NSDateFormatter

Inherits From:	NSFormatter : NSObject
Conforms To:	NSObject (NSObject) NSCoding NSCopying
Declared In:	Foundation/NSDateFormatter.h

Class Description

Instances of NSDateFormatter format the textual representation of cells that contain NSDates (including NSCalendarDates), and convert textual representations of dates and times into NSDates. You can express the representation of dates and times very flexibly: “Thu 22 Dec 1994” is just as acceptable as “12/22/94”. With natural-language processing for dates enabled, users can also express dates colloquially, such as “today,” “day after tomorrow,” and “a month from today.”

To use an NSDateFormatter, allocate an instance of it and initialize it with **initWithDateFormat:allowNaturalLanguage:** In the first argument use **strftime-**style conversion specifiers to compose the format string for textual representation. (For more information on these specifiers, see the see the description of NSCalendarDate’s **dateWithCalendarFormat:timeZone:** method.) Then use NSCell's **setFormatter:** method to associate the NSDateFormatter object with a cell. The value of a cell (NSCell) is represented by an object, typically an NSDate object in this case. When this value needs to be displayed or edited, the cell passes its object to the NSDateFormatter instance, which returns the formatted string. When the user enters a string, or when one is programmatically written in a cell (using **setStringValue:**), the cell obtains the equivalent NSDate object from the NSDateFormatter.

NSControl provides delegation methods that permit you to validate cell contents and to handle errors in formatting. See the specification of the NSFormatter class for details.

When a cell with a NSDateFormatter is copied, the new cell retains the NSDateFormatter object instead of copying it. You remove an NSDateFormatter from a cell by specifying **nil** as the argument of **setFormatter:**.

Instances of NSDateFormatter are immutable.

Method Types

Initializing an NSDateFormatter	– initWithDateFormat:allowNaturalLanguage:

Determining attributes	– allowsNaturalLanguage – dateFormat

nstance Methods

allowsNaturalLanguage

– (BOOL)**allowsNaturalLanguage**

Returns YES if the NSDateFormatter attempts to process dates entered as a vernacular string (“today,” “day before yesterday,” and so on). Returns NO if the NSDateFormatter does not do any natural-language processing of these date expressions.

dateFormat

– (NSString *)**dateFormat**

Returns the date format string used by an NSDateFormatter object. See the description of NSCalendarDate’s **dateWithCalendarFormat:timeZone:** for a list of the conversion specifiers permitted in date format strings.

initWithDateFormat:allowNaturalLanguage

– (id)**initWithDateFormat:**(NSString *)*format*
allowNaturalLanguage:(BOOL)*flag*

Initializes and returns an NSDateFormatter instance that uses the date *format* in its conversions. See the description of NSCalendarDate’s **dateWithCalendarFormat:timeZone:** for a list of conversion specifiers permitted in date format strings. Set *flag* to YES if you want the NSDateFormatter to process dates entered as expressions in the vernacular (for example, “tomorrow”); NSDateFormatter attempts natural-language processing only after it fails to interpret an entered string according to *format*. The following example creates a date formatter with the format string (as example) “Mar 15 1994” and then associates the formatter with the cells of a form (**contactsForm**).

```
NSDateFormatter *dateFormat = [[NSDateFormatter alloc]
   initWithDateFormat:@"%b %d %Y” allowNaturalLanguage:NO];
[[contactsForm cells] makeObjectsPerform:@selector(setFormatter:)
    withObject:dateFormat];
```