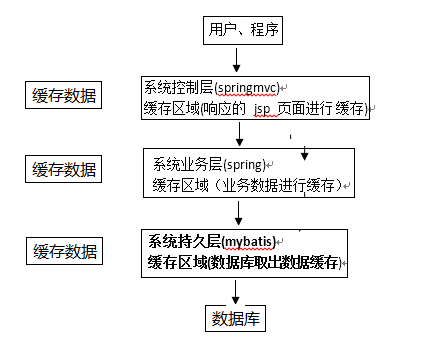
# 缓存

## 缓存的意义

将用户经常查询的数据放在缓存（内存）中，用户去查询数据就不用从磁盘上(关系型数据库数据文件)查询，从缓存中查询，从而提高查询效率，解决了高并发系统的性能问题。



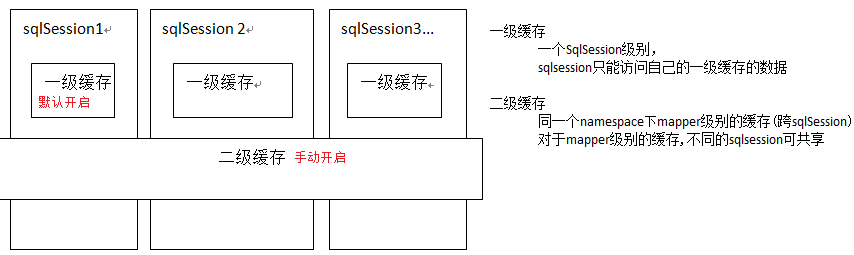
## 分类

一级缓存指的就是sqlsession，在sqlsession中有一个数据区域，是map结构，这个区域就是一级缓存区域。一级缓存中的key是由sql语句、条件、statement等信息组成一个唯一值。一级缓存中的value，就是查询出的结果对象。

二级缓存（查询缓存）作用域是一个mapper的namespace，即在同一个namespace中查询sql可以从缓存中获取数据。二级缓存是可以跨SqlSession的。

一级缓存是默认使用的。

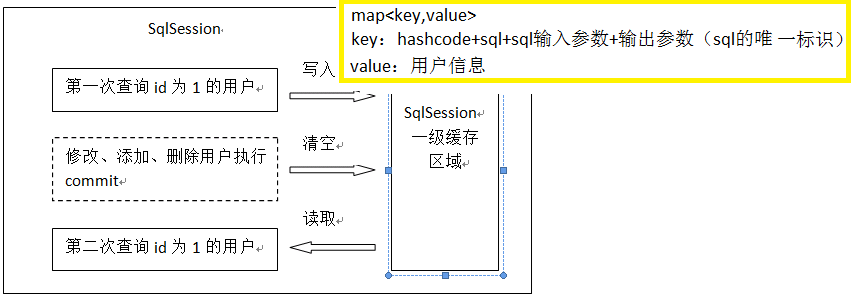
二级缓存需要手动开启。



# mybatis持久层缓存

## 一级缓存—自带

### 原理

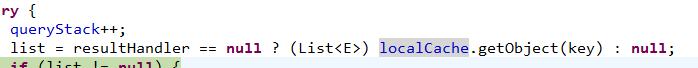


第一次发出一个查询sql，sql查询结果写入sqlsession的一级缓存中，缓存使用的数据结构是一个map<key,value>，同一个sqlsession再次发出相同的sql，就从缓存中取不走数据库。

如果两次中间出现commit操作（修改、添加、删除），本sqlsession中的一级缓存区域全部清空，下次再去缓存中查询不到所以要从数据库查询，从数据库查询到再写入缓存。

每次查询都先从缓存中查询：

如果缓存中查询到则将缓存数据直接返回。

 如果缓存中查询不到就从数据库查询：



### 一级缓存配置

mybatis默认支持一级缓存不需要配置。

注意：mybatis和spring整合后进行mapper代理开发，不支持一级缓存，mybatis和spring整合，spring按照mapper的模板去生成mapper代理对象，模板中在最后统一关闭sqlsession。

### 一级缓存测试



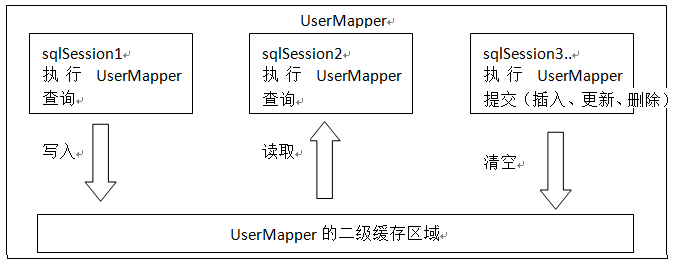
### map<key,value>

key：hashcode+sql+sql输入参数+输出参数（sql的唯一标识）

value：用户信息

## 二级缓存—查询缓存

### 原理



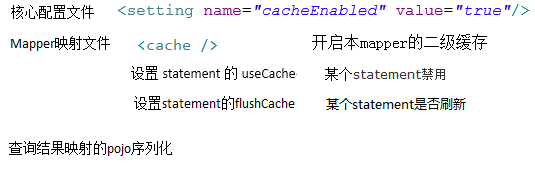
二级缓存的范围是mapper级别（mapper同一个命名空间），mapper以命名空间为单位创建缓存数据结构，结构是map<key、value>。

每次查询先看是否开启二级缓存，如果开启从二级缓存的数据结构中取缓存数据，



如果从二级缓存没有取到，再从一级缓存中找，如果一级缓存也没有，从数据库查询。

### 二级缓存配置



#### 核心配置文件

核心配置文件SqlMapConfig.xml中

<setting name=*"cacheEnabled"* value=*"true"*/>



#### 开启二级缓存

Mapper映射文件：  <cache /> ，表示此mapper开启二级缓存。



#### 某个statement禁用

某个statement禁用：对于变化频率较高的sql，需要禁用二级缓存



#### 某个statement禁用

如果sqlsession操作commit操作，对二级缓存进行刷新（全局清空）。

设置statement的flushCache是否刷新缓存，默认值是true。

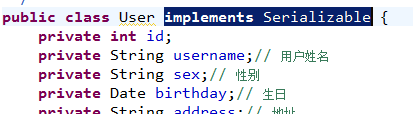
#### 查询结果映射的pojo序列化

mybatis二级缓存需要将查询结果映射的pojo实现 java.io.serializable接口，如果不实现则抛出异常：

org.apache.ibatis.cache.CacheException: Error serializing object. Cause: java.io.NotSerializableException: cn.itcast.mybatis.po.User

二级缓存可以将内存的数据写到磁盘，存在对象的序列化和反序列化，所以要实现java.io.serializable接口。

如果结果映射的pojo中还包括了pojo，都要实现java.io.serializable接口。



### 测试代码



### mybatis的cache参数（了解）

mybatis的cache参数只适用于mybatis维护缓存。

flushInterval（刷新间隔）可以被设置为任意的正整数，而且它们代表一个合理的毫秒形式的时间段。默认情况是不设置，也就是没有刷新间隔，缓存仅仅调用语句时刷新。

size（引用数目）可以被设置为任意正整数，要记住你缓存的对象数目和你运行环境的可用内存资源数目。默认值是1024。

readOnly（只读）属性可以被设置为true或false。只读的缓存会给所有调用者返回缓存对象的相同实例。因此这些对象不能被修改。这提供了很重要的性能优势。可读写的缓存会返回缓存对象的拷贝（通过序列化）。这会慢一些，但是安全，因此默认是false。

如下例子：

<cache  eviction="FIFO"  flushInterval="60000"  size="512"  readOnly="true"/>

这个更高级的配置创建了一个 FIFO 缓存,并每隔 60 秒刷新,存数结果对象或列表的 512 个引用,而且返回的对象被认为是只读的,因此在不同线程中的调用者之间修改它们会导致冲突。可用的收回策略有, 默认的是 LRU:

1. LRU – 最近最少使用的:移除最长时间不被使用的对象。
2. FIFO – 先进先出:按对象进入缓存的顺序来移除它们。
3. SOFT – 软引用:移除基于垃圾回收器状态和软引用规则的对象。
4. WEAK – 弱引用:更积极地移除基于垃圾收集器状态和弱引用规则的对象。

### 应用场景

对查询频率高，变化频率低的数据建议使用二级缓存。

对于访问多的查询请求且用户对查询结果实时性要求不高，此时可采用mybatis二级缓存技术降低数据库访问量，提高访问速度，业务场景比如：耗时较高的统计分析sql、电话账单查询sql等。

实现方法如下：通过设置刷新间隔时间，由mybatis每隔一段时间自动清空缓存，根据数据变化频率设置缓存刷新间隔flushInterval，比如设置为30分钟、60分钟、24小时等，根据需求而定。

# mybatis和ehcache缓存框架整合

mybatis二级缓存通过ehcache维护缓存数据。

## 分布缓存

将缓存数据数据进行分布式管理。

系统1

springmvc

mybatis

系统2

springmvc

mybatis

用户请求

缓存框架（远程服务器集群）

redis、memcached、ehcache

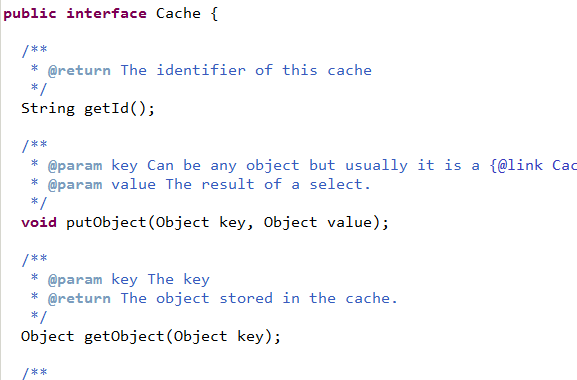
存取缓存数据

存取缓存数据

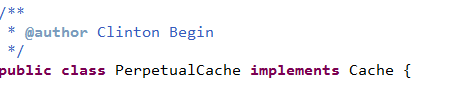
## mybatis和ehcache思路

通过mybatis和ehcache框架进行整合，就可以把缓存数据的管理托管给ehcache。

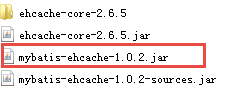
在mybatis中提供一个cache接口，只要实现cache接口就可以把缓存数据灵活的管理起来。



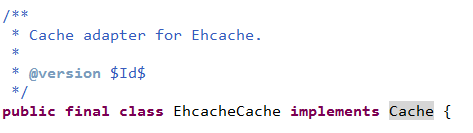
mybatis中默认实现：



## 下载和ehcache整合的jar包



ehcache对cache接口的实现类：



## 配置ehcache.xml



## 整合测试

在mapper.xml添加ehcache配置：

