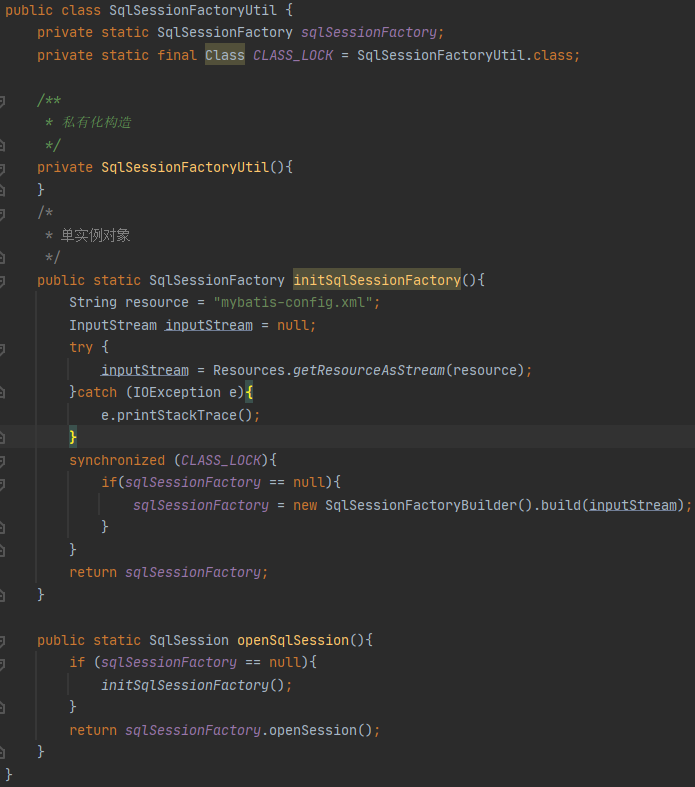
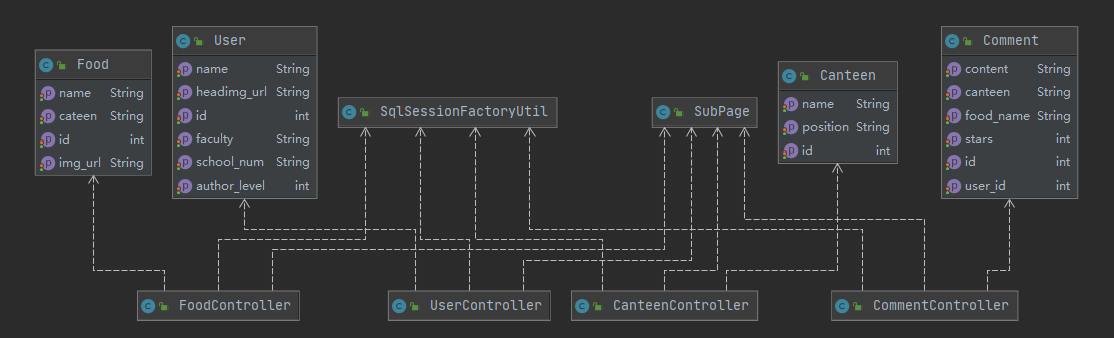
设计模式实验报告

1. **单例模式**
2. 相关代码
3. 设计思路分析

单例模式，即保证一个类只有一个实例，并提供一个访问它的全局访问点。

在数据库连接过程中，需要消耗大量资源，使用单例模式设计数据库连接类，可以节省系统资源，提高效率。

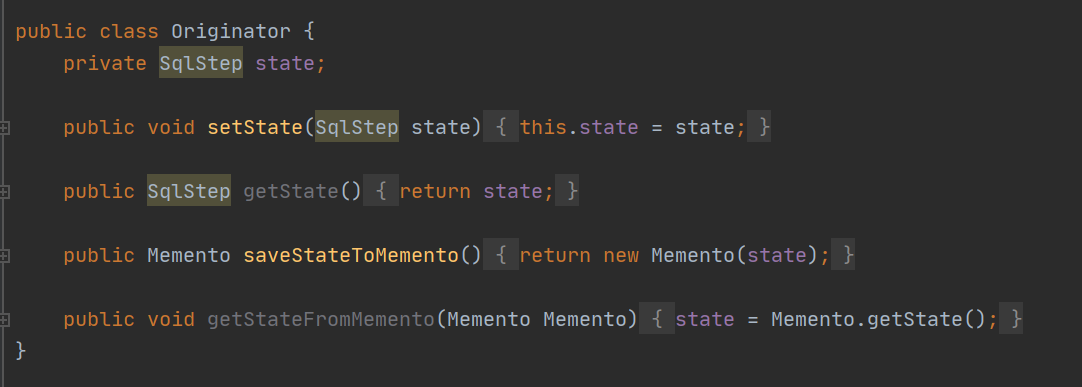
数据库中一共有四个实体类——Food，Canteen，Comment和User，分别用四个Controller来管理。这四个Controller共享一个数据库连接类（SqlSessionFactoryUtil），从而实现单例模式。SqlSessionFactoryUtil类是创建SqlSession的工厂，它的构造函数是私有的，只能自己创建自己的唯一实例。这样，四个Controller就可以重复使用这个唯一的实例来建立与数据库的连接。

1. 优势

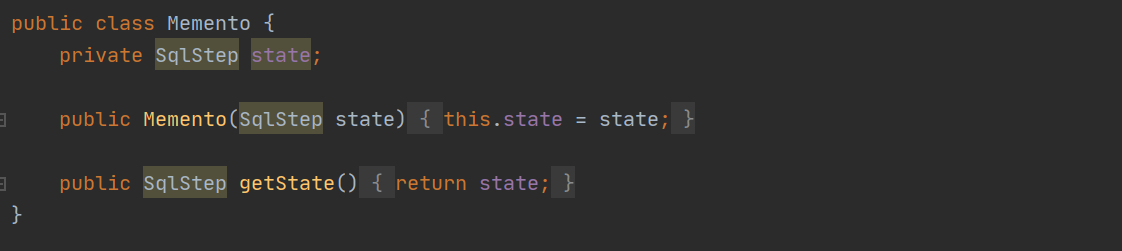
1）单例模式可以保证内存里只有一个实例，减少了内存的开销。

2）可以避免对资源的多重占用。

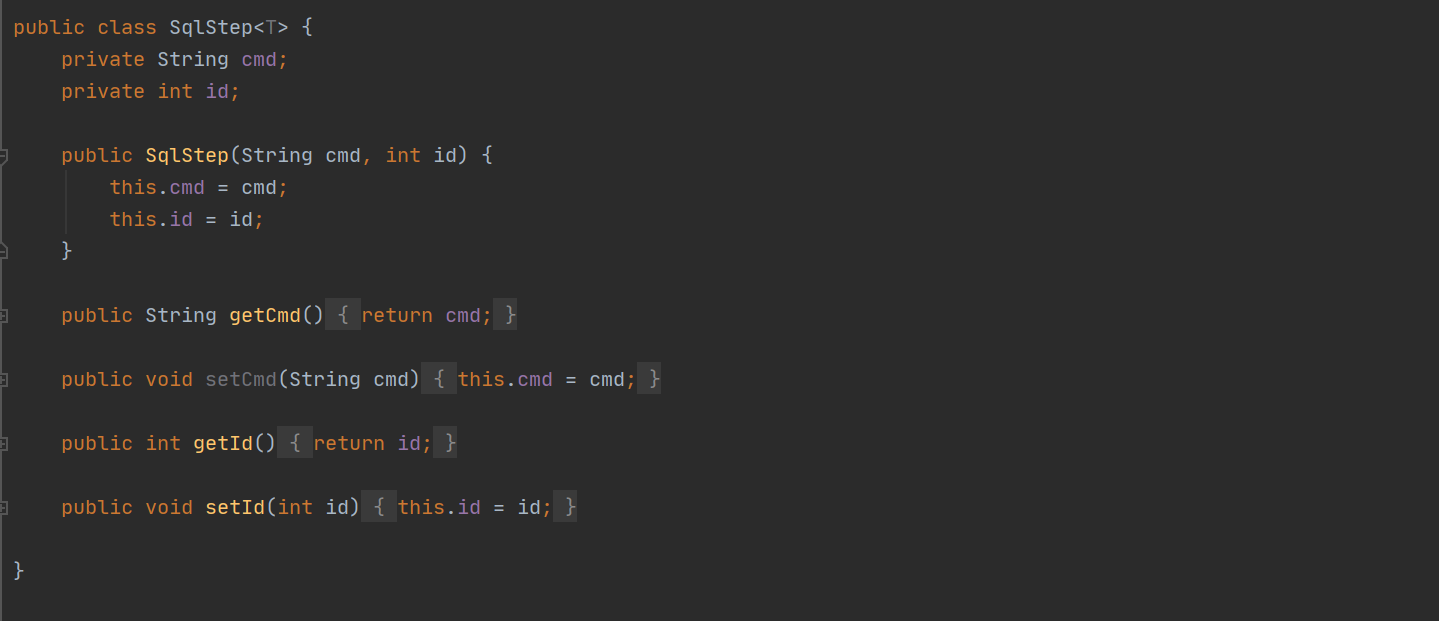
3）单例模式设置全局访问点，可以优化和共享资源的访问。

1. **备忘录模式**
2. 相关代码

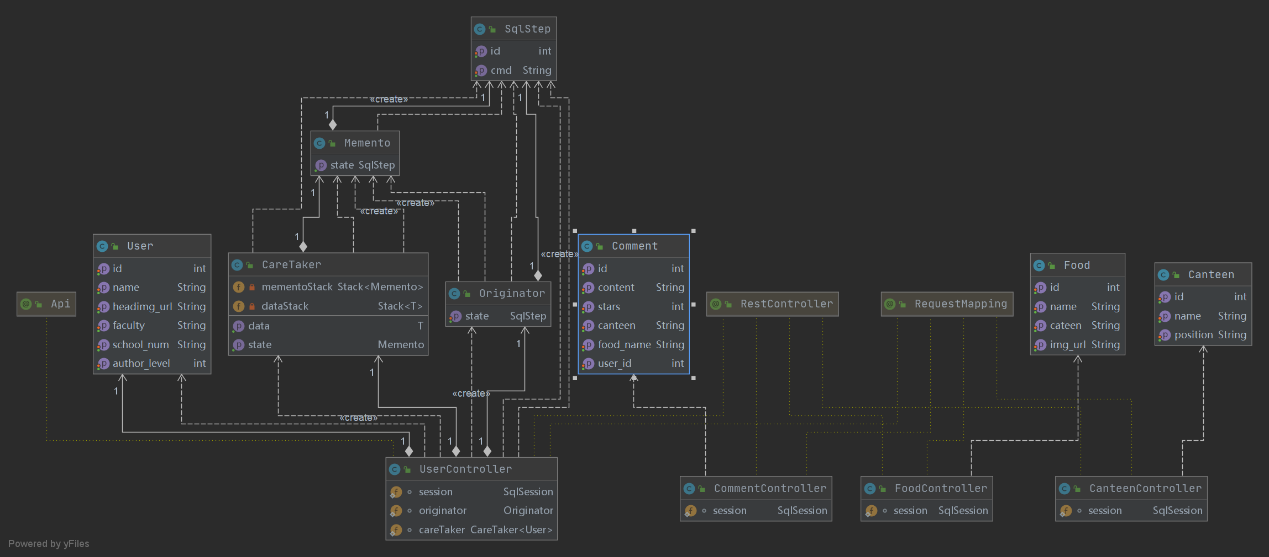
Originator类（原发器）



Memento类（备忘录）

CareTaker类（负责人）

SqlSteps类

1. 设计思路分析

使用备忘录模式，可以在不破坏封装的前提上，捕获对象的内部状态，并在该对象之外保存这个状态。对数据库的操作（SqlStep）记录到备忘录中，就可以在以后对其进行撤销，将其恢复到原先的状态。

Originator：可以创建一个备忘录，并存储它的当前状态。

Memento：存储一步SQL操作。

CareTaker：负责保存备忘录，但不能修改对象，也不需要知道对象的实现细节。

1. 优势

1）简化发起人实体类（Originator）的职责，隔离状态存储与获取，实现了信息的封装，客户端无须关心状态的保存细节。

2）提供状态回滚功能。可以撤销或恢复之前的操作。