

[博客](#)[学院](#)[下载](#)[GitChat](#)[论坛](#)[写博客](#)[发Chat](#)[登录](#)[注册](#)

木小草 专栏

<http://www.muxiaocao.cn/me>[RSS订阅](#)

3

[目录](#)[收藏](#)[评论](#)[微信](#)[微博](#)[QQ](#)

个人资料

[木小草](#)[关注](#)

原创

29

粉丝

40

喜欢

4

评论

16

等级：[博客 4](#)

访问：8万+

积分：1156

排名：4万+

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

[登录](#)[注册](#)

最新文章

- 搭建自己域名的hexo框架next风格的博客
- 漫谈Java中的互斥同步
- 系统架构设计——设计模式之模板模式
- 分布式系统架构——Mysql数据库实现主从同步
- Ubuntu15.10下Solr 6.0的搭建与IKAnalyzer中文分词结合使用

个人分类

- hadoop5篇
- 前端3篇
- bootstrap1篇
- Matlab1篇
- java11篇

展开

归档

- 2016年8月2篇
- 2016年7月3篇

2016年4月

5篇

展开

热门文章

Bootstrap的一个很漂亮的web万能模板
阅读量：10931

系统架构设计——学习篇之类的设计(UML)
阅读量：10552

系统架构设计——设计模式之代理模式
（二）CGLIB动态代理实现
阅读量：10240

分布式系统架构——dubbo与SSM整合问题
阅读量：8927

分布式系统架构——使用Redis做MyBatis的二级缓存
阅读量：8026

最新评论

Java与Matlab混合编程
Eric__F：你好，我有点问题想请教你，可以加一下我的微信或者qq吗？微信：15095411530 qq：128...

系统架构设计——学习篇之类的设计(...
shichuwu：用实例来分享

系统架构设计——学习篇之类的设计(...

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

xsh80144242: 不错啊这个 bootstrap特效对照手册: <http://t.cn/RK5JCg6>

分布式系统架构——dubbo与SS...

BlingBlingBlingU: [reply]huangpingcai[/reply]ok, 解决, 谢谢

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 🗨 客服论坛

关于 招聘 广告服务 🐾 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

原 分布式系统架构——使用Redis做MyBatis的二级缓存

2016年07月29日 16:23:15

阅读数: 8033

加入CSDN, 享受更精准的内容推荐, 与500万程序员共同成长!

登录

注册



通常为了减轻数据库的压力，我们会引入缓存。在Dao查询数据库之前，先去缓存中找是否有要找的数据，如果有则用缓存中的数据即可，就不用查询数据库了。如果没有才去数据库中查找。这样就能分担一下数据库的压力。另外，为了让缓存中的数据与数据库同步，我们应该在该数据发生变化的地方加入更新缓存的逻辑代码。这样无形之中增加了工作量，同时也是一种对原有代码的入侵。这对于有着代码洁癖的程序员来说，无疑是一种伤害。MyBatis框架早就考虑到了这些问题，因此MyBatis提供了自定义的二级缓存概念，方便引入我们自己的缓存机制，而不用更改原有的业务逻辑。下面就让我们了解一下MyBatis的缓存机制。

一、缓存概述

正如大多数持久层框架一样，MyBatis 同样提供了一级缓存和二级缓存的支持；

- 一级缓存基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域 为 Session，当 Session flush 或 close 之后，该Session中的所有 Cache 就将清空。
- 二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache、Hazelcast等。
- 对于缓存数据更新机制，当某一个作用域(一级缓存Session/二级缓存Namespaces)的进行了 C/U/D 操作后，默认该作用域下所有 select 中的缓存将被clear。
- MyBatis 的缓存采用了delegate机制 及 装饰器模式设计，当put、get、remove时，其中会经过多层 delegate cache 处理，其Cache类别有：BaseCache(基础缓存)、EvictionCache(排除算法缓存)、DecoratorCache(装饰器缓存)：
 1. BaseCache：为缓存数据最终存储的处理类，默认为 PerpetualCache，基于Map存储；可自定义存储处理，如基于EhCache、Memcached 等；
 2. EvictionCache：当缓存数量达到一定大小后，将通过算法对缓存数据进行清除。默认采用 Lru 算法(LruCache)，提供有 fifo 算法(FifoCache) 等；

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

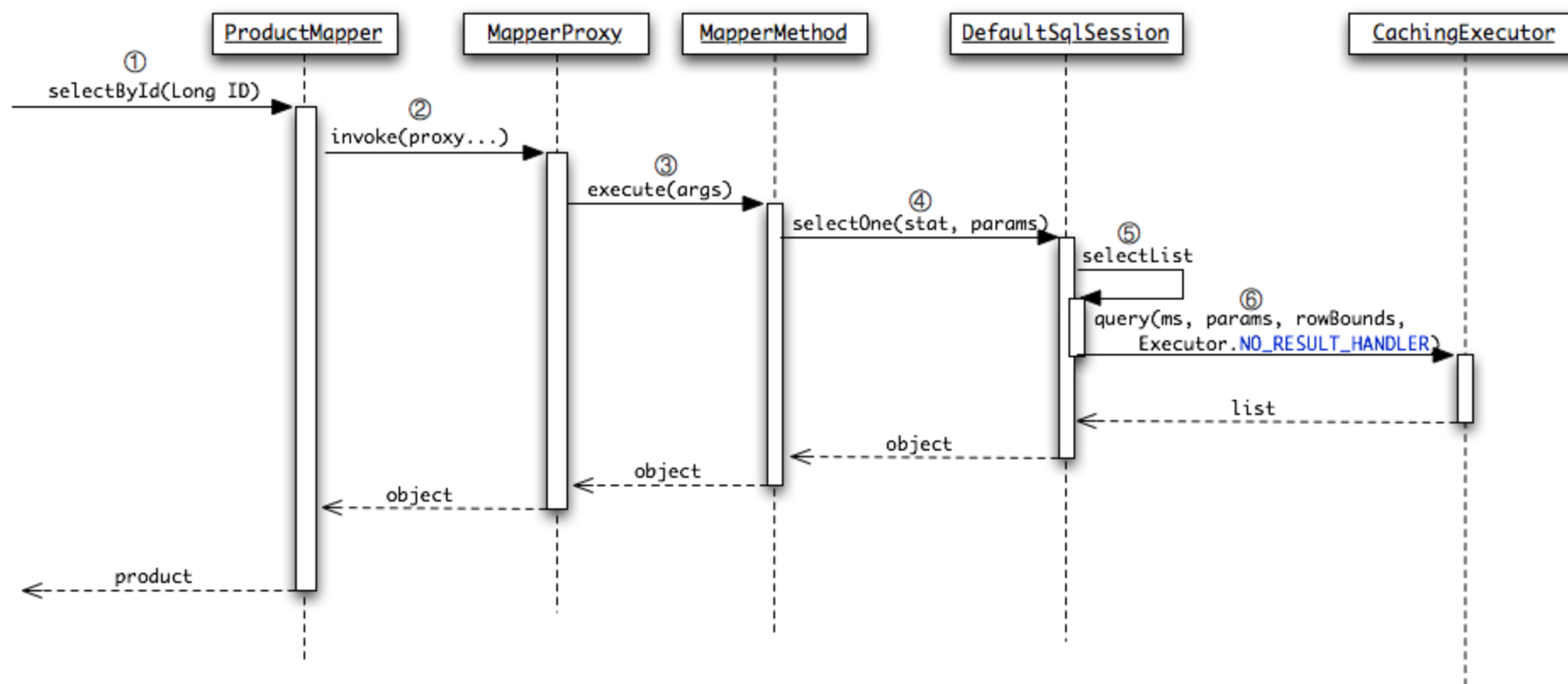


- 一般缓存框架的数据结构基本上都是 Key-Value 方式存储，MyBatis 对于其 Key 的生成采取规则为：
[hashCode : checksum : mappedStatementId : offset : limit : executeSql : queryParams]。
- 对于并发 Read/Write 时缓存数据的同步问题，MyBatis 默认基于 JDK/concurrent中的ReadWriteLock，使用 ReentrantReadWriteLock 的实现，从而通过 Lock 机制防止在并发 Write Cache 过程中线程安全问题。

二、源码剖析

2.1 执行流程分析

接下来将结合 MyBatis 序列图进行源码分析。在分析其Cache前，先看看其整个处理过程。



1. 通常我们在service层最终都会调用Mapper的接口方法，实现对数据库的操作，本例中是通过id查询product对象。
2. 我们知道Mapper是一个接口，接口是没有对象的，更不能调用方法了，而我们调用的其实是mybatis框架的mapper动态代理对象MapperProxy，而MapperProxy中有封装了配置信息的DefaultSqlSession中的Configuration。动态代理的具体实现请戳[这里](#)。调用mapper方法的具体代码如下。

```
public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    if (Object.class.equals(method.getDeclaringClass())) {
        return method.invoke(this, args);
    }
    final MapperMethod mapperMethod = cachedMapperMethod(method);
    return mapperMethod.execute(sqlSession, args);
}

private MapperMethod cachedMapperMethod(Method method) {
    MapperMethod mapperMethod = methodCache.get(method);
    if (mapperMethod == null) {
        mapperMethod = new MapperMethod(mapperInterface, method, sqlSession.getConfiguration());
        methodCache.put(method, mapperMethod);
    }
    return mapperMethod;
}
```

3. 在执行mapperMethod的execute的时候，不仅传递了方法参数，还传递了sqlSession。在执行execute，其实是通过判断配置文件的操作类型，来调用sqlSession的对应方法的。本例中，由于是select，而返回值不是list，所以下一步执行的是sqlSession的selectOne方法具体代码如下：


```
public Object execute(SqlSession sqlSession, Object[] args) {
    Object result;
    if (SqlCommandType.INSERT == command.getType()) {
        Object param = method.convertArgsToSqlCommandParam(args);
        result = rowCountResult(sqlSession.insert(command.getName(), param));
    } else if (SqlCommandType.UPDATE == command.getType()) {
        Object param = method.convertArgsToSqlCommandParam(args);
        result = rowCountResult(sqlSession.update(command.getName(), param));
    } else if (SqlCommandType.DELETE == command.getType()) {
        Object param = method.convertArgsToSqlCommandParam(args);
        result = rowCountResult(sqlSession.delete(command.getName(), param));
    } else if (SqlCommandType.SELECT == command.getType()) {
        if (method.returnsVoid() && method.hasResultHandler()) {
            executeWithResultHandler(sqlSession, args);
            result = null;
        } else if (method.returnsMany()) {
            result = executeForMany(sqlSession, args);
        } else if (method.returnsMap()) {
            result = executeForMap(sqlSession, args);
        } else {
            Object param = method.convertArgsToSqlCommandParam(args);
            result = sqlSession.selectOne(command.getName(), param);
        }
    } else {
        throw new BindingException("Unknown execution method for: " + command.getName());
    }
    if (result == null && method.getReturnType().isPrimitive() && !method.returnsVoid()) {
        throw new BindingException("Mapper method '" + command.getName()
            + " attempted to return null from a method with a primitive return type (" + method.getReturnType() + ").");
    }
    return result;
}
```

在创建之初，command通过sqlSession封装了一些配置信息。

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

4. selectOne其实调用了selectList，只不过是取了第一个。具体代码如下：

```
public <T> T selectOne(String statement, Object parameter) {  
    // Popular vote was to return null on 0 results and throw exception on too many.  
    List<T> list = this.<T>selectList(statement, parameter);  
    if (list.size() == 1) {  
        return list.get(0);  
    } else if (list.size() > 1) {  
        throw new TooManyResultsException("Expected one result (or null) to be returned");  
    } else {  
        return null;  
    }  
}
```

5. selectList经过层层调用，最终交给执行器执行。具体执行器的结构待会我们会分析。注意这里的ms参数，其实就是从Configuration中得到的一些配置信息，包括mapper文件里的sql语句。具体代码如下：

```
public <E> List<E> selectList(String statement) {
    return this.selectList(statement, null);
}

public <E> List<E> selectList(String statement, Object parameter) {
    return this.selectList(statement, parameter, RowBounds.DEFAULT);
}

public <E> List<E> selectList(String statement, Object parameter, RowBounds rowBounds) {
    try {
        MappedStatement ms = configuration.getMappedStatement(statement);
        List<E> result = executor.query(ms, wrapCollection(parameter), rowBounds, Executor.NO_RESULT_HANDLER);
        return result;
    } catch (Exception e) {
        throw ExceptionFactory.wrapException("Error querying database. Cause: " + e, e);
    } finally {
        ErrorContext.instance().reset();
    }
}
```

6. 这里的执行器execute，其实是spring注入的。excute是一个接口，而到时候具体是哪个execute执行，是看配置文件的。而我们的一级缓存和二级缓存其实都是execute中的一种。接下来，我们遍分析一下执行器（Executor）。

2.2 执行器（Executor）

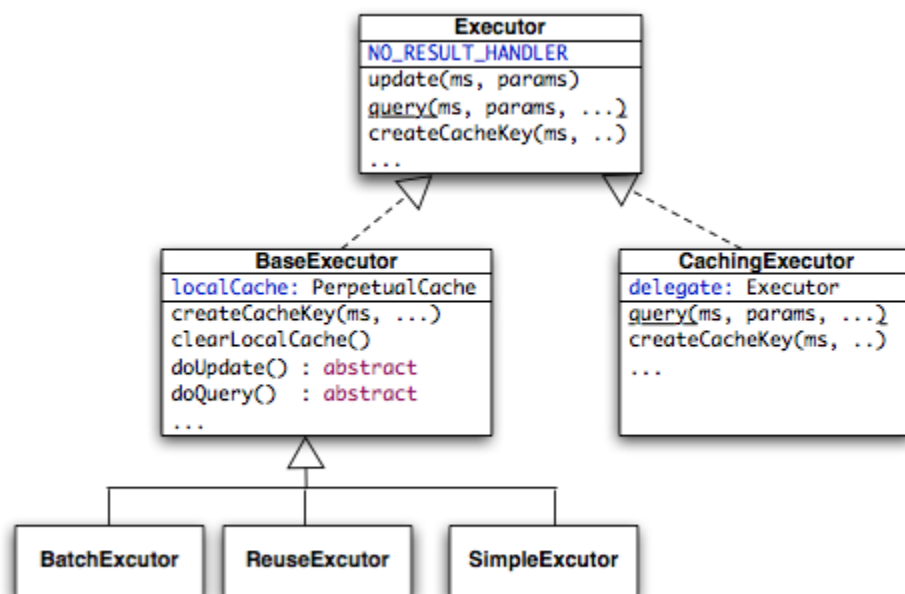
Executor:执行器接口。也是最终执行数据获取及更新的实例。其结构如下：

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册





1. BaseExecutor:基础执行器抽象类。实现一些通用方法，如createCacheKey之类的。并采用模板模式将具体的数据库操作逻辑交由子类实现。另外，可以看到变量localCache:PerpetualCache,在该类采用perpetualCache实现基于map存储的一级缓存，其query方法如下：

```

@SuppressWarnings("unchecked")
public <E> List<E> query(MappedStatement ms, Object parameter, RowBounds rowBounds, ResultHandler resultHandler, CacheKey key, BoundSql boundSql) throws SQLException {
    ErrorContext.instance().resource(ms.getResource()).activity("executing a query").object(ms.getId());
    if (closed) throw new ExecutorException("Executor was closed.");
    if (queryStack == 0 && ms.isFlushCacheRequired()) {
        clearLocalCache();
    }
    List<E> list;
    try {
        queryStack++;
        list = resultHandler == null ? (List<E>) localCache.getObject(key) : null;
        if (list != null) {
            handleLocallyCachedOutputParameters(ms, key, parameter, boundSql);
        } else {
            list = queryFromDatabase(ms, parameter, rowBounds, resultHandler, key, boundSql);
        }
    } finally {
        queryStack--;
    }
    if (queryStack == 0) {
        for (DeferredLoad deferredLoad : deferredLoads) {
            deferredLoad.load();
        }
        deferredLoads.clear(); // issue #601
        if (configuration.getLocalCacheScope() == LocalCacheScope.STATEMENT) {
            clearLocalCache(); // issue #482
        }
    }
    return list;
}

```

一级缓存，Map存储实现

一级缓存和二级缓存很相似，都是实现Cache缓存接口，然后等待调用。其中的一级缓存具体实现其实使用了Map存储，原理非常简单。PerPetualCache具体结构如下：

```

public class PerpetualCache implements Cache {

    private String id;

    private Map<Object, Object> cache = new HashMap<Object, Object>();

    private ReadWriteLock readWriteLock = new ReentrantReadWriteLock();

```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

```
    this.id = id;
}

public String getId() {
    return id;
}

public int getSize() {
    return cache.size();
}

public void putObject(Object key, Object value) {
    cache.put(key, value);
}

public Object getObject(Object key) {
    return cache.get(key);
}

public Object removeObject(Object key) {
    return cache.remove(key);
}

public void clear() {
    cache.clear();
}

public ReadWriteLock getReadWriteLock() {
    return readWriteLock;
}
```

缓存接口，实现二级缓存的时候，也必须实现该接口。其中，有7个抽象方法需要实现。

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

2. BatchExcutor、ReuseExcutor、SimpleExcutor:这三个就是简单的继承了BaseExcutor，实现了doQuery、doUpdate等发放，同样都是采用了JDBC对数据库进行操作；三者的区别在于，批量执行、重用Statement执行、普通方法执行。具体应用及长江在mybatis文档上都有详细的说明。
3. CachingExcutor:二级缓存执行器。其中使用了静态代理模式，当二级缓存中没有数据的时候，就使用BaseExecutor做代理，进行下一步执行。具体代码如下：

```
public <E> List<E> query(MappedStatement ms, Object parameterObject, RowBounds rowBounds, ResultHandler resultHandler, CacheKey key, BoundSql boundSql) throws SQLException {
    Cache cache = ms.getCache();
    if (cache != null) {
        flushCacheIfRequired(ms);
        if ((ms.isUseCache() && resultHandler == null) | {
            ensureNoOutParams(ms, parameterObject, boundSql);
            if (!dirty) {
                cache.getReadWriteLock().readLock().lock();
                try {
                    @SuppressWarnings("unchecked")
                    List<E> cachedList = (List<E>) cache.getObject(key);
                    if (cachedList != null) return cachedList;
                } finally {
                    cache.getReadWriteLock().readLock().unlock();
                }
            }
        }
        List<E> list = delegate.<E> query(ms, parameterObject, rowBounds, resultHandler, key, boundSql);
        tcm.putObject(cache, key, list); // issue #578. Query must be not synchronized to prevent deadlocks
        return list;
    }
    return delegate.<E> query(ms, parameterObject, rowBounds, resultHandler, key, boundSql);
}
```

这里的ms是封装了配置型的，可以从中得到Cache对象

二级缓存的锁

调用实现方法，得到缓存数据

当没有缓存对象，即为null的时候，或者二级缓存中得不到数据的时候，就会把执行任务交给代理对象，这里的代理对象其实就是BaseExecutor

2.3 Cache的设计

像之前所说，Cache是一个缓存接口，运行时用到的其实是在解析mapper文件的时候根据配置文件生成的对应Cache实现类。另外这个实现类的构造过程使用了建造者（Builder）模式。在build的过程中，将所有设计到的cache放入基础缓存中，并使用装饰器模式将cache进行装饰。具体设计如下：

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

3 2. useNewCache方法中使用了建造者（Builder）模式，将从配置文件中读取出来的各个元素组装起来。其中最主要的是build方法。

```
private void cacheElement(XNode context) throws Exception {
    if (context != null) {
        String type = context.getStringAttribute("type", "PERPETUAL");
        Class<? extends Cache> typeClass = typeAliasRegistry.resolveAlias(type);
        String eviction = context.getStringAttribute("eviction", "LRU");
        Class<? extends Cache> evictionClass = typeAliasRegistry.resolveAlias(eviction);
        Long flushInterval = context.getLongAttribute("flushInterval");
        Integer size = context.getIntAttribute("size");
        boolean readWrite = !context.getBooleanAttribute("readOnly", false);
        Properties props = context.getChildrenAsProperties();
        builderAssistant.useNewCache(typeClass, evictionClass, flushInterval, size, readWrite, props);
    }
}
```

3. 在build方法中，值得注意的是使用了装饰器模式，将几个基本的Cache装饰了一下。因为我们的Cache只是加入了自定义的缓存功能和逻辑，日志功能、同步功能等其实并没有。所以需要装饰一下，具体代码如下：


```
public Cache useNewCache(Class<? extends Cache> typeClass,  
    Class<? extends Cache> evictionClass,  
    Long flushInterval,  
    Integer size,  
    boolean readWrite,  
    Properties props) {  
    typeClass = valueOrDefault(typeClass, PerpetualCache.class);  
    evictionClass = valueOrDefault(evictionClass, LruCache.class);  
    Cache cache = new CacheBuilder(currentNamespace)  
        .implementation(typeClass)  
        .addDecorator(evictionClass)  
        .clearInterval(flushInterval)  
        .size(size)  
        .readWrite(readWrite)  
        .properties(props)  
        .build();  
    configuration.addCache(cache);  
    currentCache = cache;  
    return cache;  
}
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

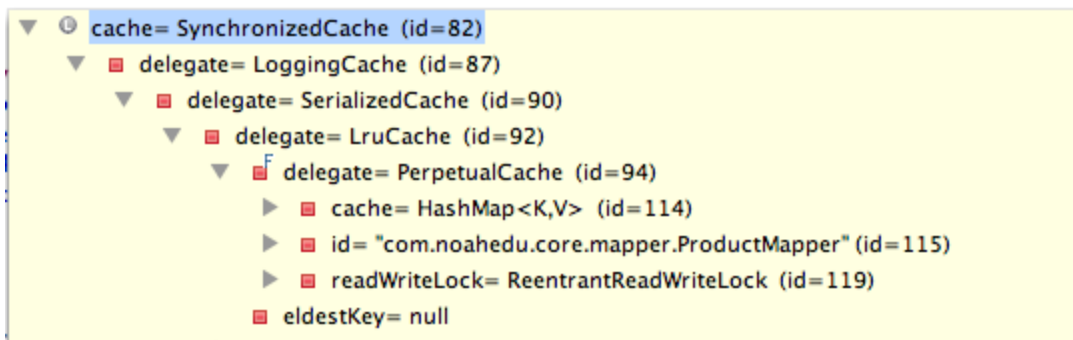
注册

×

```
public Cache build() {
    setDefaultImplementations();
    Cache cache = newBaseCacheInstance(implementation, id);
    setCacheProperties(cache);
    // issue #352, do not apply decorators to custom caches
    if (cache.getClass().getName().startsWith("org.apache.ibatis")) {
        for (Class<? extends Cache> decorator : decorators) {
            cache = newCacheDecoratorInstance(decorator, cache);
            setCacheProperties(cache);
        }
        cache = setStandardDecorators(cache);
    }
    return cache;
}
```

标准装饰器模式进行装饰

4. 最终的缓存实例对象结构：



加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



总体上看，我们可以把MyBatis关于缓存的这一部分分为三个部分：

1. 解析器：结合mybatis-spring框架，读取spring关于mybatis的配置文件。具体看是否开启缓存（这里指二级缓存），如果开启，生成的执行器为Caching Executor。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <settings>
    <!-- 开启缓存 -->
    <setting name="cacheEnabled" value="true"/>
    <!-- 是否启用 数据中 a_column 自动映射 到 java类中驼峰命名的属性。[默认:true] -->
    <setting name="mapUnderscoreToCamelCase" value="true" />
    <!-- 开启全局性设置懒加载 -->
    <setting name="lazyLoadingEnabled" value="true"/>
    <!-- 开启按需加载 -->
    <setting name="aggressiveLazyLoading" value="false"/>
  </settings>
  <!-- 别名 -->
  <typeAliases>
    <!-- <typeAlias alias="TfUser" type="com.tfdd.model.TfUser"/> -->
    <package name="com.tfdd.model"/>
    <package name="com.tfdd.dto"/>
  </typeAliases>

  <!-- 映射map 已在spring-datasource作了映射这里重复，否则报错 -->
  <mappers>
    <!-- <package name="mapper"/> -->
  </mappers>
</configuration>
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



```
public Executor newExecutor(Transaction transaction, ExecutorType executorType, boolean autoCommit) {
    executorType = executorType == null ? defaultExecutorType : executorType;
    executorType = executorType == null ? ExecutorType.SIMPLE : executorType;
    Executor executor;
    if (ExecutorType.BATCH == executorType) {
        executor = new BatchExecutor(this, transaction);
    } else if (ExecutorType.REUSE == executorType) {
        executor = new ReuseExecutor(this, transaction);
    } else {
        executor = new SimpleExecutor(this, transaction);
    }
    if (cacheEnabled) {
        executor = new CachingExecutor(executor, autoCommit);
    }
    executor = (Executor) interceptorChain.pluginAll(executor);
    return executor;
}
```

2. 动态代理：实现调用mapper接口的时候执行mybatis逻辑
3. 执行器：执行缓存处理逻辑。在这里二级缓存和一级缓存有所区别。

三、具体实现

3.1 配置文件

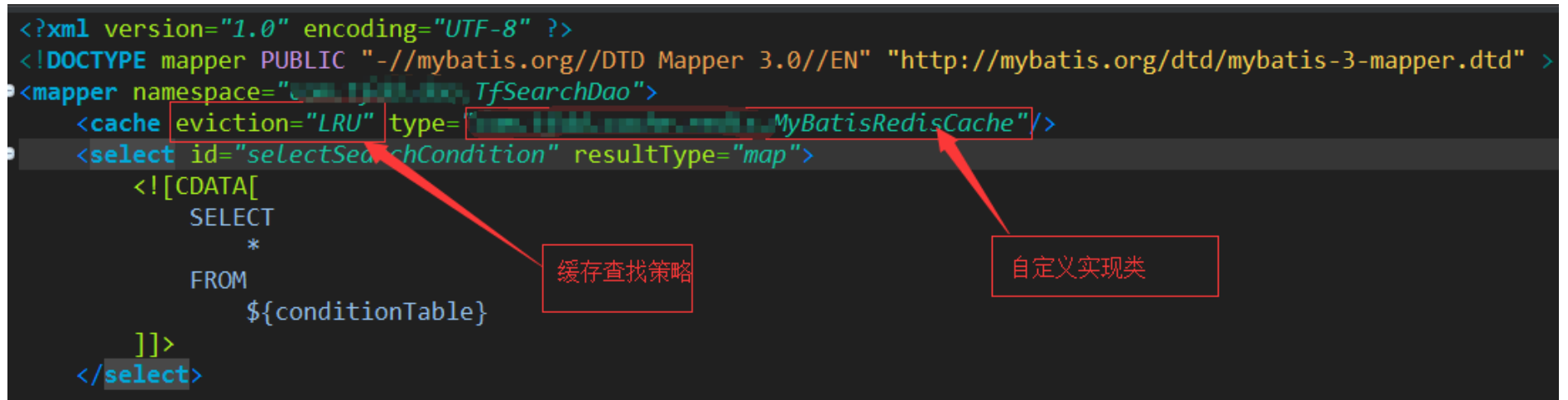
加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



1. 开启缓存：修改spring中关于mybatis的配置文件，将cacheEnabled设置为true。



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.muxiaocao.mybatis.mapper.TfSearchDao">
  <cache eviction="LRU" type="com.muxiaocao.mybatis.cache.MyBatisRedisCache"/>
  <select id="selectSearchCondition" resultType="map">
    <![CDATA[
      SELECT
        *
      FROM
        ${conditionTable}
    ]]>
  </select>
</mapper>
```

The image shows an XML configuration for a MyBatis mapper. Two red arrows point to specific attributes in the `<cache>` tag. One arrow points to the `eviction="LRU"` attribute, with a red box labeled "缓存查找策略" (Cache search strategy). The other arrow points to the `type="com.muxiaocao.mybatis.cache.MyBatisRedisCache"` attribute, with a red box labeled "自定义实现类" (Custom implementation class).

2. 添加实现Cache接口的实现类。重写方法会在查询数据库前后调用，查询、更新、删除、创建缓存需要在这几个方法中实现。值得注意的是，getObject方法，当返回的是null时，就会接着查询。如果不为null，则返回，不再查询了。

```
1  /**
2   * 使用redis做mybatis二级缓存
3   * @Description
4   * @file_name MyBatisRedisCache.java
5   * @time 2016-07-26 下午4:49:13
6   * @author muxiaocao
7   */
8  public class MyBatisRedisCache implements Cache{
9
10     @Value("#{config['redis.ip']}")
11     protected String redisIp;
12 }
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

×

```
15
16 private static Log logger = LoggerFactory.getLog(MyBatisRedisCache.class);
17
18 private Jedis redisClient = createClient();
19
20 private final ReadWriteLock readWriteLock = new ReentrantReadWriteLock();
21
22 private String id;
23
24 public MyBatisRedisCache(final String id) {
25     if (id == null) {
26         throw new IllegalArgumentException("缓存没有初始化id");
27     }
28     logger.debug("=====MyBatisRedisCache:id=" + id);
29     this.id = id;
30 }
31
32 @Override
33 public String getId() {
34     return this.id;
35 }
36
37 @Override
38 public int getSize() {
39     return Integer.valueOf(redisClient.dbSize().toString());
40 }
41
42 @Override
43 public void putObject(Object key, Object value) {
44     logger.debug("=====pubObject:" + key + "=" + value);
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



```
47  @Override
48  public Object getObject(Object key) {
49      Object value = SerializeUtil.unserialize(redisClient.get(SerializeUtil.serialize(key.toString())));
50      logger.debug("=====getObject:" + key + "=" + value);
51      return value;
52  }
53
54  @Override
55  public Object removeObject(Object key) {
56      return redisClient.expire(SerializeUtil.serialize(key.toString()), 0);
57  }
58
59  @Override
60  public void clear() {
61      redisClient.flushDB();
62  }
63
64  @Override
65  public ReadWriteLock getReadWriteLock() {
66      return readWriteLock;
67  }
68
69  private Jedis createClient() {
70      try {
71          RedisUtil.initRedis(redisIp);
72          return RedisUtil.getRedis();
73      } catch (Exception e) {
74          e.printStackTrace();
75      }
76      throw new RuntimeException("初始化redis错误");
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



3. 在相应的Mapper文件中，加入Cache节点，将自定义的cache实现类加进去。

四、总结

MyBatis的整体思路其实很简单，但是跟着源码发现，一个好的框架需要考虑的问题很多，从可扩展性、功能维护等角度考虑，如果没有一个好的设计思路会把代码设计的很乱很乱。**MyBatis**充分利用了动态代理、建造模式、装饰器模式，使他们结合在一起，让整个框架变得简单易用，其实是很难得的。

这就好比读一本书，需要先读厚再度薄一样，框架的设计最开始需要考虑到各种各样的问题，然后把一个简单的思路变得复杂。然后通过合理的设计，将复杂的问题简单的设计出来，使得代码很整洁，易于维护和读，这才是一个好的框架应该有的样子。

真的很感谢能有这么一个机会研究一下mybatis，并从中学到了许多。希望有朝一日，也能写出像mybatis一样的框架。

在这里很非常感谢<http://denger.iteye.com/blog/1126423/>。

注意：转载请标明，转自itboy-木小草。
尊重原创，尊重技术。

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qq_25689397/article/details/52066179

文章标签： [系统架构](#) [分布式](#) [mybatis](#) [redis](#)

个人分类： [架构设计](#) [设计模式](#)

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

[登录](#)

[注册](#)



[查看更多>>](#)

作者介绍： 数据工程师，全栈开发工程师，...

想对作者说点什么？

[我来说一句](#)

Phil_Jing

2017-07-05 19:24:38 #2楼

[html]

```
01. <!-- Spring-redis连接池管理工厂 -->
02. <bean id="jedisConnectionFactory"
03.     class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">
04.     <!-- IP地址 -->
05.     <property name="hostName" value="${redis.host}" />
06.     <!-- 端口号 -->
07.     <property name="port" value="${redis.port}" />
08.     <!-- 超时时间 -->
09.     <property name="timeout" value="${redis.timeout}" />
10.     <property name="password" value="" />
11.     <property name="poolConfig" ref="poolConfig" />
12. </bean>
```

这么配置吗？



Phil_Jing

2017-07-05 15:47:26 #1楼

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

[登录](#)[注册](#)

UI/UE全栈设计师特训班

该课程系统的全面涵盖了电商美工、设计师所需的实战及基础知识，品牌形象设计，banner运营视觉设计，创意海报合成，字体搭配使用，Web端专题页面设计，三维画面表现设计,本课从基础到实践案例，都是小白成长过程中急需的“干货”最为适合学习UI的全面课程！

学院 2018年04月26日 14:41

SpringMVC + MyBatis + Mysql + Redis(作为二级缓存) 配置

项目环境： 在SpringMVC + MyBatis + Mysql。Redis部署在Linux虚拟机。1、整体思路 参考Ehcache实现MyBatis二级缓存代码（Maven引用对应jar查阅） ...

 xiadi934 2016-03-03 10:37:47 阅读数：38581

Mybatis学习(十四)mybatis框架下整合 分布式缓存ehcache - CSDN博客

分布式缓存不使用分布缓存,缓存的数据在各各服务单独存储,不方便系统 开发。所以要使用 分布式缓存对缓存数据进行集中管理。 分布式缓存工作图 mybatis 本身来说是无法...

2018-4-15

mybatis整合ehcache分布式缓存框架 - 的博客 - CSDN博客

mybatis提供了一个cache接口,如果要实现自己的缓存逻辑,实现cache接口开发即可。mybatis和ehcache整合,mybatis和ehcache整合包中提供了一个cache接口的实现类。1.4.3...

2018-1-25


加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



配置mybatis使用redis作为自定义缓存mybatis自身的缓存做的并不完美，但它提供了使用自定义缓存的机会，我们可以选择使用我们喜欢的自定义缓存，下面将介绍一下，使用redis作为mybati...

 juxianze9407

2017-12-06 19:30:26

阅读数：1238

Mybatis学习(十三)mybatis查询缓存理解 - CSDN博客

上一篇 [Mybatis学习\(十二\)mybatis理解动态sql及sql片段](#) 下一篇 [Mybatis学习\(十四\)mybatis框架下整合分布式缓存ehcache](#) 查看评论 [mybatis--查询缓存](#) [查询缓存一、my...](#)

2018-3-23

J2EE 分布式架dubbo+springmvc+mybatis+ehcache+redis 分布式架构

{精华}j2ee企业大型分布式架构 dubbo+springmvc+mybatis+ehcache+redis 分布式架构...spring-springmvc-mybatis-dubbo-redis-mysql实现soa搭建,数据查询,数据缓存 dubb...

2018-4-26

Redis之实战篇（与Mybatis整合）

1，准备好ssm工程，如果有不会的，可以参考 [springmvc+mybatis整合](#) 2，准备好Redis服务器 3，构建 pom.xml 文件，这个pom文件和之前ssm的基本一样，只是添加...

 xwnxwn

2016-12-27 14:09:29

阅读数：4658

mybatis+redis+mybatis-redis实现二级缓存

说明： 1、MyBatis默认开启二级缓存 2、MyBatis默认实现了自己的二级缓存（PerpetualCache），内部使用HashMap实现，无法实现分布式，并且服务器重启后就没有缓存了。 ...

 a327919006

2017-10-19 09:57:39

阅读数：923

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



的查询结果,在sqlseesion进行更新,删除操作时,*mybatis*会清空一级缓存,保持数据的...但是二级缓存不一定存在内存,而是多种存储介质,还可以是借助ehcache实现分布式存储...

2017-7-14

MyBatis+Redis缓存实现 - CSDN博客

1、整体思路参考Ehcache实现*MyBatis*二级缓存代码(Maven引用对应jar查阅) ... xiadi934 2016年03月03日 10:37 38082 分布式系统架构——使用Redis做*MyBatis*的二级...

2018-4-16

mybatis整合redis

mybatis默认缓存是PerpetualCache, 可以查看一下它的源码, 发现其是Cache接口的实现; 那么我们的缓存只要实现该接口即可。 该接口有以下方法需要实现: String getl...

 fhx007 2013-10-13 16:48:59 阅读数: 25634

mybatis——缓存 - CSDN博客

在*mybatis*中,缓存的使用是一个十分重要的过程,在项目查询配置过程中,有着很重要的作用。缓存分为一级缓存和二级缓存,一级缓存是默认缓存的。缓存*MyBatis* 包含一...

2018-3-25

Mybatis配置分布式缓存 - CSDN博客

分布式缓存 不使用分布缓存,缓存的数据在各各服务单独存储,不方便系统 开发。所以要使用分布式缓存对缓存数据进行集中管理。 分布式缓存工作图 *mybatis* 本身来说是无法...

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



mybatis-redis的使用

参考 官方文档 <http://mybatis.org/redis-cache/> demo <https://github.com/edwinkun/MybatisRedisExample1> ma...

 qzshiyongjie123 2015-12-09 11:13:02 阅读数：2687

Redis实现Mybatis的二级缓存

一、Mybatis的缓存 通大多数ORM层框架一样，Mybatis自然也提供了对一级缓存和二级缓存的支持。一下是一级缓存和二级缓存的作用于和定义。 1、一级缓存是SqlSession...

 fengshizty 2016-01-25 17:56:42 阅读数：11186

开发框架整合与搭建：spring boot+mybatis+jedis

<properties> <project.build.sourceEncoding>UTF-8</project.build....

 kang123488 2018-03-10 22:34:12 阅读数：31


SpringMVC+Spring+mybatis+redis项目从零开始--redis缓存策略和配置实现

<http://blog.csdn.net/a123demi/article/details/78284555>

 lppl010_ 2018-03-22 13:43:40 阅读数：29

除了用作缓存数据，Redis还可以做这些

除了用作缓存数据，Redis还可以做这些 Redis应该说是目前最受欢迎的NoSQL数据库之一了。Redis通常被作为缓存组件，用作缓存数据。不过，除了可以缓存数据，其实Redis可以做的事...

 huangshulang1234 2018-03-27 15:03:38 阅读数：117

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



1. pom.xml<?xml version="1.0" encoding="UTF-8"?><project xm...

 TylorMin 2018-03-02 10:59:42 阅读数：52

redis整合spring mybatis -- 缓存方案

上一篇总结了redis sentinel（哨兵方案）的配置流程，本篇就redis整合ssm框架进行说明。目前，大多数公司用redis主要做缓存用，对于那些不常变动的数据来说，我们将其缓存在redis中...

 mrleeapple 2017-11-20 17:58:33 阅读数：753

redis与Mybatis的无缝整合让MyBatis透明的管理缓存

在上一篇文中的Cahe类存在各种问题如：一直使用同一个连接，每次都创建新的Cache，项目中老是爆出connection timeout 的异常，存储的key过长等等一系列的问题，解决问题最好的办法就...

 xwnxwn 2016-12-27 14:14:58 阅读数：676

Spring+Redis+MyBatis实现缓存整合

 cx118118 2017-09-18 18:52:29 阅读数：1646

Mybatis自定义缓存——Redis实现

最近项目需用Redis来实现Mybatis缓存方案。mybatis默认缓存是PerpetualCache，可以查看一下它的源码，发现其是Cache接口的实现；那么我们的缓存只要实现该接口即可。 ...

 nmgrlq 2012-09-19 16:53:37 阅读数：11146

使用Redis做MyBatis的二级缓存

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



 qiuyinthree 2017-03-07 20:28:30 阅读数：1999

redis实现mybatis二级缓存

在前面的文章中已经看到了mybatis二级缓存的设置，这里我们重新学习一下，mybatis 的二级缓存可以使用mybatis 自身的hashMap实现二级缓存，而如果使用mybatis 自身的二级缓存...

 QH_JAVA 2016-05-23 21:53:30 阅读数：3648

Spring+ehcache+redis两级缓存--缓存实战篇（1）

在上篇《性能优化-缓存篇》中已经从理论上介绍了缓存，其实，缓存简单易用，更多精力关注的是根据实际业务的选择缓存策略。本文主要介绍为什么要构建ehcache+redis两级缓存？以及在实战中如何实现？思...

 liaoyulin0609 2016-06-30 01:09:56 阅读数：20813

高并发二级缓存的简单实现

高并发二级缓存的简单实现 我们的应用系统使用了两台Redis做缓存，一台持久化存储重要数据，另一台就是纯粹的缓存Mysql的数据。Redis是很强大，不过也有性能瓶颈的时候。官方公平的吞吐量是1...

 weinianjie1 2016-01-30 11:12:28 阅读数：3436

基于redis的缓存机制的思考和优化

相对我们对于redis的使用场景都已经想相当的熟悉。对于大量的数据，为了缓解接口（数据库）的压力，我们对查询的结果做了缓存的策略。一开始我们的思路是这样的。1.执行查询 2.缓存中存在数据 -> 查询...

 qq_18860653 2017-02-06 17:30:45 阅读数：10208

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



二级缓存是多个SqlSession共享的，其作用域是mapper的同一个namespace，不同的sqlSession两次执行相同namespace下的sql语句且向sql中传递参数也相同即最终执行相...

 xiaolyuh123

2017-06-29 15:27:12

阅读数：5208

spring boot + mybatis +redis 整合亲测可用

2017年09月15日

27KB

下载



spring boot学习4之mybatis+redis缓存整合

上篇博文学习了spring boot+mybatis+PageHelper分页插件的整合，在此基础上继续扩展，使用redis做数据库的二级缓存。对于用redis做数据库的缓存的必要性，就不多说了，特别...

 dream_broken

2017-05-21 19:42:42

阅读数：5525

redis mybatis spring整合

最近想在框架里面加入redis，替换原因呢其实也没有，就是单纯的想替换掉 ---维基百科：redis介绍 一般开发中用户状态使用session或者cookie，两种方式各种利弊。Ses...

 jinkun520

2016-06-30 10:20:51

阅读数：6819

SpringMVC+Spring+mybatis+redis项目从零开始--Spring mybatis mysql配置实现

SSM项目-Spring mybatis mysql配置实现 上一章我们把SSM项目结构已搭建（SSM框架web项目从零开始--分布式项目结构搭建）完毕，本章将实现Spring,mybatis,mys...

 a123demi

2017-01-19 15:10:33

阅读数：8725

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



上一篇总结了redis sentinel（哨兵方案）的配置流程，本篇就redis整合ssm框架进行说明。目前，大多数公司用redis主要做缓存用，对于那些不常变动的数据来说，我们将其缓存在redis中...

 donggang1992

2016-04-05 15:12:53

阅读数：19839

mybatis-redis项目分析

redis作为现在最优秀的key-value数据库，非常适合提供项目的缓存服务。把redis作为mybatis的查询缓存也是很常见的做法。在网上发现N多人是自己做的Cache，其实在mybatis的g...

 xiaoyu411502

2016-03-30 23:24:35

阅读数：1082

redis作为mybatis的二级缓存，此时二级缓存可以作为高并发缓存吗

处理并发问题的重点不在于你的设计是怎样的 而在于你要评估你的并发，并在并发范围内处理。你预估你的并发是多少，然后测试r+m是否支持。还有要纠正你下，缓存的目的是为了应对普通对象数据库的读写限...

 Fighting_YY

2017-08-30 14:32:56

阅读数：334

redis+mybatis+spring

redis的安装<http://liuyieyer.iteye.com/blog/2078093> redis的主从高可用 <http://liuyieyer.iteye.com/blog/20780...>

 ayanami001

2015-08-30 12:30:44

阅读数：614

项目实战Springmvc+Mybatis+Maven+CMS单点登录、分布式缓...

2018年01月16日

49B

下载



加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



分布式缓存 不使用分布缓存，缓存的数据在各各服务单独存储，不方便系统 开发。所以要使用分布式缓存对缓存数据进行集中管理。 分布式缓存工作图 myba
tis本身来说是无法实现分布式缓存的，所...


 CookSjg

2017-03-06 20:18:45

阅读数：385

Mybatis学习（十四）mybatis框架下整合分布式缓存ehcache

分布式缓存 不使用分布缓存，缓存的数据在各各服务单独存储，不方便系统 开发。所以要使用分布式缓存对缓存数据进行集中管理。 分布式缓存工作图 myba
tis本身来说是无法实现分布式缓存的，所以要与分...

 sun_aichao

2015-06-11 15:16:55

阅读数：7888

Mybatis无缝集成Memcached分布式缓存系统

1. 在pom.xml中加入 org.mybatis.caches mybatis-memcached 1.0.0 2. 在classpath目录下加入memcached....

 mxj588love

2016-05-27 15:23:01

阅读数：1921

Redis整合Mybatis

将Mybatis的缓存修改成redis缓存 将缓存和数据库分开

 air_stalh

2017-07-12 19:56:30

阅读数：208

springmvc+mybatis+redis完美整合

2016年05月19日

52.55MB

下载



spring,springmvc,mybatis整合redis，redis作为缓存使用

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



spring boot + Mybatis + redis 秒杀系统

最近开了一些高并发的东西，以及一些秒杀系统，但感觉都没有完整的描述。于是自己就动手实现了一个简单版本的抢购系统。本系统采用spring boot + mybatis + redis实现。项目结构图...

 feibabm 2017-09-19 09:18:41 阅读数：804


mybatis整合REDIS远程缓存

1.嵌入redis在pom文件添加一下代码 redis.clients jedis 2.8.1 2.编写redis连接池和配置文件public final class R...

 sinat_28963819 2016-11-30 11:56:26 阅读数：875


mybatis通过配置文件方式整合redis缓存，替换mybatis二级缓存

mybatis通过redis取代二级缓存，二级缓存的缺点不再赘述。mybatis默认缓存是PerpetualCache，可以查看一下它的源码，发现其是Cache接口的实现；那么我们的缓存只要实现该...

 yjl33 2015-12-25 11:01:39 阅读数：4053

Redis入门很简单之九【SpringMvc+Mybatis与redis整合让Mybatis管理缓存】

Redis入门很简单之九【SpringMvc+Mybatis与redis整合让Mybatis管理缓存】

 it_zhaonan 2016-03-17 15:17:17 阅读数：5464

redis做mybatis的二级缓存

1，mybatis的缓存首先我们要知道mybatis有一级缓存，和二级缓存的概念 1）、一级缓存是SqlSession级别的缓存。在操作数据库时需要构造 sqlSession对象 左对象由右一个

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



Mybatis使用Redis做二级缓存

由于redis是非关系型数据库，数据是储存到内存中的，而从内存中读取数据要比从硬盘中读取数据的速度要快很多，并且redis可以持久化数据，所以可以用redis做数据的缓存。首先导入jar包：Spri...

 CSDN3436

2017-04-10 17:19:12

阅读数：917

mybatis-redis-1.0.0-beta2.jar 用于整合mybatis和redis

2016年05月04日

19KB

下载



一级缓存和二级缓存的比较

Hibernate中提供了两级Cache，第一级别的缓存是Session级别的缓存，它是属于事务范围的缓存。这一级别的缓存由hibernate管理的，一般情况下无需进行干预；第二级别的缓存是Sess...

 xiewenbo

2012-10-26 10:30:22

阅读数：666

hadoop分布式缓存

分布式缓存一个最重要的应用就是在进行join操作的时候，如果一个表很大，另一个表很小很小，我们就可以将这个小表进行广播处理，即每个计算节点上都存一份，然后进行map端的连接操作，经过我的实验验证，这种...

 guoqiangma

2014-01-21 23:28:59

阅读数：1080

spring data jpa使用二级缓存

在用spring data jpa的过程中，采用了ehcache 来做缓存, 是否需要二级缓存，一般不需要，这得看业务的需要，因为这东西如果配置不好，反而会导致性能下降，但如果是有些数据，基本不改动， ...

hadoop中的分布式缓存——DistributedCache

分布式缓存一个最重要的应用就是在进行join操作的时候，如果一个表很大，另一个表很小很小，我们就可以将这个小表进行广播处理，即每个计算节点上都存一份，然后进行map端的连接操作，经过我的实验验证，这种...

 kingjinzi_2008 2012-07-12 17:49:33 阅读数：6244

spring+ehCache+redis多级缓存自定义实现

spring+ehcache+redis 多级缓存 ehcache redis 一起用

 haiyang4988 2017-01-07 15:35:11 阅读数：4052

SpringMVC+Spring+mybatis+Redis项目从零开始--分布式项目结构搭建

Springmvcspring mybatis maven mysql redis项目从零开始 一. SSM项目-目录框架搭建实现 1. 序言 做SSM项目有一段时间，从来没有系统性...

 a123demi 2017-01-18 16:33:13 阅读数：13690

没有更多推荐了，[返回首页](#)

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

[登录](#)

[注册](#)

