一、ehcahe的介绍

EhCache 是一个纯Java的进程内缓存框架，具有快速、精干等特点，是Hibernate中默认的CacheProvider。Ehcache是一种广泛使用的开源Java分布式缓存。主要面向通用缓存,Java EE和轻量级容器。它具有内存和磁盘存储，缓存加载器,缓存扩展，缓存异常处理程序，一个gzip缓存servlet过滤器，支持REST和SOAP api等特点。

优点：

1. 快速
2. 简单
3. 多种缓存策略
4. 缓存数据有两级：内存和磁盘，因此无需担心容量问题
5. 缓存数据会在虚拟机重启的过程中写入磁盘
6. 可以通过RMI、可插入API等方式进行分布式缓存
7. 具有缓存和缓存管理器的侦听接口
8. 支持多缓存管理器实例，以及一个实例的多个缓存区域
9. 提供Hibernate的缓存实现

缺点：

1. 使用磁盘Cache的时候非常占用磁盘空间：这是因为DiskCache的算法简单，该算法简单也导致Cache的效率非常高。它只是对元素直接追加存储。因此搜索元素的时候非常的快。如果使用DiskCache的，在很频繁的应用中，很快磁盘会满。
2. 不能保证数据的安全：当突然kill掉java的时候，可能会产生冲突，EhCache的解决方法是如果文件冲突了，则重建cache。这对于Cache数据需要保存的时候可能不利。当然，Cache只是简单的加速，而不能保证数据的安全。如果想保证数据的存储安全，可以使用Bekeley DB Java Edition版本。这是个嵌入式数据库。可以确保存储安全和空间的利用率。

EhCache的分布式缓存有传统的RMI，1.5版的JGroups，1.6版的JMS。分布式缓存主要解决集群环境中不同的服务器间的数据的同步问题。

使用Spring的AOP进行整合，可以灵活的对方法的返回结果对象进行缓存。

下面将介绍Spring+EhCache详细实例。

二、详细实例讲解

**本实例的环境 eclipse + maven + spring + ehcache + junit**

**2.1、相关依赖pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.luo</groupId>

<artifactId>ehcache\_project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<!-- spring版本号 -->

<spring.version>3.2.8.RELEASE</spring.version>

<!-- junit版本号 -->

<junit.version>4.10</junit.version>

</properties>

<dependencies>

<!-- 添加Spring依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<!--单元测试依赖 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<!--spring单元测试依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

<scope>test</scope>

</dependency>

<!-- ehcache 相关依赖 -->

<dependency>

<groupId>net.sf.ehcache</groupId>

<artifactId>ehcache</artifactId>

<version>2.8.2</version>

</dependency>

</dependencies>

</project>

**2.2、添加ehcache配置文件ehcache-setting.xml**

<?xml version="1.0" encoding="UTF-8"?>

<ehcache>

<!-- 指定一个文件目录，当EhCache把数据写到硬盘上时，将把数据写到这个文件目录下 -->

<diskStore path="java.io.tmpdir"/>

<!-- 设定缓存的默认数据过期策略 -->

<defaultCache

maxElementsInMemory="10000"

eternal="false"

overflowToDisk="true"

timeToIdleSeconds="10"

timeToLiveSeconds="20"

diskPersistent="false"

diskExpiryThreadIntervalSeconds="120"/>

<cache name="cacheTest"

maxElementsInMemory="1000"

eternal="false"

overflowToDisk="true"

timeToIdleSeconds="10"

timeToLiveSeconds="20"/>

</ehcache>

这里我们配置了cacheTest策略，10秒过期。

**cache元素的属性：**

name：缓存名称

maxElementsInMemory：内存中最大缓存对象数

maxElementsOnDisk：硬盘中最大缓存对象数，若是0表示无穷大

eternal：true表示对象永不过期，此时会忽略timeToIdleSeconds和timeToLiveSeconds属性，默认为false

overflowToDisk：true表示当内存缓存的对象数目达到了

maxElementsInMemory界限后，会把溢出的对象写到硬盘缓存中。注意：如果缓存的对象要写入到硬盘中的话，则该对象必须实现了Serializable接口才行。

diskSpoolBufferSizeMB：磁盘缓存区大小，默认为30MB。每个Cache都应该有自己的一个缓存区。

diskPersistent：是否缓存虚拟机重启期数据，是否持久化磁盘缓存,当这个属性的值为true时,系统在初始化时会在磁盘中查找文件名为cache名称,后缀名为index的文件，这个文件中存放了已经持久化在磁盘中的cache的index,找到后会把cache加载到内存，要想把cache真正持久化到磁盘,写程序时注意执行net.sf.ehcache.Cache.put(Element element)后要调用flush()方法。

diskExpiryThreadIntervalSeconds：磁盘失效线程运行时间间隔，默认为120秒

timeToIdleSeconds： 设定允许对象处于空闲状态的最长时间，以秒为单位。当对象自从最近一次被访问后，如果处于空闲状态的时间超过了timeToIdleSeconds属性值，这个对象就会过期，EHCache将把它从缓存中清空。只有当eternal属性为false，该属性才有效。如果该属性值为0，则表示对象可以无限期地处于空闲状态

timeToLiveSeconds：设定对象允许存在于缓存中的最长时间，以秒为单位。当对象自从被存放到缓存中后，如果处于缓存中的时间超过了 timeToLiveSeconds属性值，这个对象就会过期，EHCache将把它从缓存中清除。只有当eternal属性为false，该属性才有效。如果该属性值为0，则表示对象可以无限期地存在于缓存中。timeToLiveSeconds必须大于timeToIdleSeconds属性，才有意义

memoryStoreEvictionPolicy：当达到maxElementsInMemory限制时，Ehcache将会根据指定的策略去清理内存。可选策略有：LRU（最近最少使用，默认策略）、FIFO（先进先出）、LFU（最少访问次数）。

**2.3、spring配置文件application.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:cache="http://www.springframework.org/schema/cache"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd

http://www.springframework.org/schema/cache

http://www.springframework.org/schema/cache/spring-cache-3.1.xsd">

<!-- 自动扫描注解的bean -->

<context:component-scan base-package="com.luo.service" />

<cache:annotation-driven cache-manager="cacheManager" />

<bean id="cacheManager" class="org.springframework.cache.ehcache.EhCacheCacheManager">

<property name="cacheManager" ref="ehcache"></property>

</bean>

<bean id="ehcache" class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">

<property name="configLocation" value="classpath:ehcache-setting.xml"></property>

</bean>

</beans>

**2.4、EhCacheTestService接口**

package com.luo.service;

public interface EhCacheTestService {

public String getTimestamp(String param);

}

**2.5、EhCacheTestService接口实现**

package com.luo.service.impl;

import org.springframework.cache.annotation.Cacheable;

import org.springframework.stereotype.Service;

import com.luo.service.EhCacheTestService;

@Service

public class EhCacheTestServiceImpl implements EhCacheTestService {

@Cacheable(value="cacheTest",key="#param")

public String getTimestamp(String param) {

Long timestamp = System.currentTimeMillis();

return timestamp.toString();

}

}

这里注解中value=”cacheTest”与ehcache-setting.xml中的cache名称属性值一致。

**2.6、单元测试类**

package com.luo.baseTest;

import org.junit.runner.RunWith;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.AbstractJUnit4SpringContextTests;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

//指定bean注入的配置文件

@ContextConfiguration(locations = { "classpath:application.xml" })

//使用标准的JUnit @RunWith注释来告诉JUnit使用Spring TestRunner

@RunWith(SpringJUnit4ClassRunner.class)

public class SpringTestCase extends AbstractJUnit4SpringContextTests {

}

package com.luo.service;

import org.junit.Test;

import org.springframework.beans.factory.annotation.Autowired;

import com.luo.baseTest.SpringTestCase;

public class EhCacheTestServiceTest extends SpringTestCase {

@Autowired

private EhCacheTestService ehCacheTestService;

@Test

public void getTimestampTest() throws InterruptedException{

System.out.println("第一次调用：" + ehCacheTestService.getTimestamp("param"));

Thread.sleep(2000);

System.out.println("2秒之后调用：" + ehCacheTestService.getTimestamp("param"));

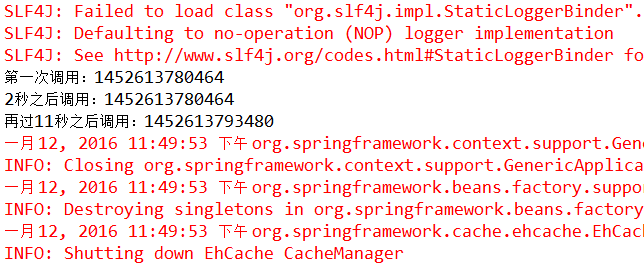
Thread.sleep(11000);

System.out.println("再过11秒之后调用：" + ehCacheTestService.getTimestamp("param"));

}

}

**2.7、运行结果**



三、工程源码下载

<http://download.csdn.net/detail/u013142781/9401689>