



Kotlin

中缀表达式

- infix

中缀表达式

- 自定义操作符
- **val** pair = "张三" *to* 30

中缀表达式使用条件

- 必须是成员函数或扩展函数
- 必须只有一个参数
- 参数不能是可变参数或者默认参数

类委托

- 房产中介
- 劳务中介

小头爸爸,围裙妈妈,大头儿子的故事

- 围裙妈妈 只做饭,不洗碗
- 小头爸爸 洗碗 一次奖励10块钱
- 大头儿子 洗碗 一次奖励1块钱



类委托

- 第一种类委托

```
//洗碗能力
interface WashPower{
    //洗碗行为
    fun wash()
}

//大头儿子
class BigHeadSon(var name:String, var age:Int):WashPower{
    override fun wash() {
        println("大头儿子开始洗碗了")
    }
}

//小头爸爸 将洗碗的能力委托给大头儿子实现 通过by
//类委托第一种实现方式
class SmallHeadFather(name: String, age: Int):WashPower by BigHeadSon(name, age)
```

类委托

- 第二种类委托

```
//洗碗能力
interface WashPower{
    //洗碗行为
    fun wash()
}

//大头儿子
class BigHeadSon(var name:String, var age:Int):WashPower{
    override fun wash() {
        println("大头儿子开始洗碗了")
    }
}

//小头爸爸 将洗碗的能力委托给大头儿子实现 通过by
//第二种类委托方式 传递具有洗碗能力的对象 将洗碗行为委托给具有洗碗能力的对象
class SmallHeadFather(name: String, age: Int, washPowre:WashPower):WashPower by washPowre
```


Java代理

- 静态代理
- 动态代理

动态代理

- 定义类实现InvocationHandler接口,复写invoke方法
- 在invoke中调用要代理对象的方法
- 通过Proxy.newProxyInstance 创建代理实例
- 通过代理实例调用方法

属性委托

- 将属性的get和set方法委托给其他对象

天天过春节该有多好啊！



压岁钱

图/王银祥 新华社发



属性委托

- 属性委托就是将字段的get和set方法委托给其他类

```
class BigSon {  
    var 压岁钱: Int by SmallFather() //将压岁钱委托给了爸爸  
}  
  
class SmallFather()  
{  
    var 儿子压岁钱: Int = 0  
    operator fun getValue(bigSon: BigSon,  
                           property: KProperty<*>): Int {  
        return 儿子压岁钱  
    }  
  
    operator fun setValue(bigSon: BigSon,  
                           property: KProperty<*>, value: Int) {  
        儿子压岁钱 = value  
    }  
}
```

by lazy惰性加载

- 懒加载
- 不着急,等我用的时候再给我
- 等我确认完之后再和你联系



by lazy惰性加载

- 对属性字段使用by lazy
`val lazyValue by lazy {"hello"}`
- 只会初始化一次
- by lazy最后一行为返回值
- by lazy是线程安全的

lateinit延迟加载

```
class Person {  
    val name:String by lazy { "张三" }  
    lateinit var email:String  
}
```

- 只能修饰类成员
- 修改var可变变量
- 不能使用在基本数据类型

扩展函数

- 不改变已有类的情况下，为类添加新的函数

扩展函数

- 扩展函数:为现有的类添加新的函数

```
//判断字符串是否为空  
fun String.myIsEmpty():Boolean{  
    return this==null || this.length==0  
}
```

扩展函数

- 可以给任意类添加扩展函数
- 扩展函数可以像成员函数一样访问对象
- 父类定义的扩展函数子类也可以使用

扩展函数是静态解析的

```
val dog1:Animal = Dog()
val dog2:Dog = Dog()

dog1.call() //动物叫
dog2.call() //狗叫
}

//动物扩展出叫声的方法
fun Animal.call() {
    println("动物叫")
}

//给狗扩展出叫声的方法
fun Dog.call() {
    println("狗叫了")
}
```

扩展函数和扩展函数

- 成员函数和扩展函数一样时,执行哪个函数?

扩展函数总结

- 在不改变已有类的情况下,为类添加新的函数
- 扩展函数一般用来代替java的util
- 扩展函数是静态解析的,区别于多态
- 成员函数优先于扩展函数

单例

- 设计模式
- 内存中只有一个对象实例
- 提供了一种创建对象的最佳方式

- 一台电脑只有一个cpu
- 一个国家只有一个领导人
- ...

- 懒汉式单例
- 饿汉式单例

Object-单例

- 单例:在内存中只有一个实例

```
//NetUtil单例模式
object NetUtil{
    var path = "www.baidu.com"
    var method = "get"
    fun sendRequest() {
        println("正在发送网络请求")
    }
}
```

Object总结

- 可以实现单例模式
- Object里面每一个字段都是static静态的
- 适合字段不多的单例

伴生对象companion object

```
class Data{  
    var name = "张三"  
    fun sayHello() {  
        println("hello")  
    }  
    //定义伴生对象  
    companion object { //伴生对象可以指定类名 也可以不指定  
        var age = 20  
        fun hehe() {  
            println("hehe")  
        }  
    }  
}
```

伴生对象总结

- 可以将定义成static静态的属性放在伴生对象里
- 外部类可以直接调用伴生对象里面的方法和属性

枚举

- 一一列举
- 一周有七天:周一 周二 周三 周四 周五 周六 周日
- 一年有12个月
- 三原色:红 蓝 绿
- ...

枚举

- 枚举就是一一列举

```
enum class Week{  
    星期一,星期二,星期三,星期四,星期五,星期六,星期日,  
}
```

枚举高级用法

```
//三元色
//r  g  b
enum class Color(var r:Int, var g:Int, var b:Int) { //三原色
    RED(r: 255, g: 0, b: 0), BLUE(r: 0, g: 0, b: 255), GREEN(r: 0, g: 255, b: 0);

    override fun toString(): String {
        return "Color(r=$r, g=$g, b=$b)"
    }
}
```

数据类

- 只保存数据,没有任何任何逻辑操作,对应java的bean类

13:29

... 78%

網易

24



头条

娱乐

热点

体育

深圳



《看客》：中俄边境 龙熊之变

• • • •



习近平心系乌镇:6年5次到访

高瞻远瞩, 呕心沥血, 精心为千年古镇设计发展之路。

专题



李克强宴请上合各参会领导人

称会议诠释了上合组织"大家庭"的深厚友谊和良好氛围。

专题



曝北大宿舍楼墙可用手指戳洞

被学生戏称为"一阳指培训基地", 校方: 情况没那么严重。

9122跟贴



新闻



阅读



视听



发现



我

数据类总结

- 数据类相当于提供了bean类的模板
- 数据类可以替代java的bean

密封类

- 超强的枚举

权力的游戏



奈德·斯塔克(Stark)



罗伯·斯塔克
(RobStark)



珊莎·斯塔克
(SansaStark)



艾丽娅·斯塔克
(AryaStark)



布兰登·斯塔克
(BrandonStark)



琼恩·雪诺
(JonSnow)

密封类

- 密封类和枚举差不多,枚举在意数据,密封类更在意类型

```
//父类
sealed class NedStark{
    //罗伯
    class RobStark:NedStark()
    //SansaStark
    class SansaStark:NedStark()
    //AryaStark
    class AryaStark:NedStark()
    //BrandonStark
    class BrandonStark:NedStark()
}

//JonSnow
class JonSnow:NedStark()

class Zhans:NedStark()
```