缓存查询工具说明

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration({"classpath:testApplicationContext.xml"})
public class QueryCacheTestCase {

    // model 查询工具
    @Resource IdentifierQueryWrapper iqw;

    // map 查询工具
    @Resource MappingQueryWrapper mqw;

    @Test
    public void modelListTest(){

    // 查询之前需要创建查询上下文
    IdentifierQueryWrapper.QueryContext queryContext=new
IdentifierQueryWrapper.QueryContext(){

            @Override //查询缓存 key，不同查询不要重复
            public Object getQueryKey() {
                return "companyDealerQueryKey";
            }

            @Override//Count 查询回调，返回记录数
            public CountQueryCallback getCountQuery() {
                return new CountQueryCallback() {

                    @Override
                    public int execute() {
                        return 3;
                    }


                };
            }

            @Override//返回 id 列表，execute 参数由工具计算得出，只需要取从 start
开始的 size 条记录，start 从 0 开始计算
            public IdBlockQueryCallback getIdBlockQuery() {
                return new IdBlockQueryCallback() {

                    @Override
                    public List<Long> execute(int blockStart, int blockSize) {
```

```
                    List l = new ArrayList();
                    l.add(111L);
                    l.add(222L);
                    l.add(333L);
                    return l;
                }};
            }};

            // 查询结果返回
            MutablePagedList<Object> pl = iqw.identifiers(queryContext, 0, 2);
            System.out.println(pl.getItems());
            Assert.assertEquals(2, pl.getItems());
    }

    @Test
    public void mapListTest(){

        // 查询之前需要创建查询上下文
    MappingQueryWrapper.QueryContext queryContext=new
MappingQueryWrapper.QueryContext(){

            @Override//查询缓存 key，不同查询不要重复
            public Object getQueryKey() {
                return "companyDealerMapQueryKey";
            }

            @Override//Count 查询回调，返回记录数
            public CountQueryCallback getCountQuery() {
                return new CountQueryCallback() {

                    @Override
                    public int execute() {
                        return 3;
                    }


                };
            }

            @Override//返回 id 列表，execute 参数由工具计算得出，只需要取从 start
开始的 size 条记录，start 从 0 开始计算
            public IdBlockQueryCallback getIdBlockQuery() {
                return new IdBlockQueryCallback() {
```

```
                @Override
                public List<Long> execute(int blockStart, int blockSize) {
                    List l = new ArrayList();
                    l.add(111L);
                    l.add(222L);
                    l.add(333L);
                    return l;
                }};
            }

            @Override//根据 id 加载一条记录
            public LoadQueryCallback getLoadQuery() {


                return new LoadQueryCallback() {

                    @Override
                    public Map<String, Object> execute(long id) {
                        Map m = new java.util.HashMap();
                        if (id==111L) {
                            m.put("name", "yellow");
                            m.put("sex", "male");
                            return m;
                        }
                        if (id==222L) {
                            m.put("name", "red");
                            m.put("sex", "female");
                            return m;
                        }
                        else throw new RuntimeException("should not run here");
                    }

                };
            }};

            // 查询结果返回
            PagedList<Map<String, Object>> pl = mqw.records(queryContext, 0,
2);
            System.out.println(pl.getItems());
            Assert.assertEquals(2, pl.getItems());
    }
}
```