**缓存数据**

**启用对缓存的支持**

**1. ConcurrentMapCacheManager(java自带的，有局限性)**

**@Configuration**

**@EnableCaching**

**public class CachingConfig {**

**@Bean**

**public CacheManager cacheManager(){**

**return new ConcurrentMapCacheManager();**

**}**

**}**

**2. 使用Ehcache缓存**

**@Configuration**

**@EnableCaching**

**public class CachingConfig{**

**@Bean**

**public EhCacheCacheManager cacheManager(CacheManager cm){**

**return new EhCacheCacheManager(cm);**

**}**

**@Bean**

**public EhCacheManagerFactoryBean ehcache(){**

**EhCacheManagerFactoryBean ehCacheFactoryBean =**

**new EhCacheManagerFactoryBean();**

**ehCacheFactoryBean.setConfigLocation(**

**new ClassPathResource("com/ehcache.xml"));**

**return ehCacheFactoryBean();**

**}**

**}**

**ehcache.xml**

**<ehcache>**

**<cache name = "spittleCache"**

**maxBytesLocalHeap="50m"**

**timeToLiveSeconds="100">**

**</cache>**

**</ehcache>**

**3. 使用Redis缓存**

**@Configuration**

**@EnableCaching**

**public class CachingConfig{**

**@Bean**

**public CacheManager cacheManager(RedisTemplate redisTemplate){**

**return new RedisCacheManager(redisTemplate);**

**}**

**@Bean**

**public JedisConnectionFactory redisConnectionFactory(){**

**JedisConnectionFactory jedisConnectionFactory =**

**new JedisConnectionFactory();**

**jedisConnectionFactory.afterPropertiesSet();**

**return jedisConnectionFactory;**

**}**

**@Bean**

**public RedisTemplate<String, String> redisTemplate(**

**RedisConnectionFactory redisCF){**

**RedisTemplate<String,String> redisTemplate =**

**new RedisTemplate<String, String>();**

**redisTemplate.setConnectionFactory(redisCF);**

**redisTemplate.afterPropertiesSet();**

**return redisTemplate;**

**}**

**}**

**4. 使用多个缓存管理器 CompositeCacheManager**

**@Bean**

**public CacheManager cacheManager(**

**net.sf.ehcache.CacheManager cm,**

**javax.cache.CacheManager jcm){**

**CompositeCacheManager cacheManager = new CompositeCacheManager();**

**List<CacheManager> managers = new ArrayList<CacheManager>();**

**managers.add(new JCacheCacheManager(jcm));**

**managers.add(new EhCacheCacheManager(cm));**

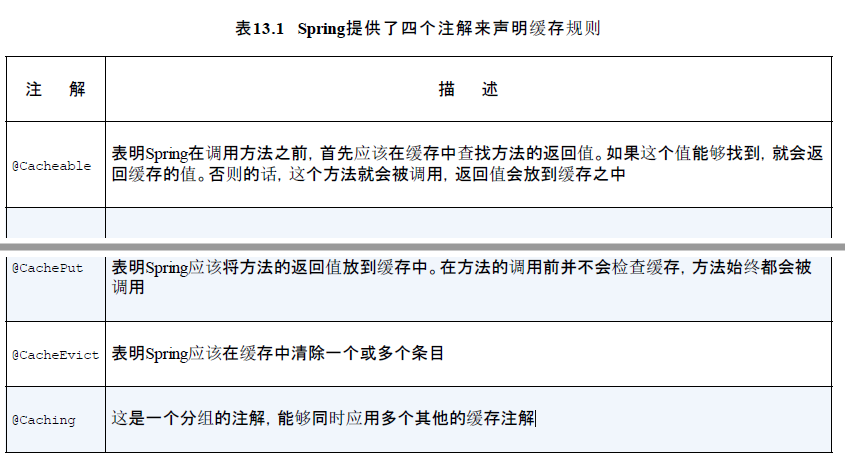
**managers.add(new ResidCacheManager(redisTemplate()));**

**return cacheManager;**

**}**

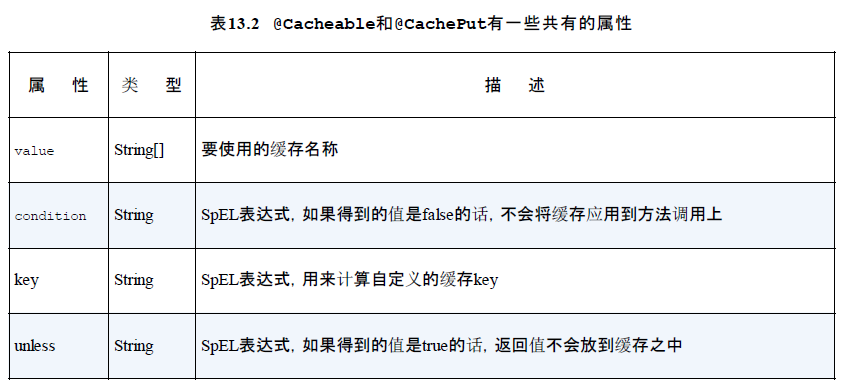
**为方法添加注解以支持缓存**

**注解放在类级别的话， 那么缓存的行为就会应用到这个类的所有方法上。 应用方法上就不说了**



**@Cacheable和@CachePut, 默认的key为方法的参数。 但是可以替换默认的key， 通过SpEL表达式计算得到的。 具体如下所示：**





**unless属性只能阻止将对象放进缓存, 但是在这个方法调用的时候， 依然会去缓存中进行查找，如果找到了匹配的值，就会返回找到的值。**

**condition, 不会去缓存进行查找，同时返回值也不会放进缓存中。**

**@CacheEvict 可以用在任何方法上， 但是@Cacheable和@CachePut 需要非void得返回值，放入缓存中**