**PKLormapping 1.1 框架介绍**

1. **特性**

* **支持面向对象，零SQL操作数据库**
* **支持模拟器版本和真机版本**
* **支持自定义SQL扩展**
* **支持1级缓存，万级数据可达到百倍速度获取**
* **支持c中基础类型映射，推荐使用NSNumber进行封装**
* **支持分页查询，排序**
* **支持线程池，默认为1，可自定义**
* **支持多线程并发处理，线程安全**
* **支持HQL语句**
* **支持多表联动查询映射**
* **支持事务处理机制**
* **支持批量插入和更新**
* **目前不支持join连接方式自动补全**

**1.1版本**

**修复：**

* **属性空值SQL映射失败BUG**
* **修复使用过程中数据内存不释放问题**

**新增：**

* **新增多表联动查询功能**
* **新增事务处理机制**
* **新增批量处理API，实测万级数据插入快3倍以上**

**优化：**

* **优化1级缓存，减少内存压力**

1. **接口说明**
2. PKLormappingSwift ：框架头文件
3. PKHQLer ：HQL帮助类，负责组装条件查询，分页和排序

)

1. PKQueryPage ：分页类，负责分页

init(rows:Int,page:Int) —— 初始化方法

var rows:Int = 0 //每页行数

var page:Int = 0 //第x页

1. PKDataBaseAccess：数据库访问线程

映射表命名规则

表名 ：T\_XXX\_XX

字段名 ：XXX\_XX

命名中数字不用下横线分割

映射对象命名规则

类名：驼峰命名方式，每个峰映射表名中的下横线，如：UserInfo -> T\_USER\_INFO

属性名：驼峰命名方式，每个峰映射字段中的下横线，如：userId -> USER\_ID

类型：int,float,double类型最好使用NSNumber 存储，否则无法识别是否为空 命名中数字不用下横线分割

/\*

\* 创建通用数据库的线程，并加入到线程池

\* sql为执行sql

\* delegate 通过代理返回结果集

\*/

func execute(sql:String,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个访问数据库的线程，并加入到线程池

\* obj 是注入对象，若请求是select需要添加注入对象

\* delegate 通过代理返回结果集

\* <T>为映射范型

\*/

func queryExecute<T:NSObject>(obj:T,hql:PKHQLer,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个插入数据库的线程，并加入到线程池

\* obj 是注入对象

\* delegate 通过代理返回结果集

\* <T>为映射范型

\*/

func insertExecute<T:NSObject>(obj:T,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个修改数据库的线程，并加入到线程池

\* obj 是注入对象

\* delegate 通过代理返回结果集

\* <T>为映射范型

\*/

func updateExecute<T:NSObject>(hql:PKHQLer,obj:T,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个删除数据库的线程，并加入到线程池

\* obj 是注入对象

\* delegate 通过代理返回结果集

\* <T>为映射范型

\*/

func deleteExecute<T:NSObject>(hql:PKHQLer,obj:T,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个查询总数数据库的线程，并加入到线程池

\* delegate 通过代理返回结果集

\* <T>为映射范型

\*/

func countExecute<T:NSObject>(hql:PKHQLer,obj:T,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个批量插入数据库线程，并加入线程池

\* batchArray 批量处理数据列表，元素必须为同类型对象

\* delegate 通过代理返回结果集

\*/

func batchInsertExecute(batchArray:NSArray,callBackTarget:PKDataAccessDelegate?)

/\*

\* 创建一个批量插入数据库线程，并加入线程池

\* batchArray 批量处理数据列表，元素必须为同类型对象

\* delegate 通过代理返回结果集

\*/

func batchUpdateExecute(hql:PKHQLer,batchArray:NSArray,callBackTarget:PKDataAccessDelegate?)

/\*

\* 关闭数据库线程

\*/

func closeDataBase()

1. PKDataAccessDelegate ：数据回调接口
2. **Demo**
3. **获取访问线程**

**框架文件引用：**import PKLormappingSwift

lazy var dataBaseAccess:PKDataBaseAccess = PKDataBaseAccess.shareAccess("数据库名")

回调必须遵循 PKDataAccessDelegate 协议

1. **Query查询/分页/联动查询**

//查询映射对象

var user:UserInfo = UserInfo()

/\*

\* 联表映射时联动对象必须为PKArray类型

\* 联动查询时必须注明PKArray对象内的属性关联字段foreginKeyMapping，否则会异常

\* SQL联动查询条件为 where 从表的userId字段 ＝ 主表的id的值

\*/

user.bookAddress.foreginKeyMapping = ["id":"userId"]

//HQL辅助类

var hql:PKHQLer = PKHQLer()

//分页信息rows一页行数,page第x页

hql.queryPage = PKQueryPage(rows: 100, page: 0)

//查询条件，表示 SQL查询条件为 where user\_name = 'packyzhou'

hql.addEqual("USER\_NAME", value: "packyzhou")

//使用queryExecute进行线程访问数据库

dataBaseAccess.queryExecute(user,hql: hql, callBackTarget: self)

1. **新增**

var user:UserInfo = UserInfo()

user.id = 8888

user.userName = "packy-insert"

user.date = NSDate()

dataBaseAccess.insertExecute(user, callBackTarget: self)

1. **修改**

var user:UserInfo = UserInfo()

user.id = 8888

user.userName = "packyzhou"

user.date = NSDate()

//HQL辅助类

var hql:PKHQLer = PKHQLer()

//更新条件为where id = 8888

hql.addEqual("id", value: "8888")

dataBaseAccess.updateExecute(hql, obj: user, callBackTarget: self)

1. **删除**

var user:UserInfo = UserInfo()

//HQL辅助类

var hql:PKHQLer = PKHQLer()

//更新条件为where id = 8888

hql.addEqual("USER\_NAME", value: "packy-delete")

dataBaseAccess.deleteExecute(hql, obj: user, callBackTarget: self)

1. **批量处理**

//创建UserInfo类型的数组

var batchArray:NSMutableArray = []

for i in 100...1000 {

var userBatch:UserInfo = UserInfo()

userBatch.id = i

userBatch.userName = "zjw-update"

userBatch.date = NSDate()

batchArray.addObject(userBatch)

}

//使用批量插入处理线程（线程安全，事务机制）

dataBaseAccess.batchInsertExecute(batchArray, callBackTarget: self)

//使用批量更新处理线程（线程安全，事务机制）

dataBaseAccess.batchUpdateExecute(hql, batchArray: batchArray, callBackTarget: self)

**EntiyDemo:**

class UserInfo: NSObject {

var id:NSNumber?

var userName:NSString?

var age:NSNumber?

var sex:NSString!

var double:NSNumber?

var float:NSNumber?

var date:NSDate?

var bookAddress:PKArray = PKArray(entityNSObject: BookAddress())

var adr:PKArray = PKArray(entityNSObject: BookAddress())

}

**⚠注意**

1. **创建Entity映射对象时，类型必须为NSObject的子类，因为swift目前其他基础类型不能使用KVC**
2. **在联表查询时必须使用PKArray,且必须添加foreginKeyMapping 联动条件**
3. **Swift语言目前版本的映射能力非常差，大家尽量别尝试强行转用基础类型**
4. **此版本PKLormapping仅支持swift1.1版本-swift1.2版本，不支持swift2.0**
5. **反馈**

作者：周老师

邮箱：packyzhou@icloud.com

Blog: http://blog.csdn.net/packyzhou