高级设计意图

陶封邑 2020K8009937014

这里介绍H2database中体现的两种高级设计意图：工厂模式、策略模式，当然H2database在整个设计实现过程中还使用了其他设计意图。

工厂模式是一种创建设计模式，它在父类中提供了创建对象的接口，但允许子类更改将要创建的对象的类型。工厂方法模式建议将直接对象构造调用（使用 new 操作符）替换为对特殊工厂方法的调用。对象仍然是通过 new 操作符创建的，但它是从工厂方法内部调用的。

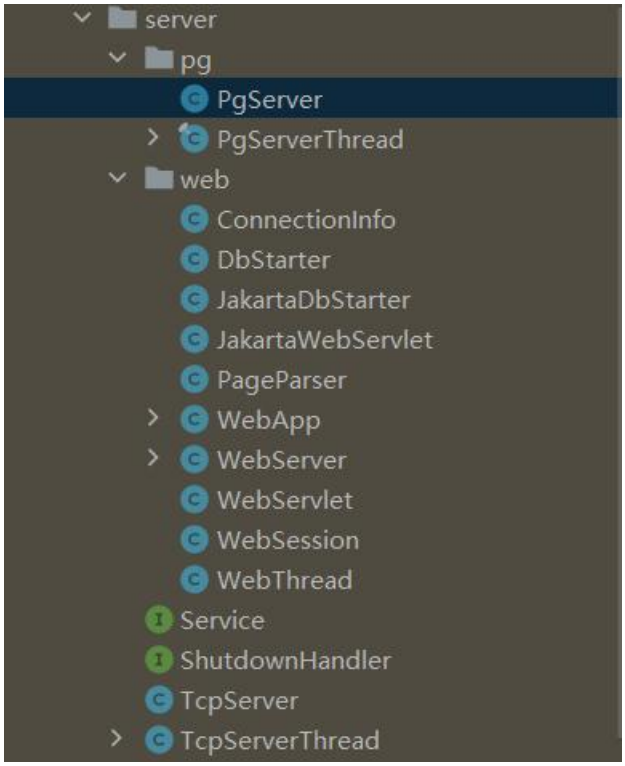
H2database 中最能体现工厂模式的是它对 Server 的实现。在 server 包里 H2database 首先实现了一个叫做 Service 的父类，定义了 init()、getURL()、start()、listen()、stop()、isRunning()、getAllowOthers()、getName()、getType()、getPort()、isDaemon()等方法，这些方法的具体实现则交由更具体的 Service 类型来完成。

```
package org.h2.server;

import java.sql.SQLException;

3 implementations
public interface Service {
```

H2database 中实现了三种不同的 Server，分别为 PgServer、WebServer、TcpServer，其中 Pg、Web、Tcp 表示三种网络连接方式，通过实现三种具体子类，就可以启动上面三个 Server。



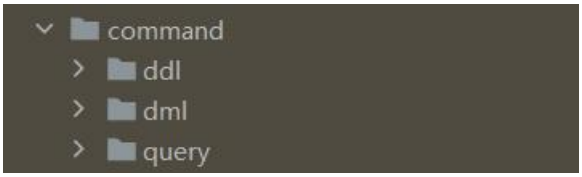
另外，启动 Server 的调用顺序是 init() → start() →listen()，用于启动的类 org.h2.tools.Server 类。

```java
public class Server extends Tool implements Runnable, ShutdownHandler {
    private final Service service;
    private Server web;
    private Server tcp;
    private Server pg;
    private ShutdownHandler shutdownHandler;
    private boolean started;

    public Server() { this.service = null; }
```

策略模式指针对一组算法，将每一个算法封装到具有共同接口的独立的类中，从而使得它们可以相互替换。策略模式使得算法可以在不影响到客户端的情况下发生变化。一个类定义了多种行为,并且这些行为在这个类的操作中以多个条件语句的形式出现。将相关的条件分支移入它们各自的类中以代替这些条件语句。

H2database 中体现策略模式的一点是它对 SQL 语句的解析过程。SQL 语句分为 DML 语句与 DDL 语句两种，两种语句的权限、语法等都不相同，因此 H2database 在实现 SQL 解析功能的时候将分析 DML 语句与 DDL 语句需要的类分别封装到两个包中。



比如 CREATE TABLE 语句对应 org.h2.command.ddl.CreateTable 类，INSERT 语句对应 org.h2.command.dml.Insert 类，SHOW 语句在 Parser 类中当成 SELECT 语句，对应 org.h2.command.dml.Select 类，相应的 CREATE TABLE 语句属于 DDL 语句，INSERT 语句与 SHOW 语句属于 DML 语句。

```java
public final class Insert extends CommandWithValues implements ResultTarget {
    private Table table;
    private Column[] columns;
    private Query query;
    private long rowNumber;
    private boolean insertFromSelect;
    private Boolean overridingSystem;
    private HashMap<Column, Expression> duplicateKeyAssignmentMap;
    private Value[] onDuplicateKeyRow;
    private boolean ignore;
    private ResultTarget deltaChangeCollector;
    private DataChangeDeltaTable.ResultOption deltaChangeCollectionMode;

    public Insert(SessionLocal var1) { super(var1); }

    public void setCommand(Command var1) {
        super.setCommand(var1);
        if (this.query != null) {
            this.query.setCommand(var1);
        }

    }
```