动态代理模式

1. 作用：在不改变源代码的基础上对特定类的特定方法实现增强
2. 前提：基于JDK接口的动态代理，被代理对象必须有接口实现，创建代理对象必须传入被代理对象的引用
3. 需求：对服务员类（Water）的service()方法进行增强
4. 实现：

- Water接口：声明服务员类方法

```
public interface Water
{
    /**
     * @Description 服务员服务方法
     */
    public void service();
}
```

- WaterImpl实现类：实现Water接口，重写service()方法

```
public class WaterImpl implements Water
{
    public void service()
    {
        System.out.println("服务中：正在服务......");
    }
}
```

- ProxyUtils工具类：提供Water对象的代理对象获取方法

```
public class ProxyUtils
{
    /**
     * @Description 获取服务员对象的动态代理对象
     * @param water
     * @return 服务员对象的动态代理对象
     */
    public static Water getWaterProxy(Water water)
    {
        ClassLoader loader = water.getClass().getClassLoader();
        Class[] interfaces = water.getClass().getInterfaces();
        InvocationHandler handler = new WaterHandlerImpl(water);
        Water proxy = (Water) Proxy.newProxyInstance(loader, interfaces,
handler);
        return proxy;
    }
}
```

- WaterHandlerImpl实现类：实现InvocationHandler接口，重写invoke()方法，对Water对象的service()方法进行前后增强

```
public class WaterHandlerImpl implements InvocationHandler
{
    private Water water;

    public WaterHandlerImpl(Water water)
    {
        this.water = water;
    }

    /**
     * @param proxy 动态代理对象
     * @param method 被代理对象的方法
     * @param args 被代理对象方法的参数
     * @return 被代理对象方法的返回值
     * @throws Throwable
     */
    public Object invoke(Object proxy, Method method, Object[] args) throws
Throwable
    {
        //增强服务员service()方法
        if("service".equals(method.getName()))
        {
            //前置增强
            System.out.println("服务前：微笑.....");

            Object object = method.invoke(water, args);

            //后置增强
            System.out.println("服务后：鞠躬.....");

            //返回增强方法返回值
            return object;
        }

        //对于非service()方法不增强直接执行
        return method.invoke(water, args);
    }
}
```

- ProxyDemo测试类：测试动态代理对Water对象的service()方法的增强

```
public class ProxyDemo
{
    public static void main(String[] args)
    {
        Water water = new WaterImpl();

        //普通服务
        System.out.println("---------普通服务--------");
        water.service();

        //代理服务
        System.out.println("---------代理服务--------");
        Water waterProxy = ProxyUtils.getWaterProxy(water);
        waterProxy.service();
    }
}
```

- 运行结果

```
D:\jdk\jdk1.8.0_144\bin\java.exe ...
---------普通服务--------
服务中：正在服务......
---------代理服务--------
服务前：微笑......
服务中：正在服务......
服务后：鞠躬......

Process finished with exit code 0
```