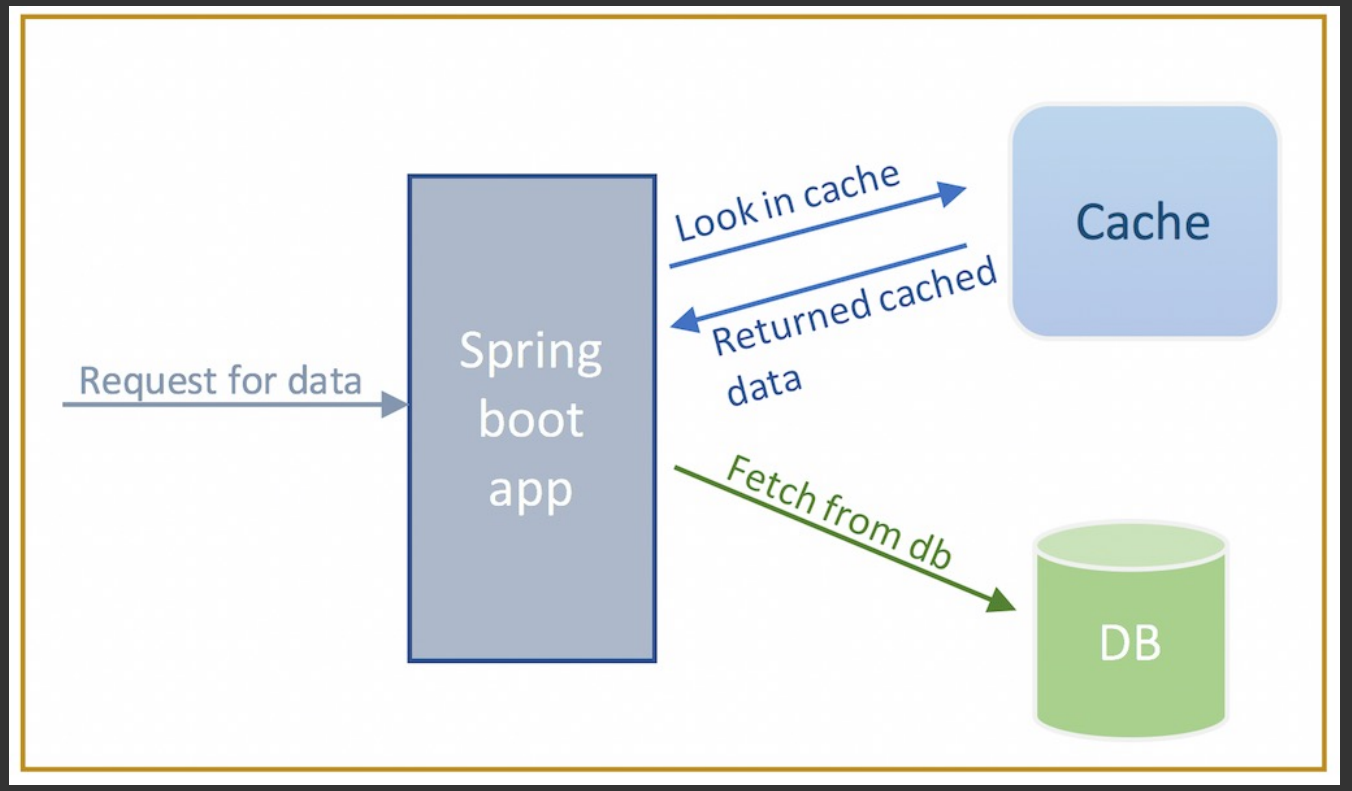
Caching

Spring boot default Caching

Spring Boot Caching

Spring boot provides a Cache Abstraction API that allow us to use different cache providers to cache objects.



Spring Boot Caching Annotations

1. @EnableCaching

It is a class level annotation to enable caching in spring boot application. By default it setup a CacheManager and creates in-memory cache using one concurrent HashMap.

2. @Cacheable

It is a method level annotation. It is used in the method whose response is to be cached. The Spring boot manages the request and response of the method to the cache that is specified in the annotation attribute.

Spring Boot Caching Annotations

3. @CachePut

It is a method level annotation. It is used to update the cache after invoking the method. By doing this, the result is put in the cache and the method is executed. It has same attributes of @Cacheable annotation.

4. @CacheEvict

It is a method level annotation. It is used to remove the data from the cache. When the method is annotated with this annotation then the method is executed and the cache will be removed / evicted.

Spring Boot Caching Internal working

1. When a method annotated with @Cacheable is called, Spring AOP intercepts the call and checks the cache before executing the method.
2. The CacheManager is called to retrieve the Cache associated with the cache name (e.g., "employees") using its get method.
3. Cache Hit or Miss:
    1. Hit: If the key exists, the cached value is returned directly.
    2. Miss: If the key does not exist, the original method is executed.
4. If the cache was missed, after the method executes, the result is stored in the cache using the put method of Cache.
5. Return Result: Finally, the result is returned to the caller, either from the cache or the method execution.