数据库CURD操作的支持.

Muppet致力于把可以封装的所有能封装的,简化简单的sql操作,例如我们熟悉的insert,update,delete,这些增加删除,修改操作,每一个sql语句相对而言都是很简单的,为什么我们还要自己去写呢?按照sql语句的定义规则,我们可以根据用户的javabean完全有可能自动生成sql语句.

基本数据库操作的实现

1. 抽象数据库表结构,数据库表由基本的列,索引,外键等做成,sql生成,以及表的元数据信息,
2. 通过定义统一接口,Generate,中只有一个generate方法,任何需要生成Sql的类都需要实现此接口,使用最频繁的类包括TableGenerate,ColumnGenerate等,其中TableGenerate还需要借助于olumnGenerate接口
3. SqlGenerateHelper Sql语句生成器助手,统一管理了Generate接口的众多实现者,这个类的设计是简单工厂设计的体现,通过key值找到相应Generate实现类.
4. 由于不同增删改查需求的不同,执行上下文针对不同操作分别设计,但是由于JDBC的通用流程,一个标准的流程包括,从DataSourceUtil中获取连接,生成Sql语句,预编译Sql语句,记录Sql执行情况,执行Sql,包装结果集等,最终释放数据库连接,或者托管于连接管理器.
5. 针对不同的执行需求,有针对的设计了SelectContext, InsertContext

,UpdateCOntext,DeleteContext分别对应查询,添加,更新,删除上下文执行对应的Sql操作

Pojo向Scheme的映射

1.读取xml配置文件，解析获取实体类的包名（StandardResourceConfig）

2.加载指定包下的类，保存在映射（Map）中(StandardResourceLoader)   
3. 解析指定包下的所有的类，这里的解析主要是指解析实体类上的所有注解，根据框架预定义的各个注解的使用规则，优先顺序  
冲突处理，将解析结果保存起来（ResourceResolver）(Resolver解析分解)

4. 数据库检查，我们做的是对象向数据库表定义的映射，当然需要在解析配置  
加载资源，解析分解完毕之后拿着我们解析到的结果跟数据库进行比较  
这个比较的重任交给了DataBaseCheck以及他的两个内部类（TableCheck,ColumnCheck）,它主要负责检查表是否存在，列是 否存在，列的定义是否不同（主要检查，是否主键，是否为空，是否不同，默认值，是否索引，数据类型，数据类型的长度等）  
如何比较呢？需要调用JDBC的API,有直接调用即可返回结果的，有的需要混合使用。

5. Sql语句生成.主要由Generate接口负责，其下有很多实现类AlterColumnGenerate,AlterAddColumnGenerate,AlterColumnPrimaryKeyGenerate,TableGenerate  
分别实现生成列修改语句，列添加语句，修改主键语句，表语句生成。  
6.执行生成的Sql语句，通过Statement执行

Criteria实现原理

Criteria是单表查询工具,目前流行的hihernate,mybatis,框架都提供了类似的功能,通过这个功能,可以避免手工调试Sql语句,通过链式编程的方式完成Sql语句的拼装,这样更加安全,也能提高开发速度.Criteria的主要操作包括,等值比较,小于值比较,小于等于值比较,大于值比较,大于等于值比较,like模糊查找比较,Criteria的or合并,排序order操作,分页limit操作,等通过这些API,可以完全覆盖单张表的查询.Crieria的具体实现并不困难,只是Sql语句组合的一些繁琐.

1. 定义select.limit,order,where等状态值,判断当前Criteria是否已经定义了相关操作,如果已经定义后再次定义则抛出异常.如果还未定义,那么就按照用户输入保存.
2. 例如Select操作需要输入,查询列,那么就可以就把相关输入保存起来,当需要执行查询操作等执行Sql时,需要临时拼装Sql,buildSql方法就是通过判断相关Sql组成例如select,order,limit ,where等部分,生成Sql,并且将对应Value保存成数组
3. 将Sql和Value数组作为参数传递给SelectContext,SelectContext负责执行.将查询结果返回给Criteria.