JavaBean对象属性拷贝之反射与Cglib

业务分层的时候通过除了Entity类之外还会有一个Dto或者Vo类，在相互传值的时候我们就需要进行一系列繁琐的setter/getter调用。而且当实体特别大的时候会导致代码繁琐，特别难读。不好维护。

所以我们需要自动化JavaBean对象属性拷贝这一过程。现提供三种解决方案及其性能对比：

1. 原生Java反射机制，原汁原味。
2. Cglib的BeanMap
3. Cglib的BeanCopier

|  |
| --- |
| <dependency>  <groupId>cglib</groupId>  <artifactId>cglib</artifactId>  <version>3.2.3</version>  </dependency> |

# 性能比较

|  |
| --- |
| public class Dto2Entity {  private static final Logger log = LoggerFactory.getLogger(Dto2Entity.class);  /\*\*  \* DTO对象转换为实体对象。如命名不规范或其他原因导致失败返回null  \*  \* @param t  \* 进行转换的对象  \* @param e  \* 进行转换的对象  \* @return 实体对象  \*/  public static <T, E> void transalte(T t, E e) {  Method[] tms = t.getClass().getDeclaredMethods();  Method[] tes = e.getClass().getDeclaredMethods();  for (Method m1 : tms) {  if (m1.getName().startsWith("get") || m1.getName().startsWith("is")) {  String mNameSubfix = "";  if (m1.getName().startsWith("get")) {  mNameSubfix = m1.getName().substring(3);  } else {  mNameSubfix = m1.getName().substring(2);  }  String forName = "set" + mNameSubfix;  for (Method m2 : tes) {  if (m2.getName().equals(forName)) {  // 如果参数类型一致，或者m1的返回类型是m2的参数类型的父类或接口  boolean canContinue = m1.getReturnType().isAssignableFrom(m2.getParameterTypes()[0]);  if (canContinue) {  try {  m2.invoke(e, m1.invoke(t));  break;  } catch (IllegalAccessException | IllegalArgumentException | InvocationTargetException e1) {  // TODO Auto-generated catch block  log.debug("DTO 2 Entity转换失败");  e1.printStackTrace();  }  }  }  }  }  }  }  public static void main(String[] args) {  TestDto a = new TestDto();  TestEntity b = new TestEntity();  TestDto c = new TestDto();  a.setId(1);  a.setSex("1");  a.setAge(18);  a.setHobby("nanren");  a.setName("dto");  long start = System.currentTimeMillis();  for (int i = 0; i < 10000 \* 1000; i++) {  //测试1  // transalte(a, b);  //测试2  // BeanMap map1 = BeanMap.create(a);  // BeanMap map2 = BeanMap.create(c);  // for (Object key : map1.keySet()) {  // if (map2.containsKey(key)) {  // map2.put(key, map1.get(key));  // }  // }  // //测试3  BeanCopier bc = BeanCopier.create(TestDto.class, TestDto.class, false);  bc.copy(a, c, null);  }  long end = System.currentTimeMillis();  System.out.println(end - start);  System.out.println(c);  }  }  性能结果：反射<BeanMap<BeanCopier(很快1000万数据1s以内)  容错性：反射最好。 |

# Dto2Entity工具类

|  |
| --- |
| public class BeanUtil {  // 使用多线程安全的Map来缓存BeanCopier，由于读操作远大于写，所以性能影响可以忽略  private static ConcurrentHashMap<String, BeanCopier> beanCopierMap = new ConcurrentHashMap<String, BeanCopier>();  private static final Object lock = new Object();  public static void copyProperties(Object source, Object target) {  String beanKey = generateKey(source.getClass(), target.getClass());  BeanCopier copier = beanCopierMap.get(beanKey);  if (copier == null) {  synchronized (lock) {  copier = beanCopierMap.get(beanKey);  if (copier == null) {  copier = BeanCopier.create(source.getClass(), target.getClass(), false);  beanCopierMap.putIfAbsent(beanKey, copier);// putIfAbsent已经实现原子操作了。  }  }  }  copier.copy(source, target, null);  }  private static String generateKey(Class<?> class1, Class<?> class2) {  return class1.toString() + class2.toString();  }    } |