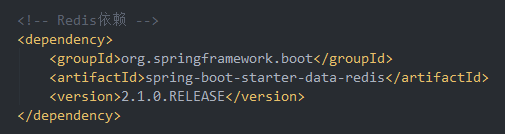
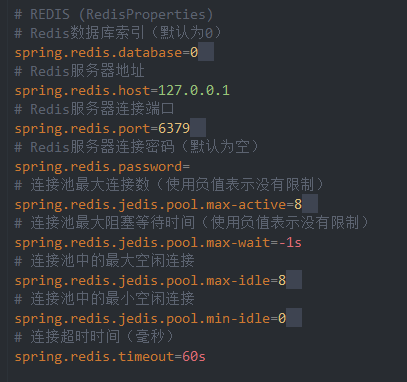
# Redis的基本使用

## 环境配置

1. 引入依赖



1. 配置文件



## 数据库操作

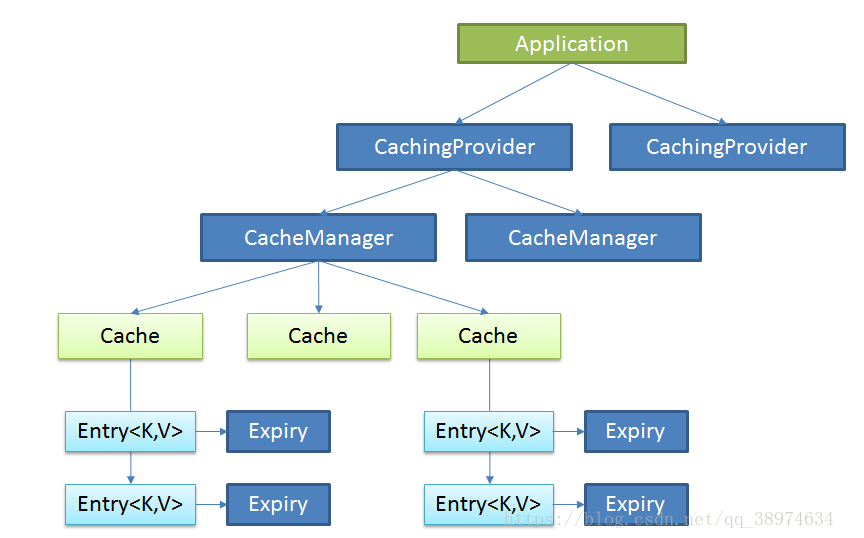


# SpringBoot的缓存机制

## 简介

* 1. JSR107是关于缓存使用的规范，主要是5个核心接口。

1. **CachingProvider**定义了创建、配置、获取、管理和控制多个CacheManager。一个应用可以在运行期访问多个CachingProvider。
2. **CacheManager**定义了创建、配置、获取、管理和控制多个唯一命名的Cache，这些Cache存在于CacheManager的上下文中。一个CacheManager仅被一个CachingProvider所拥有。
3. **Cache**是一个类似Map的数据结构并临时存储以Key为索引的值。一个Cache仅被一个CacheManager所拥有。
4. **Entry**是一个存储在Cache中的key-value对.
5. **Expiry** 每一个存储在Cache中的条目有一个定义的有效期。一旦超过这个时间，条目为过期的状态。一旦过期，条目将不可访问、更新和删除。缓存有效期可以通过ExpiryPolicy设置。



## Springboot中的缓存抽象

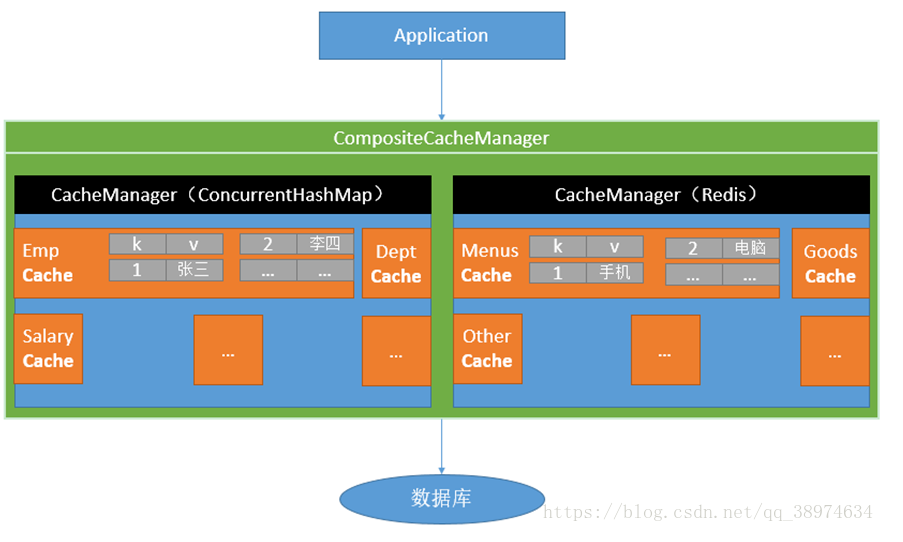
2.1 缓存的定义

1. Spring从3.1开始定义了

org.springframework.cache.Cache和org.springframework.cache.CacheManager

接口来统一不同的缓存技术

1. Cache接口为缓存的组件规范定义，包含缓存的各种操作集合；
2. Cache接口下Spring提供了各种xxxCache的实现,如RedisCache,EhCacheCache , ConcurrentMapCache等；每次调用需要缓存功能的方法时，Spring会检查检查指定参数的指定的目标方法是否已经被调用过；如果有就直接从缓存中获取方法调用后的结果，如果没有就调用方法并缓存结果返回给用户。下次直接从缓存中获取。
   1. 缓存的使用
3. 确定方法需要被缓存以及他们的缓存策略
4. 从缓存中读取之前缓存存储的数据



## 缓存组件

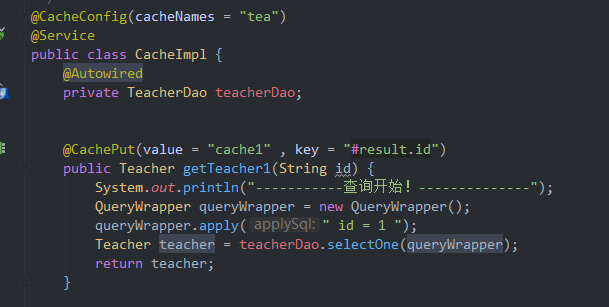
3.1 缓存相关类

1. Cache ：缓存接口，定义缓存操作。

实现类有，RedisCache,EhCacheCache,ConcurrentMapCache等。

1. CacheManager：缓存管理器，管理各种缓存（Cache）组件。
2. keyGenerator ：缓存数据时，key的生成策略。（cache块中key名称）
3. serialize ：缓存数据时，value序列化策略。（cache块名称）
   1. 缓存注解
4. @EnableCaching ：开启基于注解的缓存。
5. @CacheConfig ：用于类名上，指定该类的缓存块（Cache块名称），

具体用法为



1. @Cacheable ：配置在方法，根据方法请求参数缓存结果，并且第二次返回缓存。
2. @CachePut ：每次都调用方法，如果有变化，更新缓存。
3. @CacheEvict ：用于清空缓存。

（主要属性值）



（SpeL表达式）

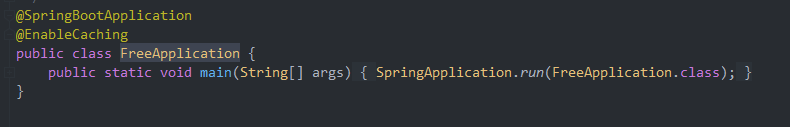


## 基本使用

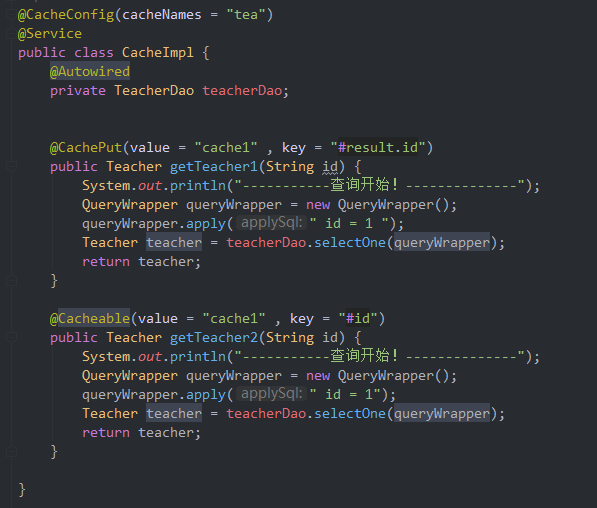
4.1 引入依赖



* 1. 启动类开启缓存注解

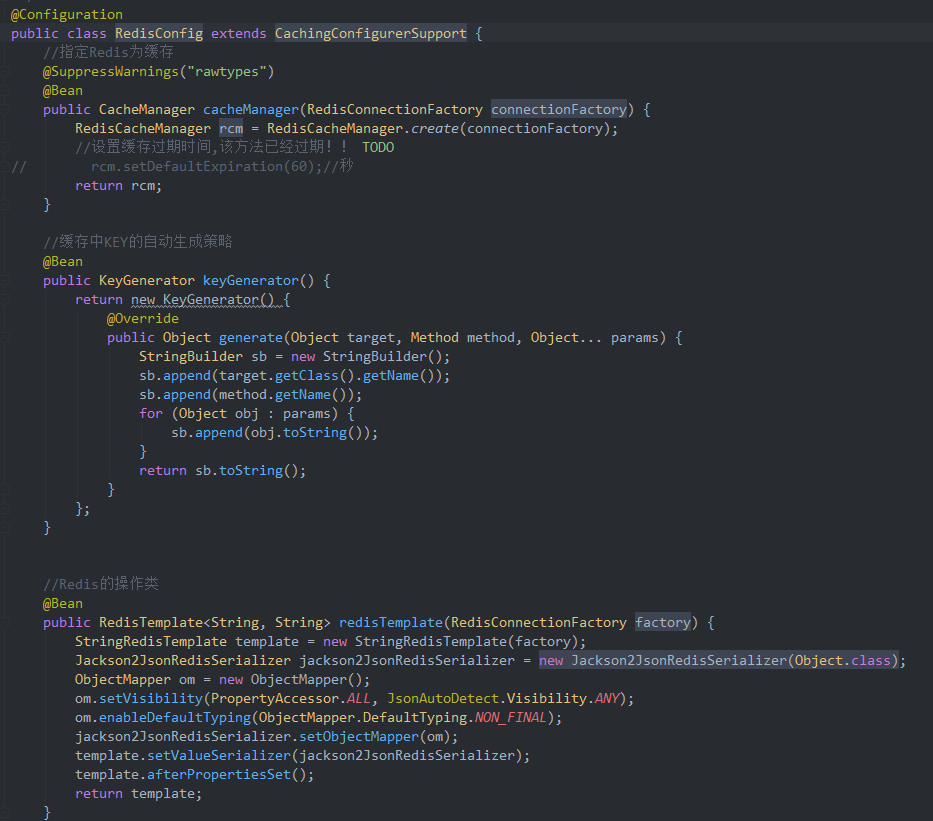


* 1. 使用缓存



# 使用Redis作为缓存

## Redis缓存配置类



## 和springBoot使用缓存方式相同

