

Delegated Properties



Delegated Properties

Allow you to control all aspects of getting or setting a property

```
class DelegatedProp<A>(constrArgs: Int, default: A) {  
    var store: A = default  
    operator fun getValue(currentRef: Any, prop: KProperty<*>): A {  
        // perform side effect whenever the property is accessed  
        // must return a value of the correct type  
        return store  
    }  
  
    operator fun setValue(currentRef: Any, prop: KProperty<*>, value: A) {  
        // perform side effect whenever the property is changed  
    }  
}  
  
class MyUsefulClass {  
    var myProperty: Int by DelegatedProp(constrArgs = 42, default = 0)  
}
```

signatures must be exact



delegated property



get() and set() will call the class' getValue() and setValue()

Standard Delegated Properties

Lazy: delay evaluation of a property until first use

```
val userData: UserData by lazy {  
    // useful to delay long computations or network calls until necessary  
    fetchUserData()  
}
```

Vetoable: accept or deny changes on a condition

```
class BankAccount(initialBalance: Double) {  
    var balance: Double by Delegates.vetoable(initialBalance) { prop, oldValue, newValue ->  
        newValue >= 0 // change is accepted only when condition is true  
    }  
}
```

Standard Delegated Properties

Observable: perform arbitrary side effects on property changes

```
class MonitoredDataset(name: String) {  
    var state: State by Delegates.observable(State.NONE) { prop, oldValue, newValue ->  
        // useful for logging and alerting  
        println("[dataset - $name] State changed: $oldValue -> $newValue")  
        if (newValue == State.STALE)  
            println("[dataset - $name] Alert: dataset is now stale, refresh data")  
    }  
}
```

Map: access map values in a statically typed way

```
class WeakObject(val attributes: Map<String, Any>) {  
    val name: String by attributes // beware this can crash if "name" is not in the map  
    val size: Int by attributes  
}  
  
// useful for bridging the gap between Kotlin and weakly typed structures (e.g. JSON)
```

Kotlin rocks

