

KVC & KVO

Schlüssel, Benachrichtigungen und Katzen.

Key Value Coding

Attribute über einen Schlüssel lesen und setzen.



This sounds
rather boring. I can
already do this using
properties!

Apple Script

KVO

Cocoa Bindings

Key Value Coding

Core Data

keyed archiving

associative references

Key Value Coding

- - valueForKey:
- -setValue:forKey:
- NSString *name;
name = [myCat valueForKey:@"name"];
- [myCat setValue:@"name" forKey:@"Seri"];

Key Paths

- Durch Punkt getrennte Liste von Schlüsseln
- Erlauben Zugriff auf den Objektgraphen

Key Paths

- - (id)valueForKeyPath:(NSString *)keyPath;
- - (void)setValue:(id)value forKeyPath:(NSString *)keyPath;
- NSNumber *age;
age = [self valueForKeyPath:@"cat.father.age"]
- NSNumber *age = [NSNumber numberWithInt:2];
[self setValue:age forKeyPath:@"cat.father.age"];

Skalare Typen und Strukturen

- NSNumber
- NSValue
- setNilValue(forKey:

KVC Funktionsweise

- KVC-konforme Getter/Setter
- Variable wird direkt angesprochen
- setValue:forUndefinedKey: führt zu
NSUndefinedKeyException

KVC-konform

- Getter
 - - <key> (-sleeping)
 - - is<Key> (-isSleeping)
- Setter
 - - set<Key> (setSleeping:)

Demo

Key Value Coding (einfache Typen & Key Paths)

Geordnete Mengen

- – mutableArrayValueForKey:
- – mutableArrayValueForKeyPath:

Geordnete Mengen

- Getter

- - <key>

- Setter

- -insertObject:in<Key>AtIndex:
- -removeObjectFrom<Key>AtIndex:
- Optional:
 - replaceObjectIn<Key>AtIndex:withObject:

Getter Alternativen:

- countOf<Key>
- objectIn:<Key>AtIndex:oder -<key>AtIndexes:
- get<Key>

Setter Alternativen:

- insert<Key>AtIndexes:
- remove<Key>AtIndexes:
- replace<Key>AtIndexes:with<Key>: (optional)

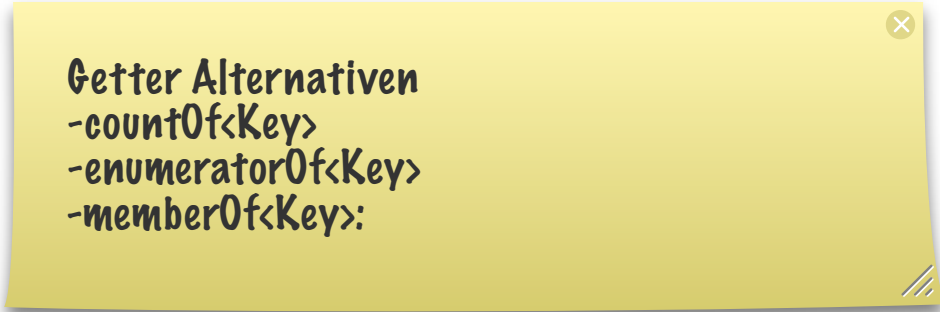
Ungeordnete Mengen

- – mutableSetValueForKey:
- – mutableSetValueForKeyPath:

Ungeordnete Mengen

- Getter

- -<key>




Getter Alternativen
-countOf<Key>
-enumeratorOf<Key>
-memberOf<Key>:

- Setter

- -add<Key>:

- -remove<Key>:

- Optional: -intersect<Key>:



Setter Alternativen
-add<Key>Object:
-remove<Key>Object:
set<Key>:

Mengenoperationen

- `<keyPathMenge>.@operator.<keyPathAttribut>`
- Beispiele
 - `@'cats.@avg.age'`

Mengenoperationen

- Einfache Operationen
 - z.B. @avg, @count
- Objekt Operationen
 - z.B. @unionOfObjects, @distinctUnionOfObjects

Demo

Mengen und Mengenoperationen

Validierung

- -set<Key>: sollte keine Validierung durchführen
- -validate<Key>:error:

Nachteile

- Tippfehler in Keys werden erst zur Laufzeit erkannt
- Keine automatische Vervollständigung
- Mitunter etwas langsam
- Datenkapselung kann umgangen werden

Key Value Observing

Observing value changes.

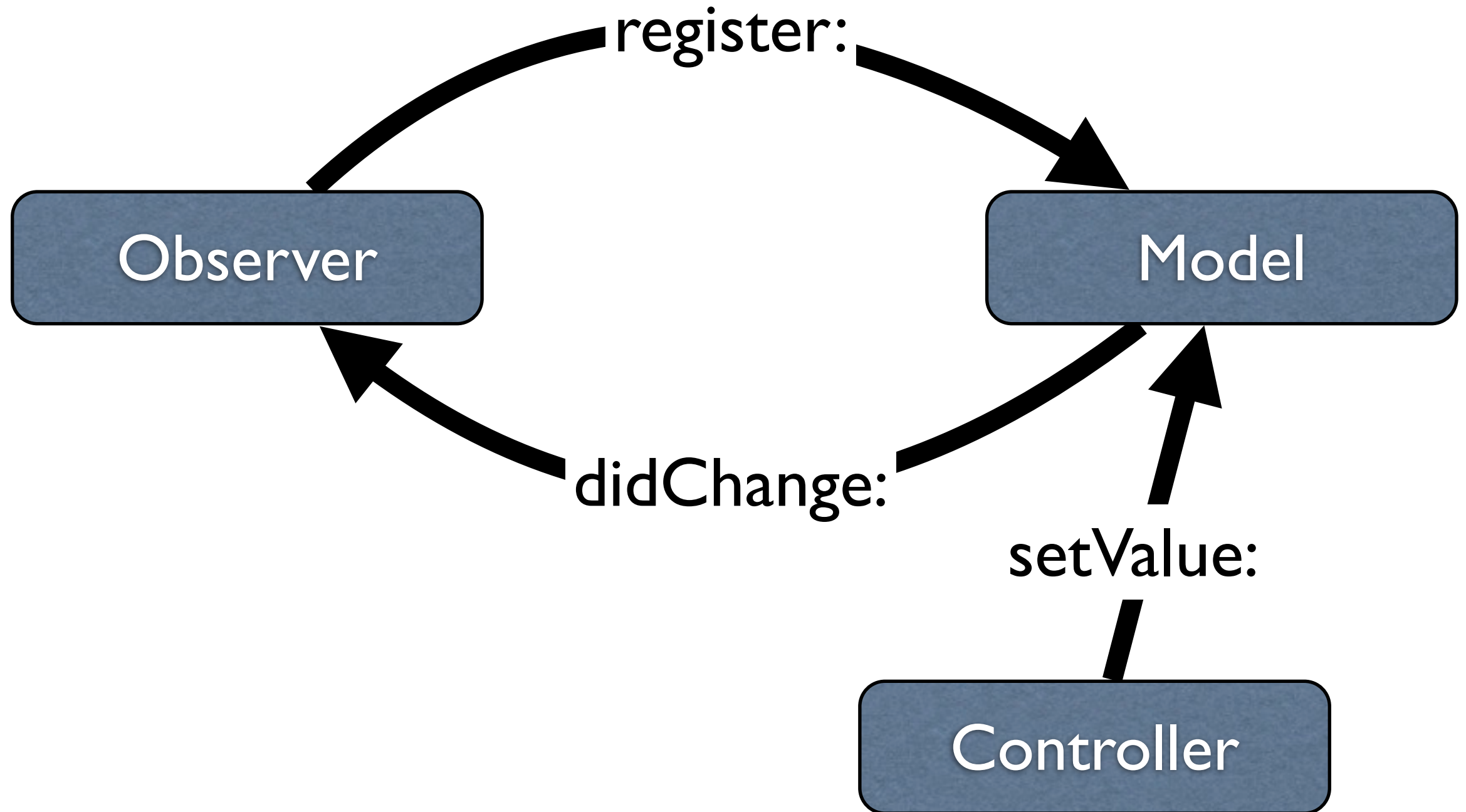


I'm your observer!

Key Value Observing

Mechanismus um bei Veränderungen an einem Attribut eines Objekts benachrichtigt zu werden.

Key Value Observing



Registrieren

- -addObserver:forKeyPath:options:context:
- -removeObserver:forKeyPath:

Verständigung

- -observeValueForKeyPath:ofObject:change:context:

Demo

KVO

Automatisches KVO

- Attribut muss KVC compliant sein

Manuelles KVO

- +automaticallyNotifiesObserversForKey:
- -willChangeValueForKey:
- -didChangeValueForKey:

Abhängige Schlüssel

- Wenn eine Änderung auch andere Attribute beeinflusst.
- `+keyPathsForValuesAffecting<Key>`
- Beispiel: Änderungen an `firstName` bzw. `lastName` beeinflusst `fullName`

Mögliche Probleme

- Notifications werden immer am aktuellen Thread abgearbeitet. Vorsicht bei View Updates.
- Veränderungen an mehreren abhängigen Schlüssel werden nicht zusammengefasst.
- addObserver: ist teuer und sollte z.B. nicht für TableView Cells verwendet werden.
- Objekte in Mengen müssen einzeln observiert werden.

KVO Funktionsweise

- Subklasse wird zur Laufzeit erzeugt
- Setter werden wie beim manuellen KVO implementiert
- isa-pointer der original Klasse wird neu gesetzt (isa-swizzling)

Cocoa Bindings

Ein Ausblick.

Talking about
bindings...



Cocoa Bindings

Automatisches Synchronisieren von View und Model über einen Controller

Demo

Cocoa Bindings

