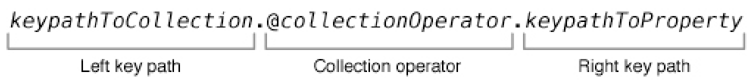
Using Collection Operators

When you send a key-value coding compliant object the `valueForKeyPath:` message, you can embed a *collection operator* in the key path. A collection operator is one of a small list of keywords preceded by an at sign (`@`) that specifies an operation that the getter should perform to manipulate the data in some way before returning it. The default implementation of `valueForKeyPath:` provided by `NSObject` implements this behavior.

When a key path contains a collection operator, any portion of the key path preceding the operator, known as the *left key path*, indicates the collection on which to operate relative to the receiver of the message. If you send the message directly to a collection object, such as an `NSArray` instance, the left key path may be omitted.

The portion of the key path after the operator, known as the *right key path*, specifies the property within the collection that the operator should work on. All the collection operators except `@count` require a right key path. Figure 4-1 illustrates the operator key path format.

Figure 4-1 Operator key path format



Collection operators exhibit three basic types of behavior:

- Aggregation Operators coalesce a collection's objects in some way, and return a single object that generally matches the data type of the property named in the right key path. The `@count` operator is an exception—it takes no right key path and always returns an `NSNumber` instance.
- Array Operators return an `NSArray` instance containing some subset of the objects held in the named collection.
- Nesting Operators work on collections that contain other collections, and return an `NSArray` or `NSSet` instance, depending on the operator, that combines the objects of the nested collections in some way.

Sample Data

The descriptions that follow include code snippets demonstrating how you invoke each operator, and the result of doing so. These rely on the `BankAccount` class, presented in Listing 2-1, which holds an array of `Transaction` objects. Each of these represents a simple checkbook entry, as declared in Listing 4-1.

Listing 4-1 Interface declaration for the `Transaction` object

```
@interface Transaction : NSObject

@property (nonatomic) NSString* payee;   // To whom
@property (nonatomic) NSNumber* amount;  // How much
@property (nonatomic) NSDate* date;      // When

@end
```

For the sake of discussion, assume your `BankAccount` instance has a transactions array populated with the data shown in Table 4-1, and that you make the example calls from inside the `BankAccount` object.

Table 4-1 Example data for the `Transactions` objects

`payee` values	`amount` values formatted as currency	`date` values formatted as month day, year
`Green Power`	`$120.00`	`Dec 1, 2015`
`Green Power`	`$150.00`	`Jan 1, 2016`
`Green Power`	`$170.00`	`Feb 1, 2016`
`Car Loan`	`$250.00`	`Jan 15, 2016`
`Car Loan`	`$250.00`	`Feb 15, 2016`
`Car Loan`	`$250.00`	`Mar 15, 2016`

payee values	amount values formatted as currency	date values formatted as month day, year
General Cable	$120.00	Dec 1, 2015
General Cable	$155.00	Jan 1, 2016
General Cable	$120.00	Feb 1, 2016
Mortgage	$1,250.00	Jan 15, 2016
Mortgage	$1,250.00	Feb 15, 2016
Mortgage	$1,250.00	Mar 15, 2016
Animal Hospital	$600.00	Jul 15, 2016

Aggregation Operators

Aggregation operators work on either an array or set of properties, producing a single value that reflects some aspect of the collection.

@avg

When you specify the `@avg` operator, `valueForKeyPath:` reads the property specified by the right key path for each element of the collection, converts it to a `double` (substituting 0 for `nil` values), and computes the arithmetic average of these. It then returns the result stored in an `NSNumber` instance.

To obtain the average transaction amount among the sample data in Table 4-1:

```
NSNumber *transactionAverage = [self.transactions valueForKeyPath:@"@avg.amount"];
```

The formatted result of `transactionAverage` is $456.54.

@count

When you specify the `@count` operator, `valueForKeyPath:` returns the number of objects in the collection in an `NSNumber` instance. The right key path, if present, is ignored.

To obtain the number of `Transaction` objects in `transactions`:

```
NSNumber *numberOfTransactions = [self.transactions valueForKeyPath:@"@count"];
```

The value of `numberOfTransactions` is 13.

@max

When you specify the `@max` operator, `valueForKeyPath:` searches among the collection entries named by the right key path and returns the largest one. The search conducts comparisons using the `compare:` method, as defined by many Foundation classes, such as the `NSNumber` class. Therefore, the property indicated by the right key path must hold an object that responds meaningfully to this message. The search ignores `nil` valued collection entries.

To obtain the maximum of the date values, which is the date of the latest transaction, among the transactions listed in Table 4-1:

```
NSDate *latestDate = [self.transactions valueForKeyPath:@"@max.date"];
```

The formatted `latestDate` value is Jul 15, 2016.

@min

When you specify the `@min` operator, `valueForKeyPath:` searches among the collection entries named by the right key path and returns the smallest one. The search conducts comparisons using the `compare:` method, as defined by many Foundation classes, such as the `NSNumber` class. Therefore, the property indicated by the right key path must hold an object that responds meaningfully to this message. The search ignores `nil` valued collection entries.

To obtain the minimum of the date values, which is the date of the earliest transaction, among the transactions listed in Table 4-1:

```
NSDate *earliestDate = [self.transactions valueForKeyPath:@"@min.date"];
```

The formatted `earliestDate` value is Dec 1, 2015.

@sum

When you specify the `@sum` operator, `valueForKeyPath:` reads the property specified by the right key path for each element of the collection, converts it to a `double` (substituting 0 for `nil` values), and computes the sum of these. It then returns the result stored in an `NSNumber` instance.

To obtain the sum of the transactions amount among the sample data in Table 4-1:

```
NSNumber *amountSum = [self.transactions valueForKeyPath:@"@sum.amount"];
```

The formatted result of `amountSum` is $5,935.00.

Array Operators

The array operators cause `valueForKeyPath:` to return an array of objects corresponding to a particular set of the objects indicated by the right key path.

> IMPORTANT
>
> The `valueForKeyPath:` method raises an exception if any of the leaf objects is `nil` when using array operators.

@distinctUnionOfObjects

When you specify the `@distinctUnionOfObjects` operator, `valueForKeyPath:` creates and returns an array containing the distinct objects of the collection corresponding to the property specified by the right key path.

To obtain a collection of `payee` property values for the transactions in `transactions` with duplicate values omitted:

```
NSArray *distinctPayees = [self.transactions
  valueForKeyPath:@"@distinctUnionOfObjects.payee"];
```

The resulting `distinctPayees` array contains one instance each of the following strings: Car Loan, General Cable, Animal Hospital, Green Power, Mortgage.

> NOTE
>
> The `@unionOfObjects` operator provides similar behavior, but without removing duplicate objects.

@unionOfObjects

When you specify the `@unionOfObjects` operator, `valueForKeyPath:` creates and returns an array containing all the objects of the collection corresponding to property specified by the right key path. Unlike @distinctUnionOfObjects, duplicate objects are not removed.

To obtain a collection of `payee` property values for the transactions in `transactions`:

```
NSArray *payees = [self.transactions valueForKeyPath:@"@unionOfObjects.payee"];
```

The resulting `payees` array contains the following strings: Green Power, Green Power, Green Power, Car Loan, Car Loan, Car Loan, General Cable, General Cable, General Cable, Mortgage, Mortgage, Mortgage, Animal Hospital. Note the duplicates.

> NOTE
> The `@distinctUnionOfArrays` operator is similar, but removes duplicate objects.

Nesting Operators

The nesting operators operate on nested collections, where each entry of the collection itself contains a collection.

> IMPORTANT
> The `valueForKeyPath:` method raises an exception if any of the leaf objects is `nil` when using nesting operators.

For the descriptions that follow, consider a second array of data called `moreTransactions`, populated with the data in Table 4-2, and collected together with the original `transactions` array (from the Sample Data section) into a nested array:

```
NSArray* moreTransactions = @[<# transaction data #>];
NSArray* arrayOfArrays = @[self.transactions, moreTransactions];
```

Table 4-2 Hypothetical `Transaction` data in the `moreTransactions` array

payee values	amount values formatted as currency	date values formatted as month day, year
General Cable - Cottage	$120.00	Dec 18, 2015
General Cable - Cottage	$155.00	Jan 9, 2016
General Cable - Cottage	$120.00	Dec 1, 2016
Second Mortgage	$1,250.00	Nov 15, 2016
Second Mortgage	$1,250.00	Sep 20, 2016
Second Mortgage	$1,250.00	Feb 12, 2016
Hobby Shop	$600.00	Jun 14, 2016

@distinctUnionOfArrays

When you specify the `@distinctUnionOfArrays` operator, `valueForKeyPath:` creates and returns an array containing the distinct objects of the combination of all the collections corresponding to the property specified by the right key path.

To obtain the dis

On This Page

```
NSArray *collectedDistinctPayees = [arrayOfArrays
  valueForKeyPath:@"@distinctUnionOfArrays.payee"];
```

The resulting `collectedDistinctPayees` array contains the following values: Hobby Shop, Mortgage, Animal Hospital, Second Mortgage, Car Loan, General Cable - Cottage, General Cable, Green Power.

> NOTE
> The `@unionOfArrays` operator is similar, but does not remove duplicate objects.

@unionOfArrays

When you specify the `@unionOfArrays` operator, `valueForKeyPath:` creates and returns an array containing the all the objects of the combination of all the collections corresponding to the property specified by the right key path, without removing duplicates.

To obtain the values of the `payee` property in all the arrays within `arrayOfArrays`:

```
NSArray *collectedPayees = [arrayOfArrays valueForKeyPath:@"@unionOfArrays.payee"];
```

The resulting `collectedPayees` array contains the following values: Green Power, Green Power, Green Power, Car Loan, Car Loan, Car Loan, General Cable, General Cable, General Cable, Mortgage, Mortgage, Mortgage, Animal Hospital, General Cable - Cottage, General Cable - Cottage, General Cable - Cottage, Second Mortgage, Second Mortgage, Second Mortgage, Hobby Shop.

> NOTE
>
> The `@distinctUnionOfArrays` operator is similar, but removes duplicate objects.

@distinctUnionOfSets

When you specify the `@distinctUnionOfSets` operator, `valueForKeyPath:` creates and returns an `NSSet` object containing the distinct objects of the combination of all the collections corresponding to the property specified by the right key path.

This operator behaves just like `@distinctUnionOfArrays`, except that it expects an `NSSet` instance containing `NSSet` instances of objects rather than an `NSArray` instance of `NSArray` instances. Also, it returns an `NSSet` instance. Assuming the example data had been stored in sets instead of arrays, the example call and results are the same as those shown for `@distinctUnionOfArrays`.

On This Page