**groovy(10)-闭包委托策略**

/\*

**闭包的三个重要变量:this,owner,delegate**

**区别在于:this代表闭包定义处最近的对象(不包含闭包),owner代表闭包定义出最近的对象可以是闭包.delegate默认与owner一致.**

**delegate可以修改**

\*/

scriptClouser={

    println("scriptClouser this::"+this)

    println("scriptClouser owner:::"+owner)

    println("scriptClouser delegate::::"+delegate)

**//所以大部分情况下 ,this,owner,delegate 是一样//以上三个输出的结果是一样**

/\*scriptClouser this::variable.Clousershujujiegou@17046283

scriptClouser owner:::variable.Clousershujujiegou@17046283

scriptClouser delegate::::variable.Clousershujujiegou@17046283\*/

}

scriptClouser.call()

**//文件中定义一个内部类**

class Preson{

**//类中的闭包**

    def static classClouse={

    println("bscriptClouser this::"+this)

    //bscriptClouser this::classvariable.Preson

    println("bscriptClouser owner:::"+owner)

    //bscriptClouser owner:::class     variable.Preson

    println("bscriptClouser delegate::::"+delegate)

    //bscriptClouser     delegate::::class variable.Preson

}

**//类中的方法的闭包**

    def static fun(){

            def  classClouse={

                    println("funscriptClouser this::"+this)

                    //funscriptClouser this::class variable.Preson

                    println("funscriptClouser owner:::"+owner)

                    //funscriptClouser owner:::class variable.Preson

                     println("funscriptClouser delegate::::"+delegate)

                    //funscriptClouser delegate::::class variable.Preson

               }

              classClouse.call()}

}

Preson.classClouse.call()

Preson.fun()

**//输出的都是 类的 字节码文件对象.class variable.Preson.因为是静态的方法**

class Preson2{

**//类中的闭包**

    def  classClouse={

            println("p2bscriptClouser this::"+this)

            //p2bscriptClouser this::variable.Preson2@1dde4cb2

            println("p2bscriptClouser owner:::"+owner)

            //p2bscriptClouser owner:::variable.Preson2@1dde4cb2

            println("p2bscriptClouser delegate::::"+delegate)

            //p2bscriptClouser delegate::::variable.Preson2@1dde4cb2

    }

            //类中的方法的闭包

       def fun(){

             def  classClouse={

                        println("p2funscriptClouser this::"+this)

                        //p2funscriptClouser this::variable.Preson2@1dde4cb2

                        println("p2funscriptClouser owner:::"+owner)

                        //p2funscriptClouser owner:::variable.Preson2@1dde4cb2

                        println("p2funscriptClouser delegate::::"+delegate)

                        //p2funscriptClouser delegate::::variable.Preson2@1dde4cb2

               }

            classClouse.call()}

        }

}

Preson2 p2=new Preson2();

p2.classClouse.call()

p2.fun()

**//总结1:对象中包含闭包:this,owner,delegate指向的都是离他最近的一个类或对象**

//scriptClouser中的this,owner,delegate ,输出结果是variable.Clousershujujiegou@17046283(即文件对应的对象)

//对应的静态方法中Person中输出的是class variable.Preson ,Person2中输出的是variable.Preson2@1dde4cb2(即文件中包含闭包的类对应的对象)

//闭包中定义一个闭包

def nestClouser={

        def innerClouser={

                println("innerfunscriptClouser this::"+this)

                //innerfunscriptClouser this::variable.Clousershujujiegou@9225652

                println("innerfunscriptClouser owner:::"+owner)

     //innerfunscriptClouser owner:::variable.Clousershujujiegou$\_run\_closure2@72057ecf

                println("innerfunscriptClouser delegate::::"+delegate)

    //innerfunscriptClouser  delegate::::variable.Clousershujujiegou$\_run\_closure2@72057ecf

            }

        innerClouser.delegate=p2

        //delegate修改后:innerfunscriptClouser     delegate::::variable.Preson2@6eda5c9

        innerClouser.call()

}

nestClouser.call()

**//总结2:闭包中包含闭包:this指向他所定义处的文件对象,owner指向离他最近的闭包nestCloust的一个对象,二delegate默认和owner一致.**

/\*

**闭包的委托策略**

Closure.OWNER\_FIRST

Closure.OWNER\_ONLY

Closure.DELEGATE\_FIRST

Closure.DELEGATE\_ONLY

\*/

class Student{

    String name

     def pretty={"my name is ${name}"}

     StringtoString(){    pretty.call()}

}

class Teacher{

    String name

}

def stu=new Student(name:'lich')

def teacher=new Teacher(name:'lilei')

println(stu.toString())//my name is lich.

**//stu的name是通过构造方法传递的,我们通过更改其委托,让他指向teacher对象的name**

stu.pretty.delegate=teacher

**//将stu中的pretty闭包的delegate指向为teacher对象,但是因为其默认的委托策略还是Closure.OWNER\_FIRST(优先从owner中找),所以要设置为Closure.DELEGATE\_FIRST(让他优先从delegate中找)**

stu.pretty.resolveStrategy=Closure.DELEGATE\_FIRST

println(stu.toString())//my name is lilei

**//当我们把Teacher中name去掉后,delegate中找不到时,他又会从owner中去找,所以讲teacher中的name去掉后,结果是:my name is lich.当把stu的委托策略改成Closure.DELEGATE\_ONLY时,会报错,因为此时找不到name**