从Spring3.1开始，增加了抽像缓存框架，利用Spring3.1提供的注释，可以很方便的使用缓存，Spring官方给出基于Map和EHCache的实现，正好最近在用Memcached，参考了EHCacheCacheManager的源码，写了一个基于XMemcached的MemcachedCacheManager。

直接上代码(注：只支持Spring3.2以上版本)

1、MemcachedCacheManager.java

public class MemcachedCacheManager extends AbstractTransactionSupportingCacheManager

{

private ConcurrentMap<String, Cache> cacheMap = new ConcurrentHashMap<String, Cache>();

private Map<String, Integer> expireMap = new HashMap<String, Integer>();

private MemcachedClient memcachedClient;

public MemcachedCacheManager()

{

}

@Override

protected Collection<? extends Cache> loadCaches()

{

Collection<Cache> values = cacheMap.values();

return values;

}

@Override

public Cache getCache(String name)

{

Cache cache = cacheMap.get(name);

if (cache == null)

{

Integer expire = expireMap.get(name);

if (expire == null)

{

expire = 0;

expireMap.put(name, expire);

}

cache = new MemcachedCache(name, expire.intValue(), memcachedClient);

cacheMap.put(name, cache);

}

return cache;

}

public void setMemcachedClient(MemcachedClient memcachedClient)

{

this.memcachedClient = memcachedClient;

}

public void setConfigMap(Map<String, Integer> configMap)

{

this.expireMap = configMap;

}

}

 2、MemcachedCache.java

public class MemcachedCache implements Cache

{

private final String name;

private final MemcachedClient memcachedClient;

private final MemCache memCache;

public MemcachedCache(String name, int expire, MemcachedClient memcachedClient)

{

this.name = name;

this.memcachedClient = memcachedClient;

this.memCache = new MemCache(name, expire, memcachedClient);

}

@Override

public void clear()

{

memCache.clear();

}

@Override

public void evict(Object key)

{

memCache.delete(key.toString());

}

@Override

public ValueWrapper get(Object key)

{

ValueWrapper wrapper = null;

Object value = memCache.get(key.toString());

if (value != null)

{

wrapper = new SimpleValueWrapper(value);

}

return wrapper;

}

@Override

public String getName()

{

return this.name;

}

@Override

public MemcachedClient getNativeCache()

{

return this.memcachedClient;

}

@Override

public void put(Object key, Object value)

{

memCache.put(key.toString(), value);

}

}

 3、MemCache.java

public class MemCache

{

private static Logger log = LoggerFactory.getLogger(MemCache.class);

private Set<String> keySet = new HashSet<String>();

private final String name;

private final int expire;

private final MemcachedClient memcachedClient;

public MemCache(String name, int expire, MemcachedClient memcachedClient)

{

this.name = name;

this.expire = expire;

this.memcachedClient = memcachedClient;

}

public Object get(String key)

{

Object value = null;

try

{

key = this.getKey(key);

value = memcachedClient.get(key);

}

catch (TimeoutException e)

{

log.warn("获取 Memcached 缓存超时", e);

}

catch (InterruptedException e)

{

log.warn("获取 Memcached 缓存被中断", e);

}

catch (MemcachedException e)

{

log.warn("获取 Memcached 缓存错误", e);

}

return value;

}

public void put(String key, Object value)

{

if (value == null)

return;

try

{

key = this.getKey(key);

memcachedClient.setWithNoReply(key, expire, value);

keySet.add(key);

}

catch (InterruptedException e)

{

log.warn("更新 Memcached 缓存被中断", e);

}

catch (MemcachedException e)

{

log.warn("更新 Memcached 缓存错误", e);

}

}

public void clear()

{

for (String key : keySet)

{

try

{

memcachedClient.deleteWithNoReply(this.getKey(key));

}

catch (InterruptedException e)

{

log.warn("删除 Memcached 缓存被中断", e);

}

catch (MemcachedException e)

{

log.warn("删除 Memcached 缓存错误", e);

}

}

}

public void delete(String key)

{

try

{

key = this.getKey(key);

memcachedClient.deleteWithNoReply(key);

}

catch (InterruptedException e)

{

log.warn("删除 Memcached 缓存被中断", e);

}

catch (MemcachedException e)

{

log.warn("删除 Memcached 缓存错误", e);

}

}

private String getKey(String key)

{

return name + "\_" + key;

}

}

 4、配置文件

<?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"

xsi:schemaLocation="

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

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

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

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

default-autowire="byName">

<bean id="memcachedClientBuilder" class="net.rubyeye.xmemcached.XMemcachedClientBuilder">

<constructor-arg>

<list>

<bean class="java.net.InetSocketAddress">

<constructor-arg value="localhost"/>

<constructor-arg value="11211"/>

</bean>

</list>

</constructor-arg>

<property name="connectionPoolSize" value="5"/>

<property name="commandFactory">

<bean class="net.rubyeye.xmemcached.command.BinaryCommandFactory"/>

</property>

<property name="transcoder">

<bean class="net.rubyeye.xmemcached.transcoders.SerializingTranscoder" />

</property>

</bean>

<bean id="memcachedClient" factory-bean="memcachedClientBuilder" factory-method="build" destroy-method="shutdown"/>

<bean id="cacheManager" class="xxx.MemcachedCacheManager">

<property name="memcachedClient" ref="memcachedClient" />

<property name="configMap">

<map>

<!-- key：@Cacheable、@CachePut、@CacheEvict等的name属性。value：缓存过期时间(单位：秒)，默认值：0 -->

<entry key="typeList" value="3600" />

</map>

</property>

</bean>

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

</beans>