

**这里采用的是spring3.2、ehcache2.7、tomcat7.0(必须)**

1.web.xml

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

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

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<!-- SpringMVC核心分发器 -->

<servlet>

<servlet-name>SpringMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>SpringMVC</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

2.applicationContext.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:mvc="http://www.springframework.org/schema/mvc"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

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

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd

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

http://www.springframework.org/schema/cache/spring-cache-3.2.xsd

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

http://www.springframework.org/schema/context/spring-context-3.2.xsd">

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

<!-- 缓存配置 -->

<!-- 启用缓存注解功能(请将其配置在Spring主配置文件中) -->

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

<!-- Spring自己的基于java.util.concurrent.ConcurrentHashMap实现的缓存管理器(该功能是从Spring3.1开始提供的) -->

<!-- <bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">

<property name="caches"> <set> <bean name="myCache" class="org.springframework.cache.concurrent.ConcurrentMapCacheFactoryBean"/>

</set> </property> </bean> -->

<!-- 若只想使用Spring自身提供的缓存器,则注释掉下面的两个关于Ehcache配置的bean,并启用上面的SimpleCacheManager即可 -->

<!-- Spring提供的基于的Ehcache实现的缓存管理器 -->

<bean id="cacheManagerFactory"

class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">

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

</bean>

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

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

</bean>

</beans>

3.ehcache.xml

<!-- Ehcache2.x的变化(取自https://github.com/springside/springside4/wiki/Ehcache) -->

<!-- 1)最好在ehcache.xml中声明不进行updateCheck -->

<!-- 2)为了配合BigMemory和Size Limit,原来的属性最好改名 -->

<!-- maxElementsInMemory->maxEntriesLocalHeap -->

<!-- maxElementsOnDisk->maxEntriesLocalDisk -->

<ehcache>

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

<defaultCache

maxElementsInMemory="1000"

eternal="false"

timeToIdleSeconds="120"

timeToLiveSeconds="120"

overflowToDisk="false"/>

<cache name="myCache"

maxElementsOnDisk="20000"

maxElementsInMemory="2000"

eternal="true"

overflowToDisk="true"

diskPersistent="true"/>

</ehcache>

<!--

<diskStore>==========当内存缓存中对象数量超过maxElementsInMemory时,将缓存对象写到磁盘缓存中(需对象实现序列化接口)

<diskStore path="">==用来配置磁盘缓存使用的物理路径,Ehcache磁盘缓存使用的文件后缀名是\*.data和\*.index

name=================缓存名称,cache的唯一标识(ehcache会把这个cache放到HashMap里)

maxElementsOnDisk====磁盘缓存中最多可以存放的元素数量,0表示无穷大

maxElementsInMemory==内存缓存中最多可以存放的元素数量,若放入Cache中的元素超过这个数值,则有以下两种情况

1)若overflowToDisk=true,则会将Cache中多出的元素放入磁盘文件中

2)若overflowToDisk=false,则根据memoryStoreEvictionPolicy策略替换Cache中原有的元素

eternal==============缓存中对象是否永久有效,即是否永驻内存,true时将忽略timeToIdleSeconds和timeToLiveSeconds

timeToIdleSeconds====缓存数据在失效前的允许闲置时间(单位:秒),仅当eternal=false时使用,默认值是0表示可闲置时间无穷大,此为可选属性

即访问这个cache中元素的最大间隔时间,若超过这个时间没有访问此Cache中的某个元素,那么此元素将被从Cache中清除

timeToLiveSeconds====缓存数据在失效前的允许存活时间(单位:秒),仅当eternal=false时使用,默认值是0表示可存活时间无穷大

即Cache中的某元素从创建到清楚的生存时间,也就是说从创建开始计时,当超过这个时间时,此元素将从Cache中清除

overflowToDisk=======内存不足时,是否启用磁盘缓存(即内存中对象数量达到maxElementsInMemory时,Ehcache会将对象写到磁盘中)

会根据标签中path值查找对应的属性值,写入磁盘的文件会放在path文件夹下,文件的名称是cache的名称,后缀名是data

diskPersistent=======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==磁盘缓存的清理线程运行间隔,默认是120秒

diskSpoolBufferSizeMB============设置DiskStore（磁盘缓存）的缓存区大小,默认是30MB

memoryStoreEvictionPolicy========内存存储与释放策略,即达到maxElementsInMemory限制时,Ehcache会根据指定策略清理内存

共有三种策略,分别为LRU(最近最少使用)、LFU(最常用的)、FIFO(先进先出)

-->

4.UserController.java

package com.controller;

import net.sf.ehcache.CacheManager;

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import com.service.UserService;

@Controller

public class UserController

{

@Autowired

private UserService userService;

@Autowired

private CacheManager cacheManager;

@RequestMapping(value = "test", method = RequestMethod.GET)

public String test()

{

System.out.println(cacheManager.getCache("myCache").get("test"));

userService.test();

return "hello";

}

}

6.UserService.java

package com.service;

import java.util.List;

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

import org.springframework.cache.annotation.Cacheable;

import org.springframework.stereotype.Service;

import com.dao.UserDao;

import com.model.User;

@Service

public class UserService

{

@Autowired

private UserDao userDao;

@Cacheable(value = "myCache", key = "'test'")

public List<User> test()

{

System.out.println("select from db");

return userDao.test();

}

}

7.UserDao.java

package com.dao;

import java.util.ArrayList;

import java.util.List;

import org.springframework.stereotype.Repository;

import com.model.User;

@Repository

public class UserDao

{

private List<User> userlist;

private User user;

public UserDao()

{

userlist = new ArrayList<User>();

user = new User();

user.setId(0);

userlist.add(user);

user = new User();

user.setId(1);

userlist.add(user);

user = new User();

user.setId(2);

userlist.add(user);

user = new User();

user.setId(3);

userlist.add(user);

}

public List<User> test()

{

return userlist;

}

}

8.User.java

package com.model;

import java.io.Serializable;

public class User implements Serializable

{

/\*\*

\*

\*/

private static final long serialVersionUID = 5104855734772499069L;

private int id;

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

@Override

public String toString()

{

return "User [id=" + id + "]";

}

}

9.浏览器输入<http://localhost:8080/springcache/test>，第一次输出null,select from db,第二次输出[cache.........]