1. 将查询函数和修改函数分离

// 重构前

class Person

{

    int age;

    int SetAge(int newAge){

        age = newAge;

        return age;

    }

}

// 重构后

class Person

{

    int age;

    void SetAge(int newAge){

        age = newAge;

}

    void GetAge(){

        return age;

    }

}

1. 如果几个函数执行相似，那么提取公共方法

Function1();

Function2();

替换为

Function1();

Function2();

Common();

1. 如果函数因传入值不同而执行不同的片段，则分离函数

Function(type){

If(type == 1){ .... }

If(type == 2){ .... }

}

替换为

Function1(){

}

Function2(){

}

1. 保持对象完整

如果函数需要依赖于某个对象，那就传入整个对象，而不是传入对象的某些参数

1. 如果函数可以直接获取到参数的值，那就不需要传入

// 重构前

class Person

{

    int age;

    void 方法1(){

        方法2(age);

}

    void 方法2(int age){

        ...

    }

}

// 重构后

class Person

{

int age;

    void 方法1(){

        方法2();

}

    void 方法2(){

        var curAge = age;

        ...

    }

}

1. 引入参数对象

如果参数总是一起出现，那么可以为这些参数新起一个对象

1. 移除设值函数

如果某个属性在对象创建后不再改变，那么应该设为 private set;

1. 以工厂方法取代构造函数

// 重构前

class Person

{

    public Person(int state);

}

// 重构后

class Person

{

protected Person(int state);

public static Person CreateChildren() => new Person(0);

    public static Person CreateAdult() => new Person(1);

}

A，这里有些同学就会问了，这里Person使用了状态码，是不是应该使用类替换掉，是的

1. 以异常取代错误码

如果我们的业务代码无法往下执行，那么我们应该抛出异常

我们的业务逻辑层或者领域层不应该关联错误码

对于webapi，可以根据异常类型返回对应的错误码

1. 以判断取代异常