工厂模式在分层架构中的应用

乐 艺
（南京广播电视大学，江苏 南京 210002）

摘 要：分析了工厂模式的特点，阐述了分层架构体系的设计思路。以数据访问层的设计为例，从设计模式的角度探讨了可复用的数据访问层的实现方法，并重点分析了工厂模式的具体应用过程。

关键词：分层架构；工厂模式；数据访问
中图分类号：TP311 **文献标识码**：A **文章编号**：1674-7720(2012)14-0088-02

Application of factory pattern in layered architecture

Le Yi
(Nanjing Radio and TV University, Nanjing 210002, China)

Abstract: This paper analyses the characteristics of factory pattern, elaborates the design thoughts of layered architecture, then on the basis of the data access layer, discusses the reusable data access layer as the implementation method from the design pattern, and selective analyses the specific application process of the factory pattern.

Key words: layered architecture; factory pattern; data access

随着时间的推移，软件需求不可避免地会有所改变，如何缩减软件开发和维护费用是开发设计人员共同面临的问题。分层架构在软件体系架构的设计中属于一种最为常见及重要的结构，分层架构的使用可以确保软件可维护、易修改。与此同时，软件的设计还必须是稳定的，在软件设计的过程中要尽量满足“开闭原则”，以达到提高可维护性的复用目的。“开闭原则”从某种角度论述就是“对可变性的封装原则”，即“找到一个系统的可变因素，将它封装起来”[1]。考虑到设计模式其实就是对不同可变性的封装[2]，因此，在分层架构的设计中，灵活运用设计模式可以使软件系统在不同程度上达到“开闭原则”的要求，从而更好地应对变化、提高复用性。

1 模式的概述

建筑大师 Christopher Alexander 最早提出了模式的概念，他认为“每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心”[3]，这一思想随即通过开创性著作[GOF95]引入到了软件领域。

模式的目标是要找出共通性问题的不变部分，必须在不断实践的过程中通过积累经验才能提取出其中的规则。在软件领域中，不同的层面上有不同的模式，从架构到实现，依次分为架构模式、设计模式、实现模式。如分层架构就是常见的架构模式，属于模式中的最高层次；而在[GOF95]一书中总结的 23 个基本设计模式则是用来处理设计中反复出现的问题，是模式中的第二层次；实现模式涉及到具体编程，也称为代码模式。

2 工厂模式的定义

创建型设计模式、结构型设计模式、行为型模式是设计模式的经典类型，其中创建型模式通过建立对象来解决问题，工厂模式就属于此类型中的常见种类。工厂模式通过专门负责实例化的工厂类来获得具体的对象[4]，由工厂动态地决定实例化哪一个类。工厂模式的形态主要有简单工厂模式、工厂方法模式、抽象工厂模式。简单工厂模式可以根据传入的参数决定创建哪一个类的实例，是不同的工厂模式在一定程度上的简化形式[5]。简单工厂模式的类图如图 1 所示，其中工厂类角色 Creator

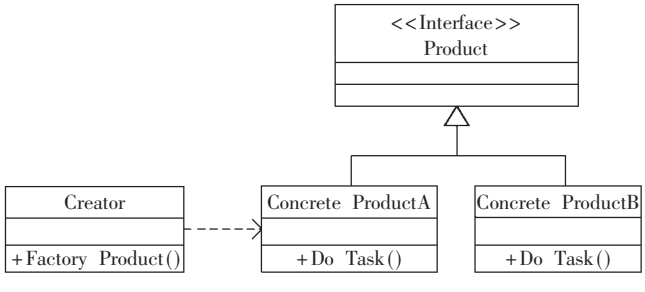

图 1 简单工厂模式类图

是简单工厂模式的核心，根据传入的参数创建产品对象；抽象产品角色 Product 是产品对象的共同接口；具体产品角色 Concrete Product 是一种接口的多种实现。

3 工厂模式的应用

3.1 分层架构体系的设计

区分层次的目的是实现系统的高内聚、低耦合，在分层架构的应用中，通常将系统划分为用户界面层（UI）、业务逻辑层（BLL）、实体层（Model）、数据访问层（DAL）。

以已经开发的教材管理系统为例，该系统的用户界面层为客户端提供对应用程序的访问，用于显示数据和接收用户输入的数据。业务逻辑层负责处理用户界面层的请求，实现应用程序的业务功能，其中用户界面层通过业务逻辑层来访问数据访问层，数据访问层提供数据服务。同时实体层被各层所调用，因为实体层是各层之间作为数据参数传递的结构通道。教材管理系统中的实体层比较简单，如 bookinfo 实体类，其代码主要由各字段的 get 和 set 方法组成，它对应的是数据库中的数据表，没有行为，只有属性，作为数据的载体可以被任一层的 book 类多次引用。

数据访问功能是分层设计的核心工作，在数据访问层中提供了与数据库的直接交流，如何设计一个灵活的、可扩展的数据访问层，使系统能方便地实现不同数据访问的迁移，是系统设计时要考虑的一个关键问题。

3.2 工厂模式在数据访问层中的应用

为使教材管理系统能实现多数据库支持，将简单工厂模式应用到数据访问层的设计中。简单工厂模式是一种简单、灵活的创建型模式，在分层结构下采用该设计模式不仅可以使软件系统的层次更分明，还能最大限度地实现软件复用，增强系统的可维护性和可扩展性。

在数据访问层中使用简单工厂模式如图 2 所示，其中 IdataProvider 属于数据访问层的接口子层，即这个类层次结构是针对数据库访问的接口。同时结合.Net 中的反射机制，调用该接口的具体实现类。在运行时，根据不同配置返回数据访问层中的不同实现，如 OracleDAL、ACCESSDAL 或 SQLServerDAL，其中创建具体接口类的对象的任务由 DALFactory 来实现，即工厂是提供转换好的接口的实例类。

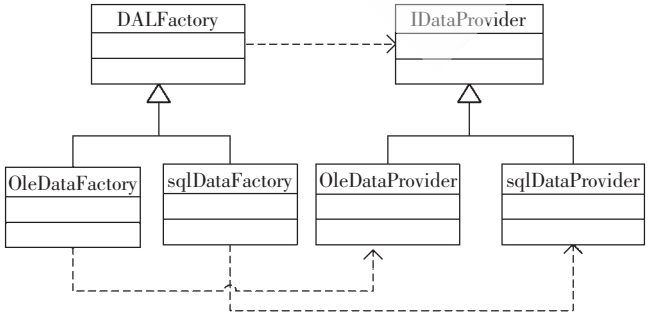

图 2 使用简单工厂模式的结构图

为实现数据访问层的可适应性，简单工厂模式为数据库提供者实现不更改程序代码的装配。首先，在 ASP.NET 中的配置文件 web.config 的 appSettings 节中配置所要创建的程序集中的对象；然后，利用工厂类 DALFactory 的 Create 方法读取该节从而获取数据库类型字符串；最后，创建相应的数据访问对象，通过父类 IdataProvider 引用返回。

3.3 数据访问层的具体实现

教材管理系统的数据访问层内部包含两个部分，一是修改自微软提供的 SQLHelper 类，该类提供了创建访问参数、命令对象、适配器以及执行存储过程和查询的方法，负责将存储在数据库中的数据公开，是真正面向数据库连接和访问的。另外一个才是面向实际项目的数据访问层，以 SQLServerDAL 的 book.cs 中的 UpdateBook 方法为例，这是与存储过程 Pr_UpdateBook 对应的修改教材的方法。首先通过 sqlHelper 类中的 CreateInParam 方法创建访问数据库的参数，然后执行 Pr_UpdateBook 存储过程，其中的核心是 sqlHelper.RunProc ("Pr_UpdateBook", paramList)，即通过 sqlHelper 类中的方法最终操作到了数据库中的表。可见，数据访问层起到了将存储过程对应到实际项目的作用。

但是教材管理系统的设计不能仅满足于运行在 SQLServer 之上，还要考虑到未来数据库的可移植性，因此将数据访问层具体实现的功能接口，包含对数据库的基本操作（如创建、增加、删除、选择、更新等）集中定义在数据访问层的接口子层。这些接口被抽象为一个单独的接口模块，提供了业务逻辑层访问数据的通道。正是由于接口模块的抽象性，同一个 Ibook 接口在数据访问层可以有多个实现，只要 OracleDAL、ACCESSDAL 或 SQLServerDAL 及其他的数据库访问层满足接口模块中定义的接口即可。

结合简单工厂模式的应用思路，仍以创建基于 SQLServer 数据库的接口实现类为例，在<appSettings>节中进行如下配置：<add key = "WebDAL" value = "Leyi.SQLServerDAL"/>；

在工厂类 DALFactory 中利用反射原理创建对象，先获取所配置的数据：

```
Private static string path =System.Web.Configuration.WebConfigurat ionManager.AppSettings["WebDAL"];
```

然后创建 dataProvider 对象：

```
public static Ibook Create()
{
string className = path+".book";
return (Ibook)Assembly.Load (path).CreateInstance (className);
}
```

其中，return(Ibook)Assembly.Load(path).CreateInstance

(className)是简单工厂模式的核心语句。这里的 Assembly.Load(path)是通过对 path 的定义把刚才的 SQLServerDAL 程序集动态载入;CreateInstance 方法是创建该程序集的实例;通过(Ibook)将该实例的类型强制转换为接口实例。

最后返回接口供业务逻辑层调用:

```
Ibook idap=DALFactory.Create();
```

由于用户可以根据参数获得对应的类实例，因此，采用简单工厂模式可以方便地实现其他数据库的装配，只要修改 web.config 配置节中的 value 值即可，无需修改类本身，这样不仅通过工厂避免了直接实例化类，还将业务逻辑层和数据访问层之间进行了解耦。

设计模式不仅能减少程序冗余度、提高程序效率，而且对将来的问题和需求也有足够的通用性，使得复用成功的设计和体系构造也变得更加简单方便。简单工厂模式作为重要的模式之一，带来了很多开发上的优点，实践证明在分层架构体系的设计中引入简单工厂模式，一定程度上保证了系统的可扩展性和可移植性，使得数据访问层更加灵活，即便简单工厂模式在某种程度上并不能完全满足“开闭原则”，但是仍然有效地实现了不同数据源的访问，满足了业务变更的需求，显著地改善了系统结构。

参考文献

[1] 阎宏.Java 与模式[M].北京:电子工业出版社,2002.

[2] GAMMA E,HELM R,JOHNSON R,et al.Design patterns:elements of reusable object-oriented software[M].Addison-Wesley Professional,1995.

[3] 亚历山大 C.建筑的永恒之道[M].赵冰,译.北京:知识产权出版社,2002.

[4] 李礁,李敏.基于工厂模式的易复用数据访问层设计[J].软件导刊,2011(3):8-9.

[5] 秦澎涛,王苏文.简单工厂模式在数据访问层中的应用[J].计算机工程与设计,2009,30(7):1799-1801.

(收稿日期:2012-03-16)

作者简介:

乐艺,女,1977年生,讲师,工程硕士,主要研究方向:软件工程。