个人整理 - Java 后端面试题 - 设计模式篇

作者：valarchie
原文链接：https://ld246.com/article/1583476977456
来源网站：链滴

● 标★号的知识点为重要知识点

java中有哪些代理模式?

最原始的静态代理。 以及JDK和Cglib的动态代理。

IO流熟悉吗，用的什么设计模式?

装饰器模式、适配器模式

懒汉式的单例模式如何实现单例?

通过双重检测以及synchronized,volatile关键字实现。

介绍一下策略模式?

通过定义一个策略接口，将具体的算法逻辑实现交给子类实现，源代码只需要调用策略接口就行。

说说你所熟悉或听说过的j2ee中的几种常用模式?

IO流的装饰器模式，Web过滤器的责任链模式，Spring的单例模式和工厂模式，
Spring中根据不同配置方式进行初始化的策略模式

★简述一下你了解的Java设计模式（总结）

标星号的为常用设计模式

★单例模式：保证某个类只能有一个唯一实例，并提供一个全局的访问点。
★简单工厂：一个工厂类根据传入的参数决定创建出那一种产品类的实例。
工厂方法：定义一个创建对象的接口，让子类决定实例化那个类。
抽象工厂：创建一组相关或依赖对象族，比如创建一组配套的汉堡可乐鸡翅。
★建造者模式：封装一个复杂对象的构建过程，并可以按步骤构造，最后再build。
★原型模式：通过复制现有的实例来创建新的实例，减少创建对象成本（字段需要复杂计算或者创建本高）。

★适配器模式：将一个类的方法接口转换成我们希望的另外一个接口。
★组合模式：将对象组合成树形结构以表示“部分-整体”的层次结构。（无限层级的知识点树）
★装饰模式：动态的给对象添加新的功能。
★代理模式：为对象提供一个代理以增强对象内的方法。
享元（蝇量）模式：通过共享技术来有效的支持大量细粒度的对象（Integer中的少量缓存）。
★外观模式：对外提供一个统一的方法，来访问子系统中的一群接口。
桥接模式：将抽象部分和它的实现部分分离，使它们都可以独立的变化（比如插座和充电器，他们之相插是固定的，
但是至于插座是插在220V还是110V，充电器是充手机还是pad可以自主选择）。

★模板方法模式：定义一个算法步骤，每个小步骤由子类各自实现。
解释器模式：给定一个语言，定义它的文法的一种表示，并定义一个解释器。
★策略模式：定义一系列算法，把他们封装起来，并且使它们可以相互替换。
★状态模式：允许一个对象根据其内部状态改变而改变它的行为。

★观察者模式：被观测的对象发生改变时通知它的所有观察者。
备忘录模式：保存一个对象的某个状态，以便在适当的时候恢复对象。
中介者模式：许多对象利用中介者来进行交互，将网状的对象关系变为星状的（最少知识原则）。
命令模式：将命令请求封装为一个对象，可用于操作的撤销或重做。
访问者模式：某种物体的使用方式是不一样的，将不同的使用方式交给访问者，而不是给这个物体。例如对铜的使用，造币厂
造硬币。雕刻厂造铜像，不应该把造硬币和造铜像的功能交给铜自己实现，这样才能解耦）
★责任链模式：避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连成一条链，
并且沿着这条链传递请求，直到有对象处理它为止。
迭代器模式：一种遍历访问聚合对象中各个元素的方法，不暴露该对象的内部结构。

单例模式的7种写法：懒汉2种，枚举，饿汉2种，静态内部，双重校验锁（推荐）。

- 懒汉式：懒加载，线程不安全

```
public class Singleton
{
    private static Singleton singleton;

    private Singleton()
    {
    }

    public static Singleton getInstance()
    {
        if (singleton == null)
            singleton = new Singleton();
        return singleton;
    }
}
```

- 懒汉式线程安全版：同步效率低

```
public class Singleton
{
    private static Singleton singleton;

    private Singleton()
    {
    }

    public synchronized static Singleton getInstance()
    {
        if (singleton == null)
            singleton = new Singleton();
        return singleton;
    }
}
```

- 饿汉式：

```
public class Singleton
{
    private static Singleton singleton = new Singleton();

    private Singleton()
    {
    }

    public static Singleton getInstance()
    {
        return singleton;
    }
}
```

- 饿汉式变种：

```
public class Singleton
{
    private static Singleton singleton;
    static
    {
        singleton = new Singleton();
    }

    private Singleton()
    {
    }

    public static Singleton getInstance()
    {
        return singleton;
    }
}
```

- 静态内部类方式:利用JVM的加载机制，当使用到SingletonHolder才会进行初始化。

```
public class Singleton
{
    private Singleton()
    {
    }

    private static class SingletonHolder
    {
        private static final Singleton singleton = new Singleton();
    }

    public static Singleton getInstance()
    {
        return SingletonHolder.singleton;
    }
}
```

- 枚举：

```
public enum Singletons
{
    INSTANCE;
    // 此处表示单例对象里面的各种方法
    public void Method()
    {
    }
}
```

- 双重校验锁：

```
public class Singleton
{
    private volatile static Singleton singleton;

    private Singleton()
    {
    }

    public static Singleton getInstance()
    {
        if (singleton == null)
        {
            synchronized (Singleton.class)
            {
                if (singleton == null)
                {
                    singleton = new Singleton();
                }
            }
        }
        return singleton;
    }
}
```

★在工作中遇到过哪些设计模式，是如何应用的

- 工厂模式（生产题型）。
- 策略模式（进行判题）。
- 模板方法模式（阅卷、判断题目信息是否正确，如条件1,2,3,三个条件分别由子类实现），
- 建造者模式（组装试卷生成器）
- 状态模式（根据试卷类型进行不同抽题）
- 适配器模式（适配其他微服务，类似防腐层）
- 外观模式（将一些使用较工具类封装简单一点）
- 代理模式（AOP切面编程）
- 责任链模式（推送、日志等额外操作）
- 组合模式（无限层级的知识点）

★在装饰器模式和代理模式之间的区别?

装饰器模式注重的是对类的扩展。

代理模式针对的是对类中方法的增强。

★Dubbo源码使用了哪些设计模式?

● 责任链模式：责任链中的每个节点实现Filter接口，然后由ProtocolFilterWrapper，将所有Filter串起来。

Dubbo的许多功能都是通过Filter扩展实现的，比如监控、日志、缓存、安全、telnet以及RPC本身都。

● 观察者模式：消费者在初始化的时候回调用subscribe方法，注册一个观察者，如果观察者引用的务地址列表发生改变，

就会通过NotifyListener通知消费者。

● 装饰器模式：比如ProtocolFilterWrapper类是对Protocol类的修饰。

● 工厂模式：如ExtenstionLoader.getExtenstionLoader(Protocol.class).getAdaptiveExtenstion()。

★Spring当中用到了哪些设计模式?

● 模板方法模式：例如jdbcTemplate，通过封装固定的数据库访问比如获取connection、获取statement，关闭connection、关闭statement等

然后将特殊的sql操作交给用户自己实现。

● 策略模式：Spring在初始化对象的时候，可以选择单例或者原型模式。

● 简单工厂：Spring中的BeanFactory就是简单工厂模式的体现，根据传入一个唯一的标识来获得bea对象。

● 工厂方法模式：一般情况下,应用程序有自己的工厂对象来创建bean.如果将应用程序自己的工厂对交给Spring管理，那么Spring管理的就不是普通的bean,而是工厂Bean。

● 单例模式：保证全局只有唯一一个对象。

● 适配器模式：SpringAOP的Advice有如下：BeforeAdvice、AfterAdvice、AfterAdvice，而需要这些增强转为aop框架所需的

对应的拦截器MethodBeforeAdviceInterceptor、AfterReturningAdviceInterceptor、ThrowsAdvieInterceptor。

● 代理模式：Spring的Proxy模式在aop中有体现，比如JdkDynamicAopProxy和Cglib2AopProxy。

● 装饰者模式：如HttpServletRequestWrapper，自定义请求包装器包装请求，将字符编码转换的作添加到getParameter()方法中。

● 观察者模式：如启动初始化Spring时的ApplicationListener监听器。

转自我的github

技术讨论群QQ:1398880