Jordan Loh consolidated-locks are now objects 9c40e75 on 24 Jul

0 contributors

136 lines (127 sloc) 5.97 KB Raw Blame History

```
1   //The purpose of this snippet is to demonstrate below the use an attenuating object,
2   //caretaker, to mediate access in an OCap system.
3   use collections = "collections"
4   actor Main
5       let env: Env
6       new create(env':Env)=>
7           env = env'
8           env.out.print("---Initial Conditions---")
9           let alice: Node ref = Node.create(env,"alice")
10          let bob: Node ref = Node.create(env,"bob")
11          let carol: Node ref = Node.create(env,"carol")
12          let diane: Node ref = Node.create(env,"diane")
13          try
14          //initial conditions
15          alice.recCap("bob",bob)
16          alice.recCap("carol",carol)
17          carol.recCap("diane",diane)
18          diane.sendProp("diane_prop1","true","diane")
19          env.out.print("---Initial Conditions Completed---")
20
21          //Alice passing a caretaker for Carol, to Bob for Bob's use
22          alice.createCareT("carol-CT","carol") //carol-CT caretaker created
23          alice.sendCap("carol-CT","bob") //alice sends carol-CT to bob
24          //Bob sending his own capability to Carol
25          bob.sendCap("bob","carol-CT")
26          //Carol sending Diane's capability to Bob
27          carol.sendCap("diane","bob")
28          //Bob tells sets prop1 in Carol to be true
29          bob.sendProp("carol_prop1","true","carol-CT") //bob sends property (prop1 = true) to carol-CT
30          env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true
31
32          //POST-LOCK
33
34          //Alice changes lock of Carol-CT
35          alice.changelock(true,"carol-CT-lock") //alice locks carol-CT
36          //Bob tries to change prop1=false on carol-CT and the lock should prevent him from doing so
37          bob.sendProp("carol_prop1","false","carol-CT") //bob tries to change prop1 = false to carol-CT
38          env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true
39          //Bob tries to change prop1=false on diane and will succeed because nothing is preventing him from doing so
40          bob.sendProp("diane_prop1","false","diane") //bob tries to change prop1 = false to diane
41          env.out.print("MAIN:diane_prop1 is "+diane.getProp("diane_prop1")) //because caretaker is locked, should return true
42
43          end
44
45  class Lock
46      var _state: Bool val
47      new ref create()=>
48          _state = false
49      fun ref unlock()=>
50          _state = false
51      fun ref lock()=>
52          _state = true
53      fun box state():Bool val=>
54          _state
55
56  class Caretaker
```

```
    let _target: (Caretaker ref|Node ref)
    var _lock: Lock ref
    new ref create(target':(Caretaker ref|Node ref), lock':Lock ref)=>
        _target = target'
        _lock = lock'
    fun box _locked():Bool val=>
        _lock.state()
    fun box getProp(id:String val):String val?=>
        try
            if _locked() is false then _target.getProp(id) else error end
        else error end
    fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
        try
            if _locked() is false then _target.sendProp(id,prop,rec) else error end
        else error end
    fun ref recProp(id:String val,prop:String val)=>
        if _locked() is false then _target.recProp(id,prop) end
    fun ref getCap(id:String val): (Node ref|Lock ref|Caretaker ref)?=>
        try
            if _locked() is false then _target.getCap(id) else error end
        else error end
    fun ref sendCap(id:String val, rec:String val)?=>
        try
            if _locked() is false then _target.sendCap(id,rec) end
        else error end
    fun ref recCap(id:String val, cap':(Node ref|Lock ref|Caretaker ref))=>
        if _locked() is false then _target.recCap(id,cap') end
    fun ref delCap(id:String val)?=>
        try
            if _locked() is false then _target.delCap(id) end
        else error end
    fun ref createCareT(id:String val,target:String val):Caretaker ref?=>
        try
            if _locked() is false then _target.createCareT(id,target) else error end
        else error end

class Node
    let env: Env
    let name: String
    let _caps: collections.Map[String val, (Node ref|Lock ref|Caretaker ref)] = _caps.create()
    let _props: collections.Map[String val, String val] = _props.create()

    new ref create(env':Env, name':String)=>
        env = env'; name = name'
        _caps(name)=this
    fun ref changelock(lock:Bool val,rec: String val)?=>
        try if lock is true then (getCap(rec) as Lock ref).lock()
        else (getCap(rec) as Lock ref).unlock() end
        env.out.print(name+": changing lock of "+rec+" to "+lock.string())
        else error end
    fun box getProp(id:String val):String val ?=>
        try _props(id) else error end
    fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
        env.out.print(name+":sending ("+id+" as "+prop+") to "+rec)
        try (getCap(rec) as (Caretaker ref|Node ref)).recProp(id,prop) else error end
    fun ref recProp(id:String val,prop:String val) =>
        env.out.print(name+":"+id+" changed to "+prop)
        _props(id)=prop
    fun ref getCap(id:String val): (Node ref|Lock ref|Caretaker ref)?=>
        try  _caps(id) else error end
    fun ref sendCap(id:String val, rec:String val)?=>
        env.out.print(name+":sending capability of "+id+" to "+rec)
        try (getCap(rec) as (Caretaker ref|Node ref)).recCap(id, getCap(id)) else error end
    fun ref recCap(id:String val, cap':(Node ref|Lock ref|Caretaker ref))=>
        _caps(id) = cap'
        env.out.print(name+":received capability of "+id)
    fun ref delCap(id:String val) ?=>
        try _caps.remove(id) else error end
    fun ref createCareT(id:String val,target':String val):Caretaker ref?=>
        env.out.print(name+":creating caretaker "+id+" for "+target')
        try
            let cap = (getCap(target') as (Caretaker ref|Node ref))
            let lockname: String val = id+"-lock"
            let lock:Lock ref = Lock.create()
```

```
131             let caretaker:Caretaker ref = Caretaker.create(cap,lock)
132             recCap(lockname, lock)
133             recCap(id, caretaker)
134             caretaker
135         else error end
```