Making Core Data Your Bitch

Daniel Tull

Categories in Objective-C

* Allow you to add methods (and properties) to any class
* No ivar storage
* Duplicate methods will overwrite (or not, who knows, but they will definitely cause issues), so prefix methods:
  * `-dct_description`
  * `-dctDescription`

Making Core Data Sensible

Creating a Managed Object

Default:

Ask *NSEntityDescription(??)* to create a new managed object with the given entity name and insert it into the given context

```
NSString *const TweetEntityName = @“Tweet”;

Tweet *tweet = [NSEntityDescription
insertNewObjectForEntityForName:TweetEntityName
inManagedObjectContext:context];
```

Creating a Managed Object

Sensible:

> Ask NSManagedObjectContext for an inserted managed object with the given entity name

```
NSString *const TweetEntityName = @“Tweet”;
```

```
Tweet *tweet = [context
dct_insertNewObjectForEntityName:TweetEntityName];
```

Using NSManagedObjectContext (DCTDataFetching)

Making Core Data Less Repetitive

Fetching Managed Objects

Default:

Get an entity description with a given entity name and context

Create a fetch request and set the entity (description)

Use the context to execute the fetch request

Check for errors

```
NSEntityDescription *entity = [NSEntityDescription
entityForName:TweetEntityName
inManagedObjectContext:context];

NSFetchRequest *request = [[NSFetchRequest alloc] init];

[request setEntity:entity];

NSError *error = nil;

NSArray *tweets = [context executeFetchRequest:request
error:&error];

if (error) {/* handle error */}

[request release];
```

Fetching Managed Objects

Less Repetitive:

Get the object(s) from the context using an entity description

```
NSArray *tweets = [context
dct_objectsForEntityName:TweetEntityName];
```

Using NSManagedObjectContext (DCTDataFetching)

Making Core Data Automated

Creating a Managed Object from a Dictionary

```
{
    name = “Daniel Tull”,
    username = “danielctull”,
    tweets = [
        {•••},
        {•••},
        {•••}
     ]
}
```

Creating a Twitter User

Default:

Check if a managed object already exists

If not, get a new inserted managed object

Set each property from the dictionary

for relationships, create those objects and set the relationship

// Really need to check for existence of user first!

```
TwitterUser *user = [context
dct_insertNewObjectForEntityName:TwitterUserEntityName];

user.name = [userDictionary valueForKey:@“name”];

user.username = [userDictionary valueForKey:@“username”];

NSArray *tweets = [userDictionary valueForKey:@“tweets”];

for (NSDictionary *tweetDictionary in tweets) {

   Tweet *tweet = [context
   dct_insertNewObjectForEntityName:TweetEntityName];

   •••

   tweet.user = user;

}
```

Creating MOs from an Array of Dictionaries

Automated:

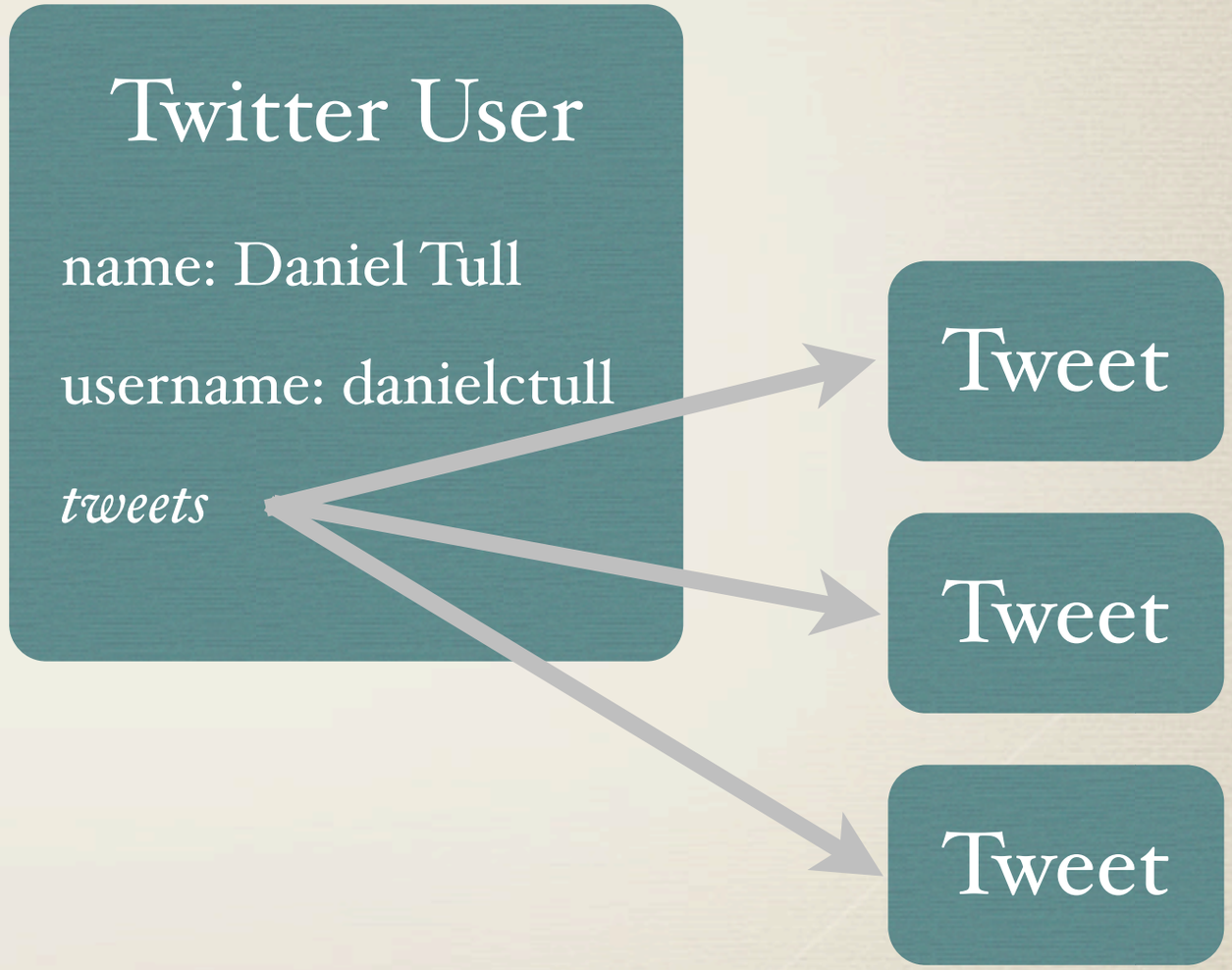
Ask the object class for a managed object set up with the dictionary and inserted into a context

```
TwitterUser *user = [TwitterUser
dct_objectForDictionary:userDictionary
managedObjectContext:context];
```

Using NSManagedObject (DCTAutomatedSetup)

How it works

```
{
    name = “Daniel Tull”,
    username = “danielctull”,
    tweets = [
        {•••},
        {•••},
        {•••}
     ]
}
```



Using DCTAutomatedSetup

Any managed object must conform to the `DCTManagedObjectAutomatedSetup` protocol

If it doesn't, it won't get setup

Property name conversion

Dictionary key names can differ from the property names of the model object

Solution is to map the remote names to the model object names

```
+dct_mappingFromRemoteNamesToLocalNames
```

```
+ (NSDictionary *)dct_mappingFromRemoteNamesToLocalNames {

   return [NSDictionary dictionaryWithObject:@“userID”
   forKey:@“id”];

}

// “id” => “userID”
```

Property name conversion

```
{
    id = 12345,
    name = “Daniel Tull”
    username = “danielctull”
}
```

Twitter User

userID: 12345

name: Daniel Tull

username: danielctull

Converting Value Types

Most values will be strings or numbers, so will need to convert the type of some of the values.

```
-dct_handleKey:value:

+dct_convertValue:toCorrectTypeForKey:
```

```
- (BOOL)dct_handleKey:(NSString *)key value:(id)value {
   if ([key isEqualToString:@“date”]) {
      self.date = [NSDate date];
      return YES;
   }
   return NO;
}
```

```
+ (id)dct_convertValue:(id)value toCorrectTypeForKey:
(NSString *)key {

   if ([key isEqualToString:@“date”]) {

      NSDate *date = [NSDate dateFromTwitterString:value];

      return date;

   }

   return value;

}
```

Converting Value Types

```
{
    id = 12345,

    text = “OMG Awesome
    presentation by
    @danielctull!!”

    date = “2011 05 1 7:45pm”
}
```

Tweet

tweetID: 12345

text: OMG Awesome presentation by @danielctull!!

date: <NSDate 19:45 5/1/2011>

Existing Objects

Need to check for an existing managed object to use

The following methods give information to the setup process to fetch existing objects:

```
+dct_uniqueKey

-dct_uniqueKeys
```

```
+ (NSString *)dct_uniqueKey {
    return @“tweetID”;
}
```

```
- (NSArray *)dct_uniqueKeys {

    return [NSArray arrayWithObjects:@“userID”, @“username”,
    nil];

}
```

Existing Objects

```
{
    id = 12345,
    name = “Daniel Tull”
    username = “danielctull”
}
```

Twitter User

userID: 12344

username: mikeabdullah

Twitter User

userID: 12345

username: danielctull

name: Daniel Tull

Twitter User

userID: 12344

username: dannygreg

Daniel Tull

@danielctull

github.com/danielctull/DCTCoreData

danieltull.co.uk/DCTCoreData/Documentation