Making Core Data Your Bitch

Daniel Tull

DCTCoreData

Simpler data fetching

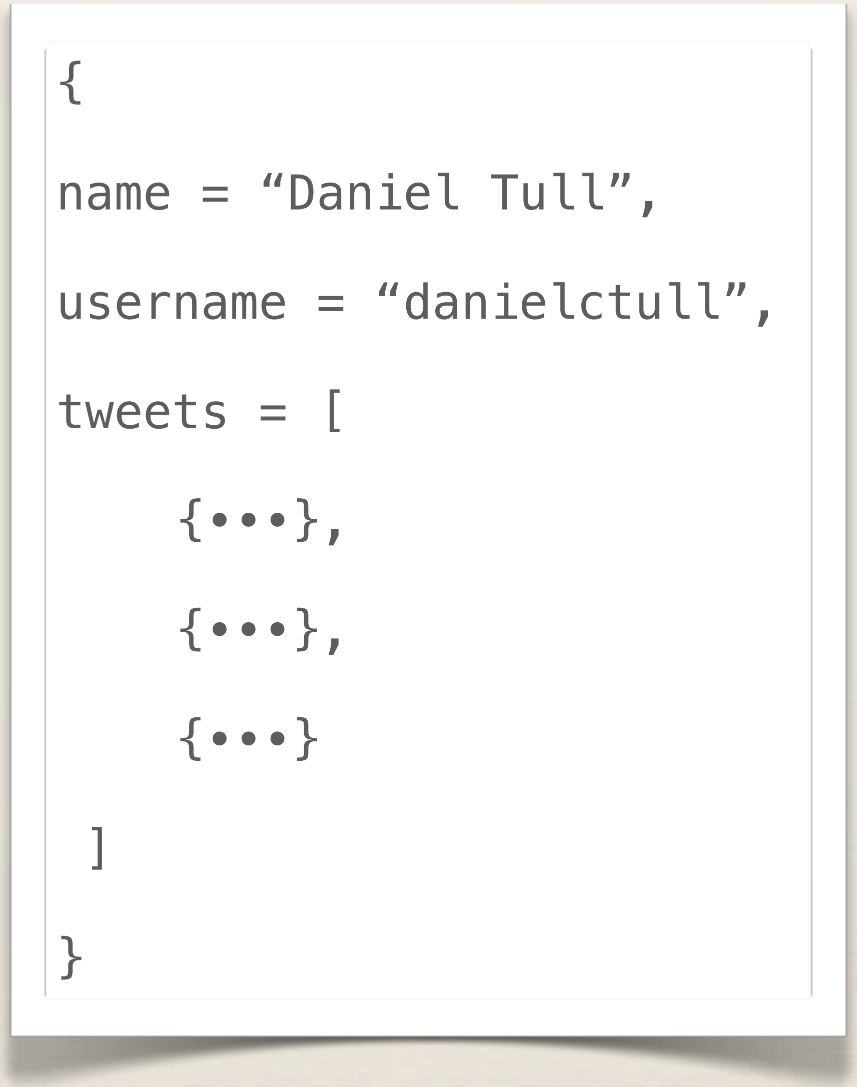
Asynchronous data fetching and tasks

Ordered relationships

Automated setup from dictionaries

Making Core Data Automated

Getting a Managed Object for a Dictionary



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

Getting a Managed Object for a Dictionary

Twitter User

name: Daniel Tull

usename: danielctull

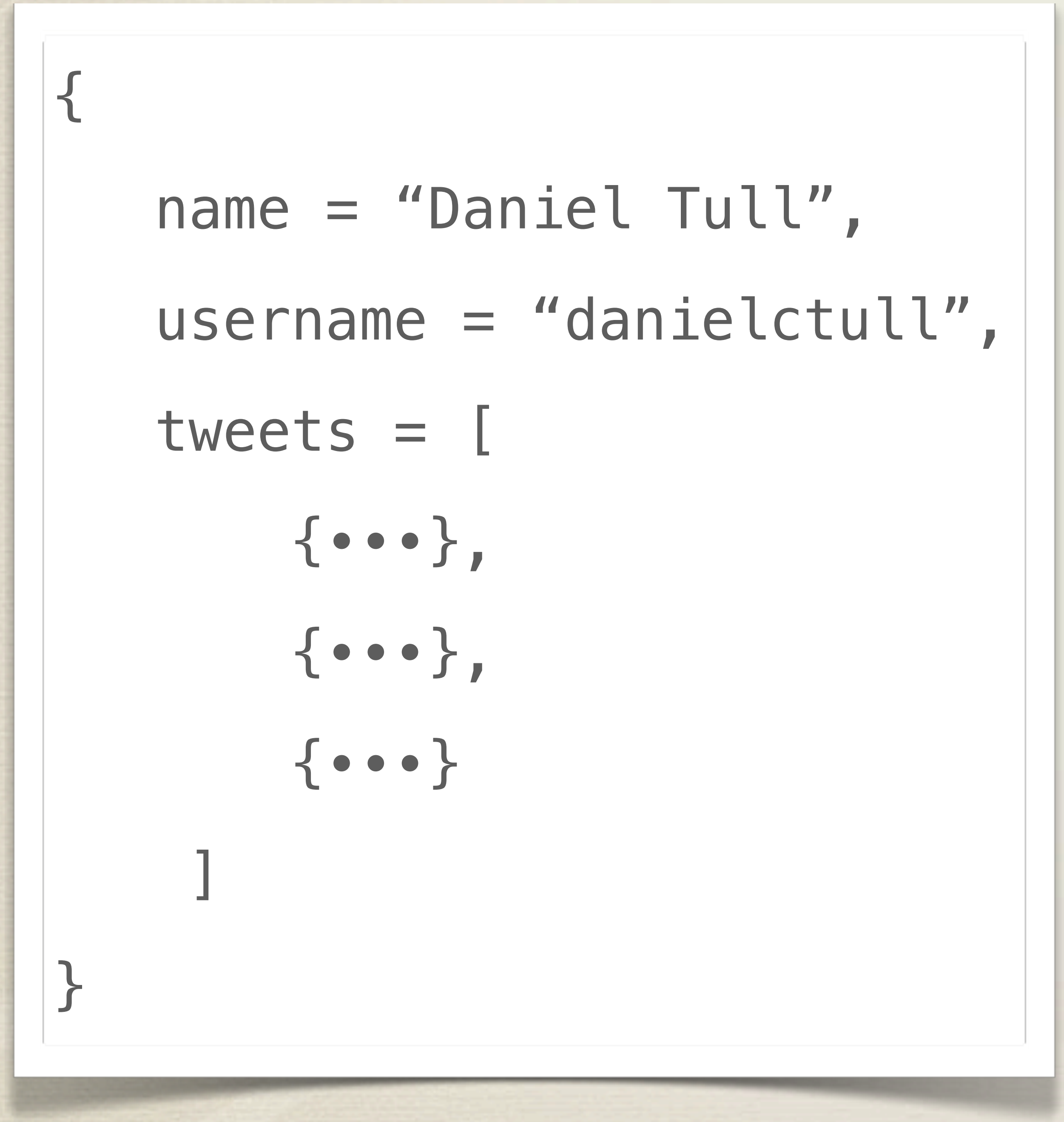
tweets

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

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

Using NSManagedObject (DCTAutomatedSetup)

How it works



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

How it works

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

Twitter User

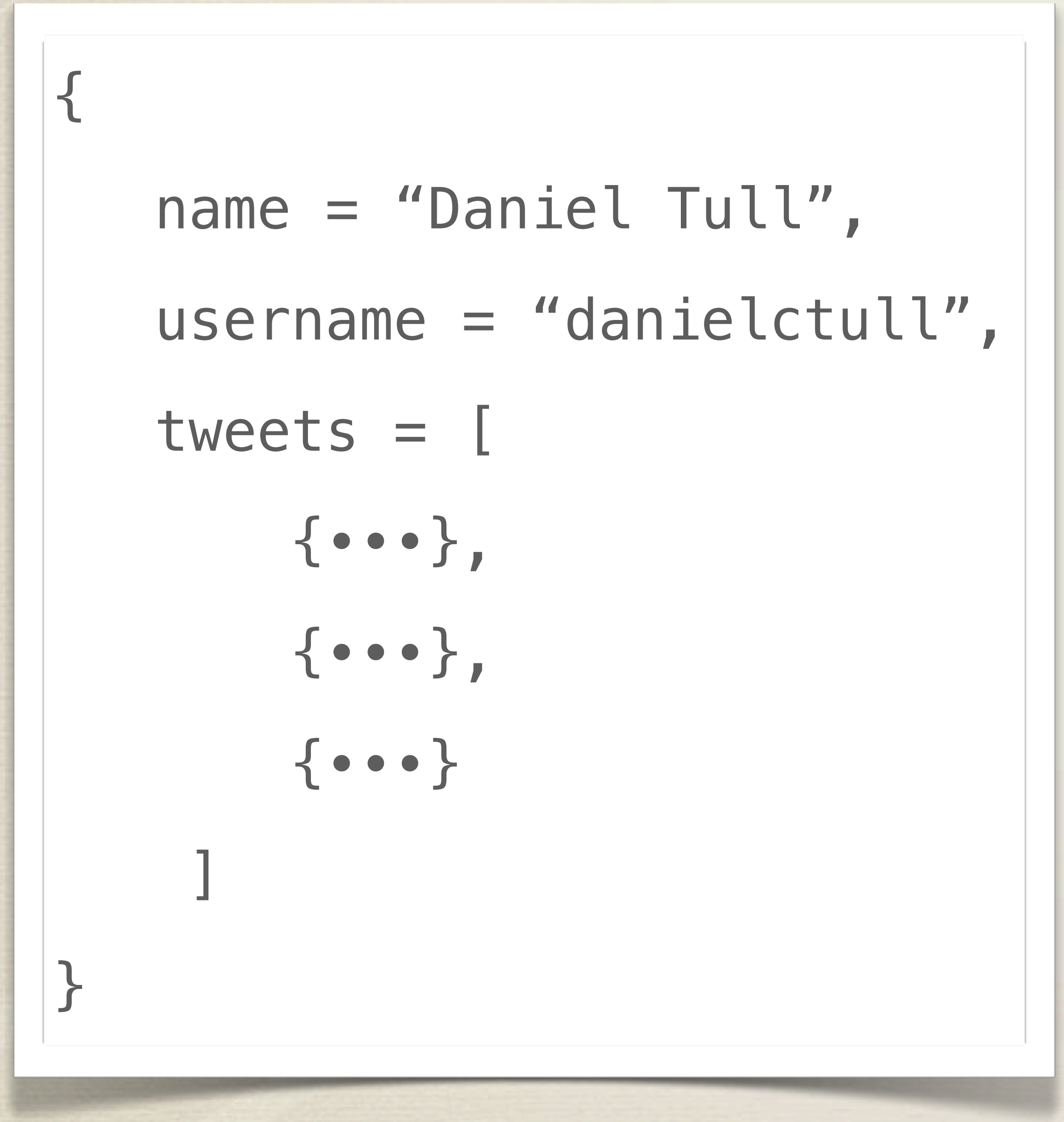
How it works

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

Twitter User

name: Daniel Tull

How it works



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

Twitter User

name: Daniel Tull

username: danielctull

How it works

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

Twitter User

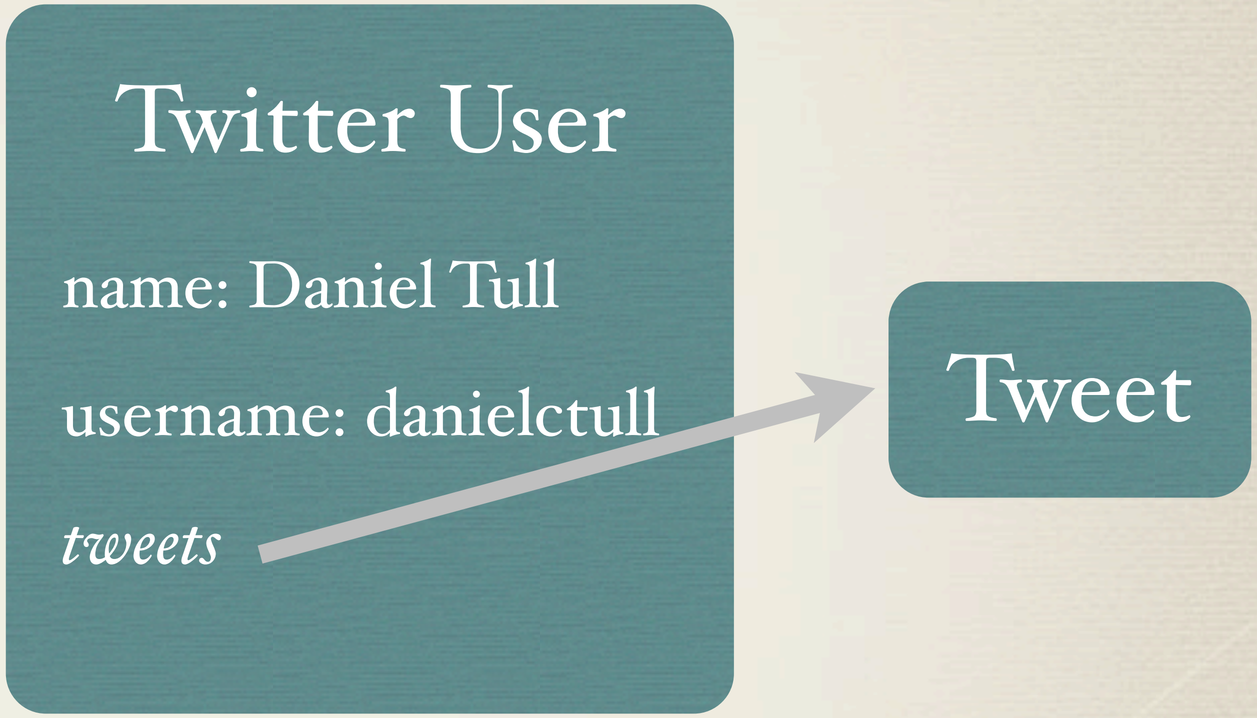
name: Daniel Tull

username: danielctull

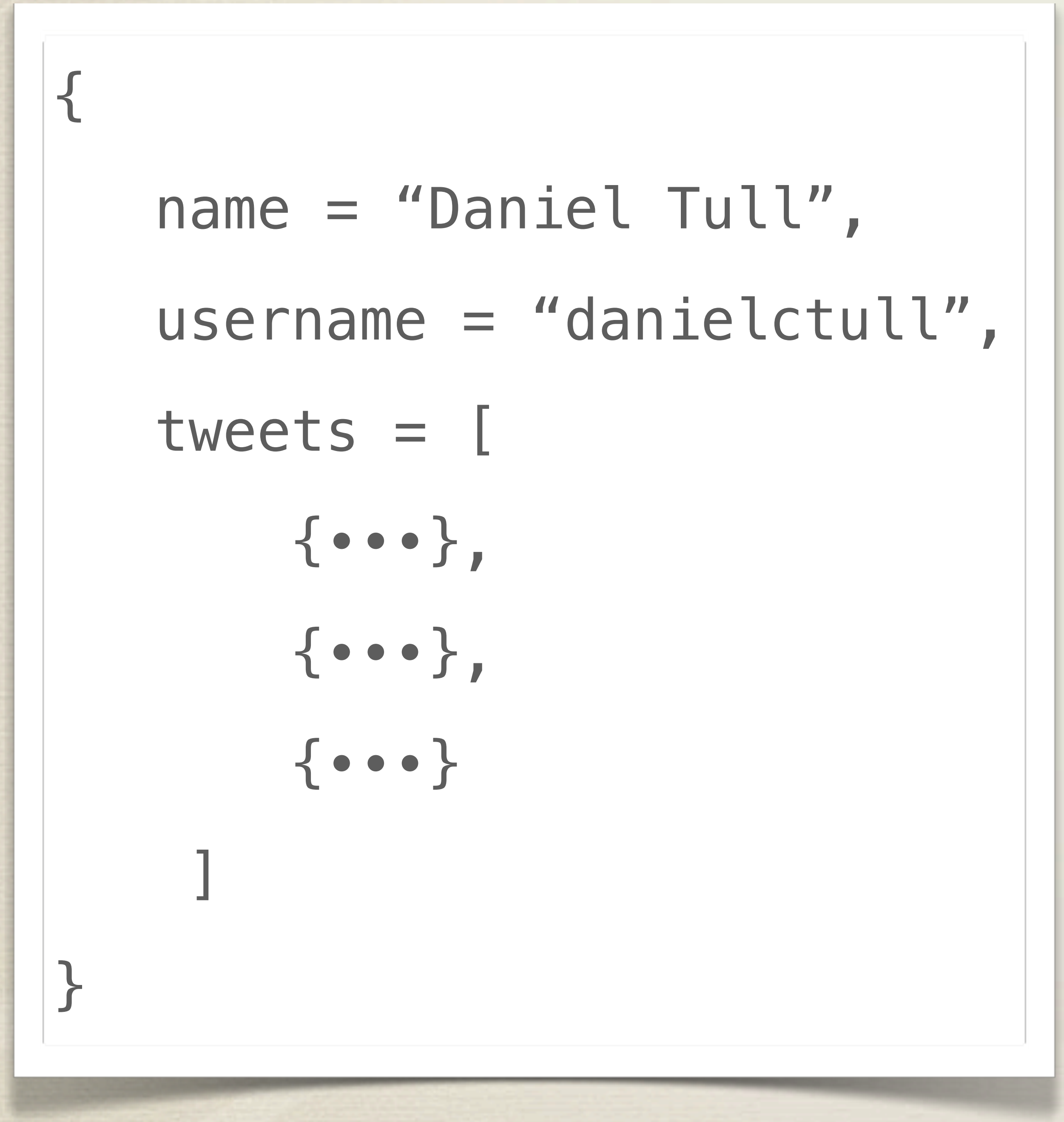
tweets

How it works

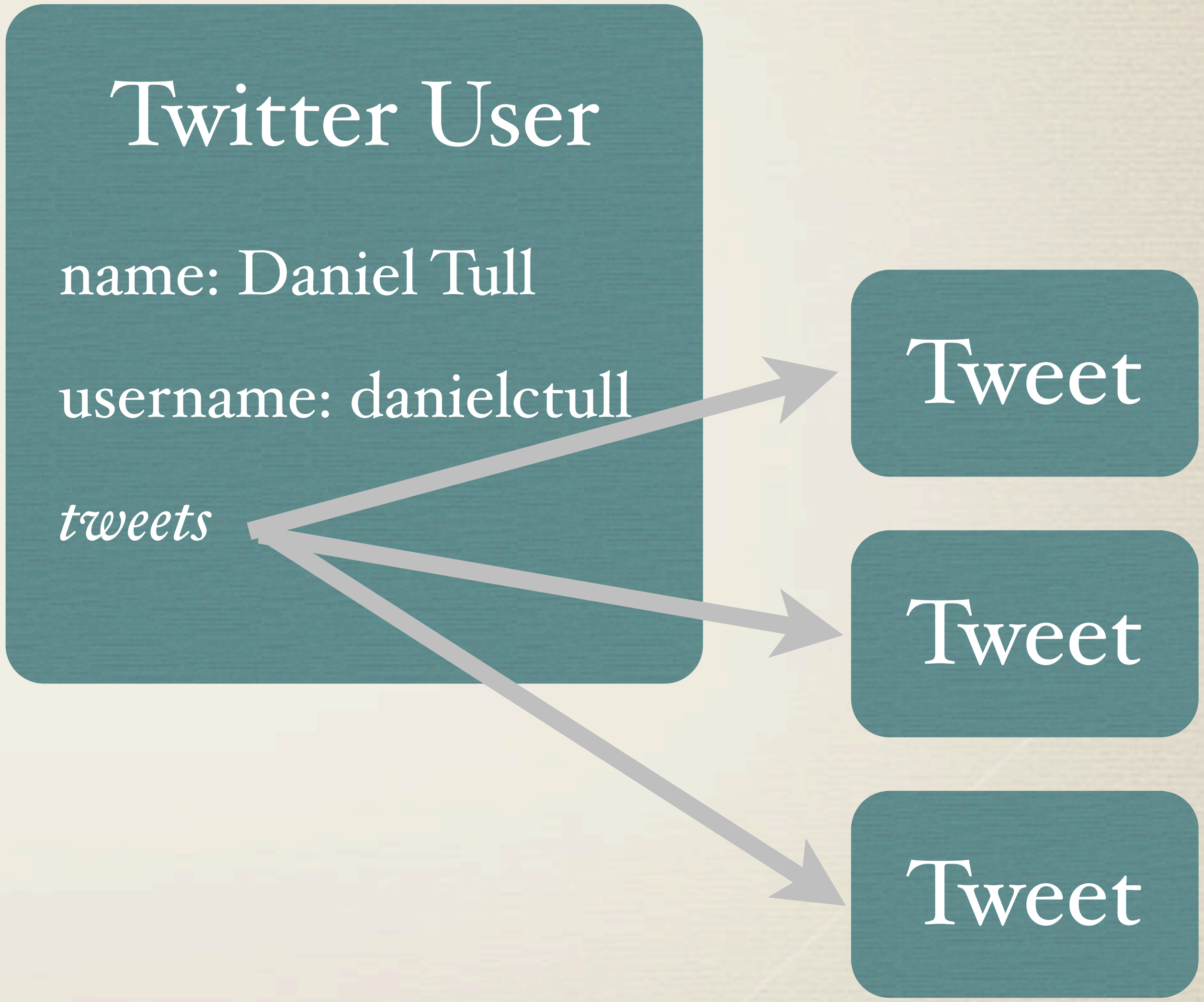
```
{
   name = “Daniel Tull”,
   username = “danielctull”,
   tweets = [
       {•••},
       {•••},
       {•••}
    ]
}
```



How it works



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



Twitter User

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

Allows a simple way to opt in to the process

Property name conversion

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

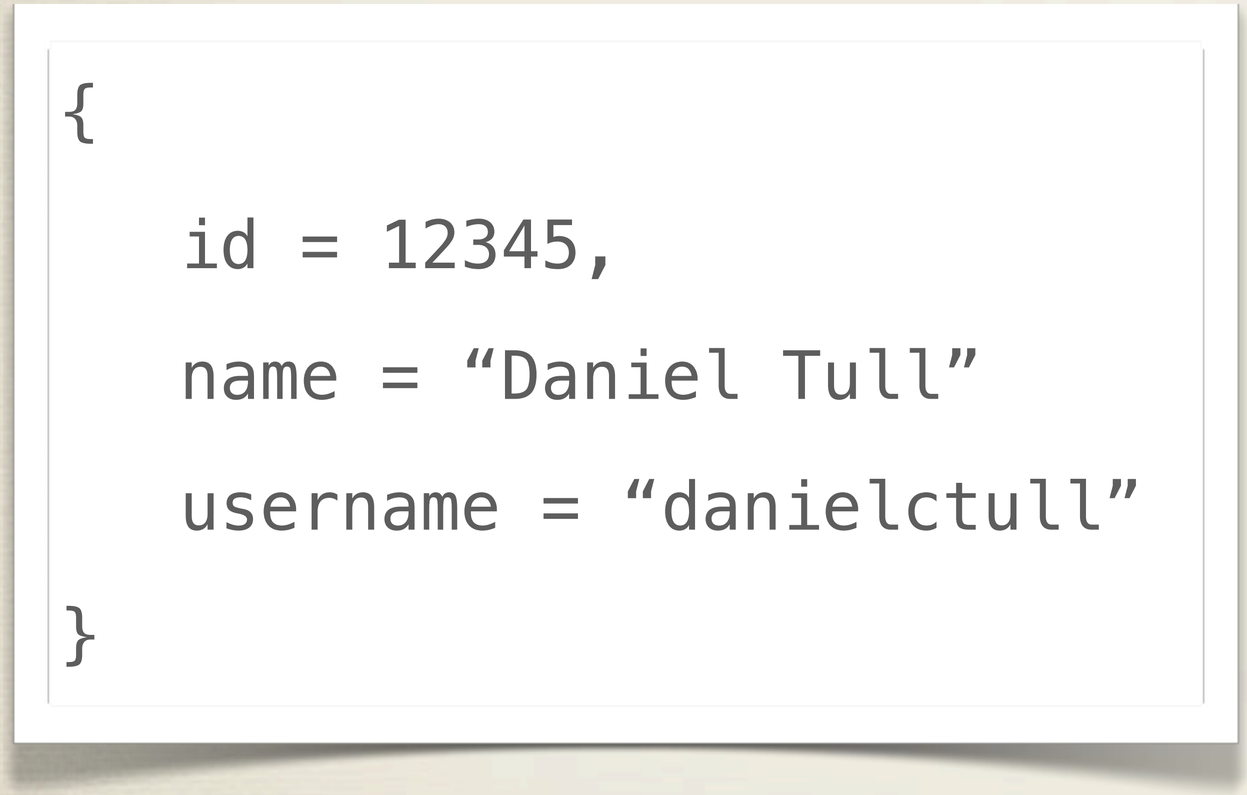
Solution is to map the remote names to the model object names

```
+dct_mappingFromRemoteNamesToLocalNames
```

```
+ (NSDictionary *)dct_mappingFromRemoteNamesToLocalNames {

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

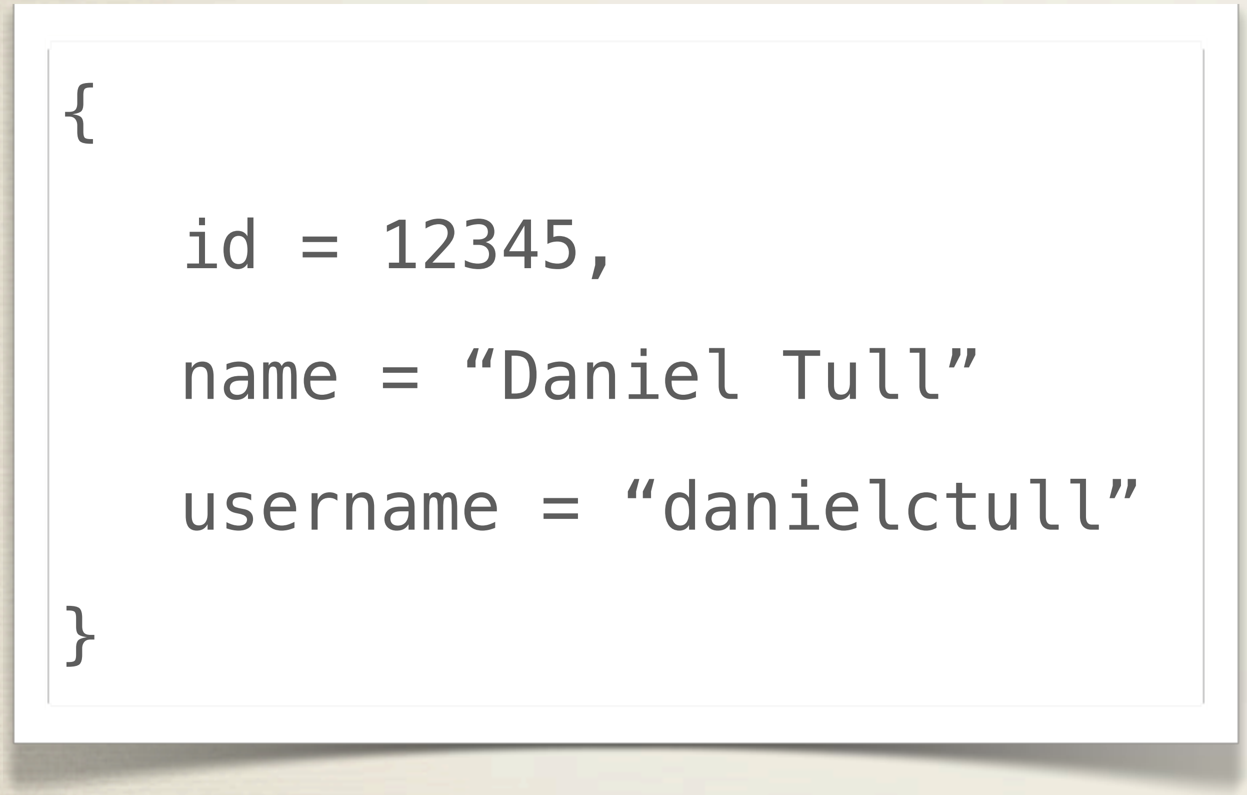
// “id” => “userID”

Property name conversion



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

Property name conversion



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

Twitter User

userID: 12345

name: Daniel Tull

username: danielctull

Converting Value Types

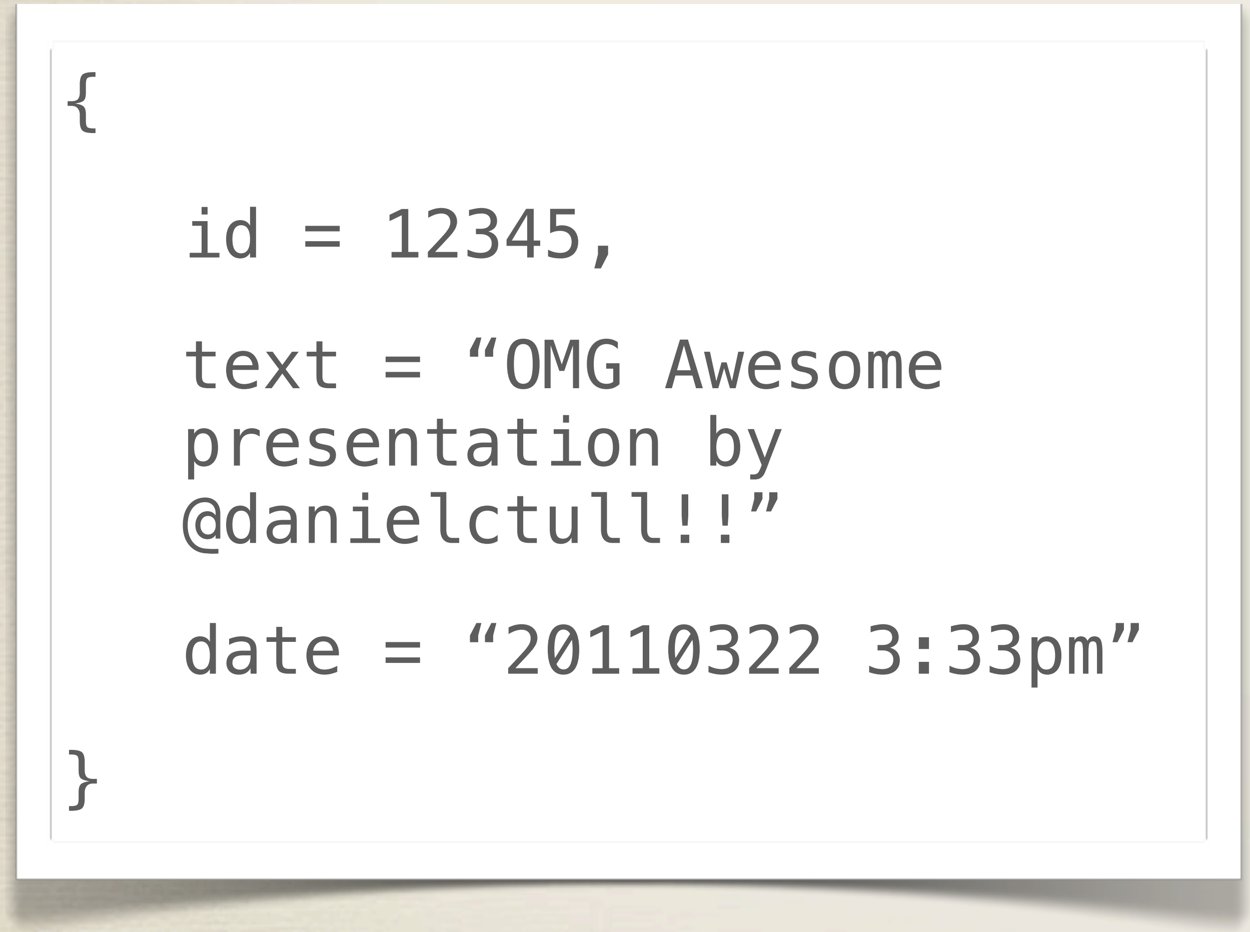
Most values will be strings or numbers, so will need to convert the type of some of the values.

```
-dct_setSerializedValue:forKey:

+dct_convertValue:toCorrectTypeForKey:
```

```
- (BOOL)dct_setSerializedValue:(id)value
                        forKey:(NSString *)key {

  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 = “20110322 3:33pm”
}
```

Converting Value Types

```
{

   id = 12345,

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

   date = “20110322 3:33pm”

}
```

Tweet

tweetID: 12345

text: OMG Awesome presentation by @danielctull!!

date: <NSDate 15:33 22/03/2011>

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_uniqueKeys
```

```
+ (NSArray *)dct_uniqueKeys {

  return [NSArray arrayWithObject:@“tweetID”];
}
```

```
+ (NSArray *)dct_uniqueKeys {

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

Unique Key Conversions

To work effectively, use:

```
+dct_convertValue:toCorrectTypeForKey:
```

Existing Objects

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

Existing Objects

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

Twitter User

userID: 12344

username: mikeabdullah

Twitter User

userID: 12345

username: danielctull

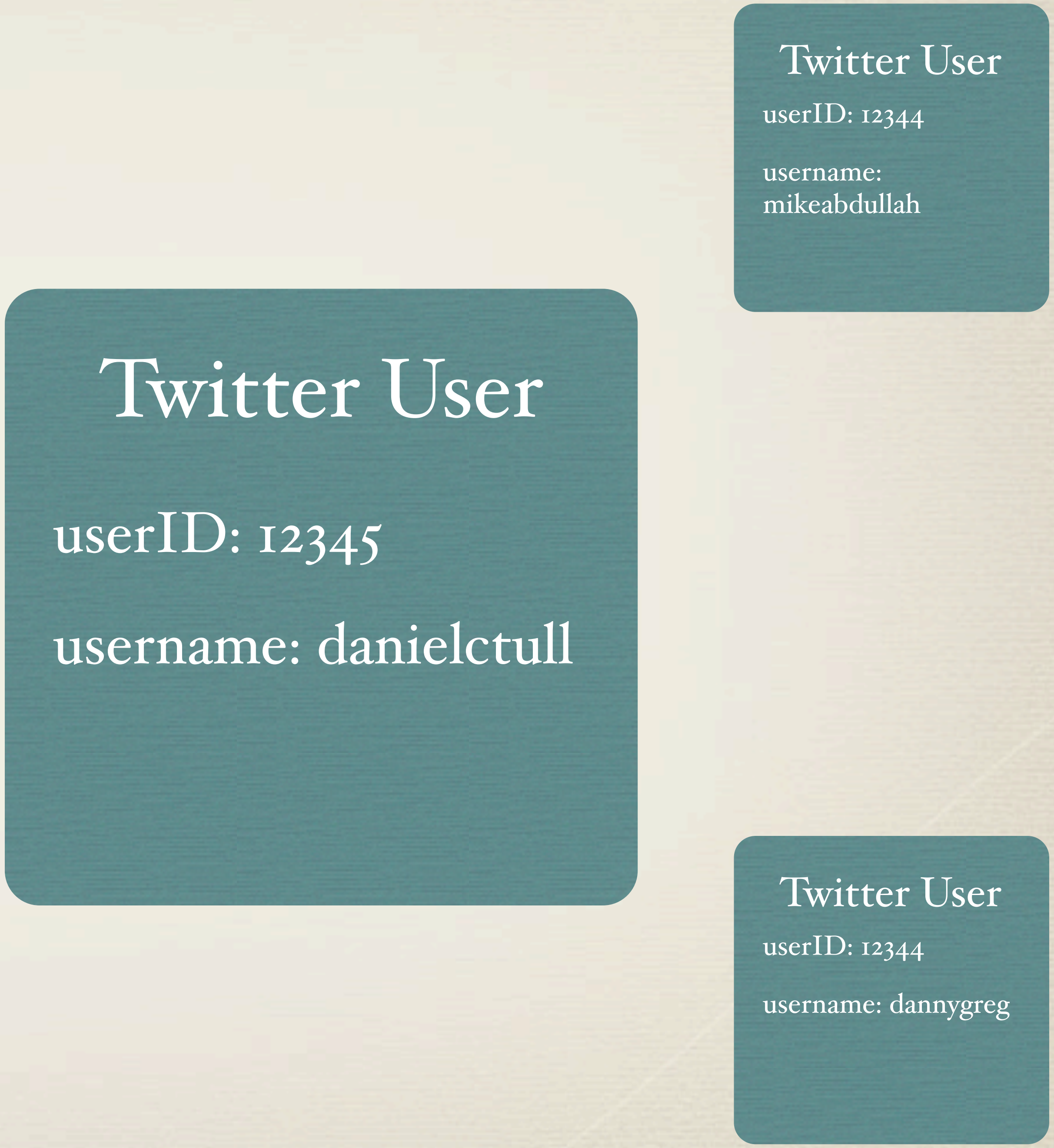
Twitter User

userID: 12344

username: dannygreg

Existing Objects

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



Twitter User

userID: 12345

username: danielctull

Twitter User

userID: 12344

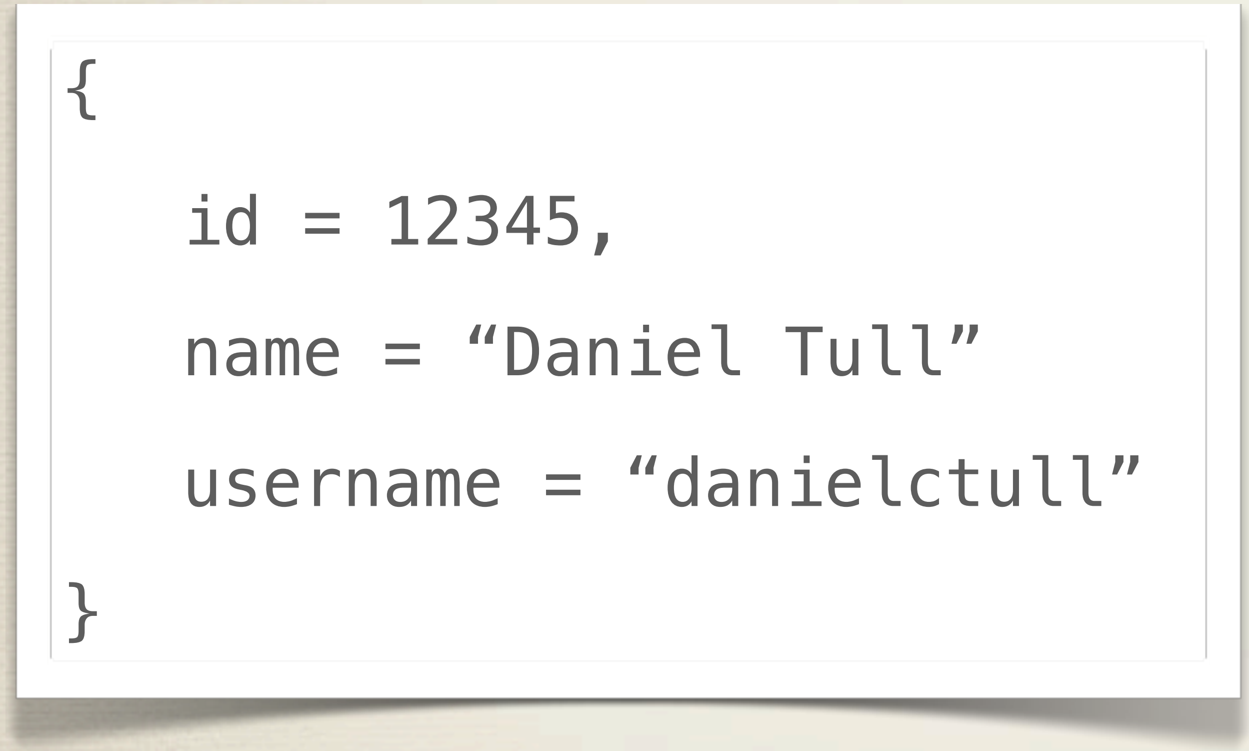
username: mikeabdullah

Twitter User

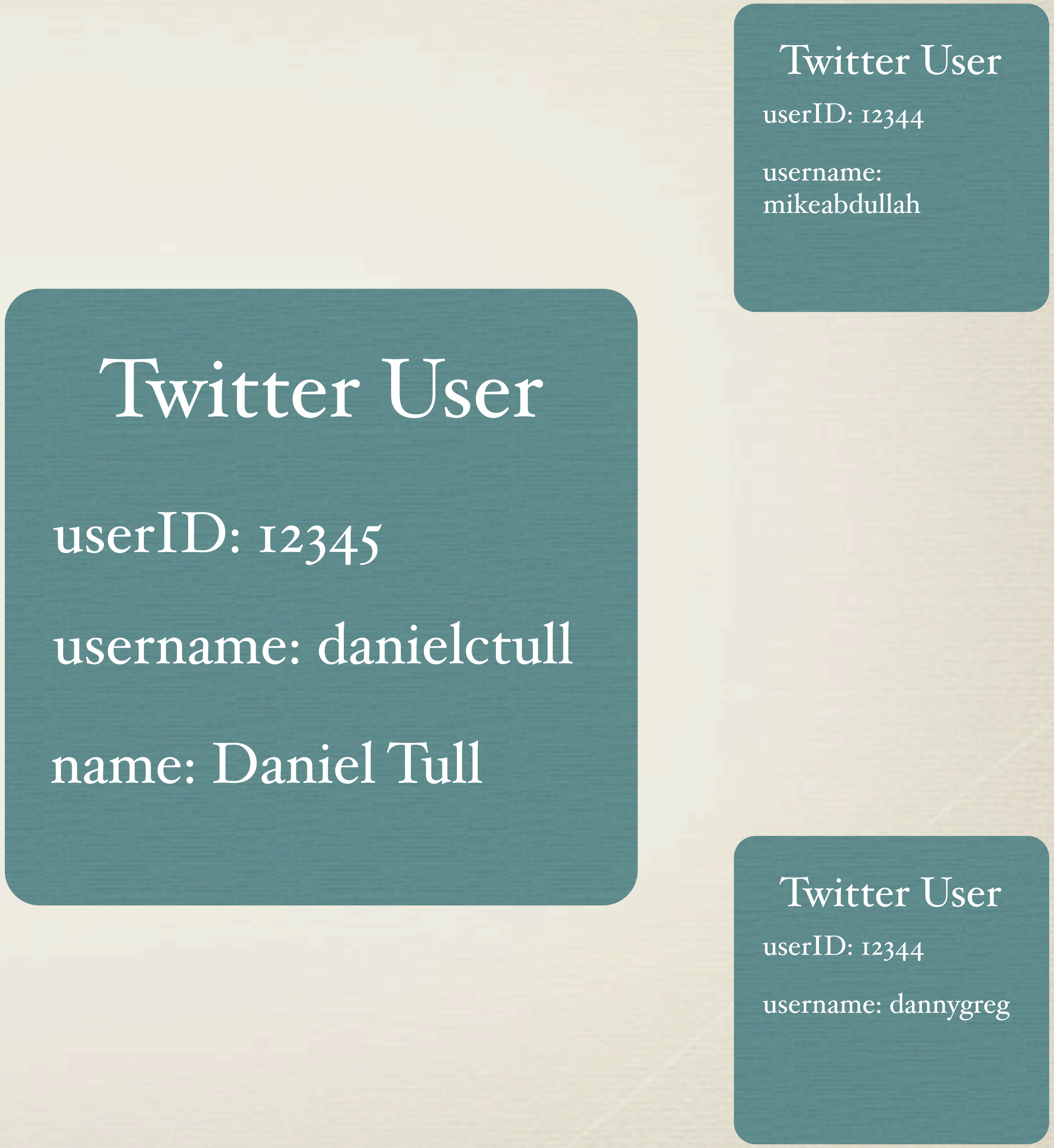
userID: 12344

username: dannygreg

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

Daniel Tull

@danielctull

github.com/danielctull/DCTCoreData

danieltull.co.uk/DCTCoreData/Documentation