NSDate

Inherits From: NSObject

Conforms To: NSCoding, NSCopying
NSObject (NSObject)

Declared In: Foundation/NSDate.h

Class Description

NSDate is an abstract class that provides behavior for creating dates, comparing dates, representing dates, computing time intervals, and similar functionality. It presents a programmatic interface through which suitable date objects are requested and returned. NSDate objects are lightweight and immutable since they represent a invariant point in time. This class is designed to provide the foundation for arbitrary calendrical representations. Its subclass NSCalendarDate offers date objects that are suitable for representing dates according to western calendrical systems.

"Date" as used above implies clock time as well. The standard unit of time for date objects is a value typed as NSTimeInterval (a **double**) and expressed as seconds. The NSTimeInterval type makes possible a wide and fine-grained range of date and time values, giving accuracy within milliseconds for dates 10,000 years apart.

NSDate and its subclasses compute time as seconds *relative* to an absolute reference date. This reference date is the first instant of January 1, 2001. NSDate converts all date and time representations to and from NSTimeInterval values that are relative to this absolute reference date. A positive interval relative to a date represents a point in the future, a negative interval represents a time in the past.

Note: Conventional UNIX systems implement time according to the Network Time Protocol (NTP) standard, which is based on Coordinated Universal Time. The private implementation of NSDate follows the NTP standard. However, this standard doesn't account for leap seconds and therefore isn't synchronized with International Atomic Time (the most accurate).

Like various other Foundation classes, NSDate lets you obtain operating-system functionality (dates and times) without depending on operating-system internals. It also provides a basis for the NSRunLoop and NSTimer classes, which use concrete date objects to implement local event loops and timers.

NSDate's sole primitive method, **timeIntervalSinceReferenceDate**, provides the basis for all the other methods in the NSDate interface. It returns a time value relative to an absolute reference date.

Using NSDate

The date objects dispensed by NSDate give you a diverse range of date and time functionality. To obtain dates, send one of the **date...** messages to the NSDate class object. One of the most useful is **date** itself, which returns a date object representing the current date and time. You can get new date objects with date and time values adjusted from existing date objects by sending **addTimeInterval:**.

You can obtain relative date information by sending the **timeInterval...** messges to a date object. For instance, **timeIntervalSinceNow** gives you the time, in seconds, between the current time and the receiving date object. Compare dates with the **isEqual:**, **compare:**, **laterDate:**, and **earlierDate:** methods and use the **description** method to obtain a string object that represents the date in a standard international format.

Creating an NSDate Object

+ (NSDate *)**date**	Creates and returns an NSDate set to the current date and time.
+ (NSDate *)**dateWithTimeIntervalSinceNow:**(NSTimeInterval)*seconds*	Creates and returns an NSDate set to *seconds* seconds from the current date and time.
+ (NSDate *)**dateWithTimeIntervalSince1970:**(NSTimeInterval)*seconds*	Creates and returns an NSDate set to to *seconds* seconds from the reference date used by UNIX® systems. Use a negative argument value to specify a date and time before the reference date.
+ (NSDate *)**dateWithTimeIntervalSinceReferenceDate:**(NSTimeInterval)*seconds*	Creates and returns an NSDate set to *seconds* seconds from the absolute reference date (the first instant of 1 January, 2001). Use a negative argument value to specify a date and time before the reference date.
+ (NSDate *)**distantFuture**	Creates and returns an NSDate that represents a date in the distant future (in terms of centuries). You can use this object in your code as a control date, a guaranteed outer temporal limit.
+ (NSDate *)**distantPast**	Creates and returns an NSDate that represents a date in the distant past (in terms of centuries). You can use this object in your code as a control date, a guaranteed temporal boundary.
– (id)**init**	Initializes a newly allocated NSDate to the current date and time.
– (id)**initWithString:**(NSString *)*description*	Returns an NSDate with a date and time value specified by the international string-representation format: YYYY-MM-DD HH:MM:SS ±HHMM, where ±HHMM is a time zone offset in hours and minutes from Greenwich Mean Time.
– (NSDate *)**initWithTimeInterval:**(NSTimeInterval)*seconds* **sinceDate:**(NSDate *)*anotherDate*	Returns an NSDate initialized relative to another date object by *seconds* (plus or minus).

– (NSDate *)**initWithTimeIntervalSinceNow:**(NSTimeInterval)*seconds*

Returns an NSDate initialized relative to the current date and time by *seconds* (plus or minus).

– (id)**initWithTimeIntervalSinceReferenceDate:**(NSTimeInterval)*seconds*

Returns an NSDate initialized relative to the reference date and time by *seconds* (plus or minus).

Converting to an NSCalendar Object

– (NSCalendarDate *)**dateWithCalendarFormat:**(NSString *)*formatString* **timeZone:**(NSTimeZone *)*timeZone*

Returns an NSCalendarDate object bound to the format string *formatString* and the time zone *timeZone*. If you specify **nil** after either or both of these arguments, the default format string and time zone are assumed.

Representing Dates

– (NSString *)**description**

Returns a string representation of the receiver. The representation conforms to the international format YYYY-MM-DD HH:MM:SS ±HHMM, where ±HHMM represents the time-zone offset in hours and minutes from Greenwich Mean Time (GMT).

– (NSString *)**descriptionWithCalendarFormat:**(NSString *)*formatString* **timeZone**:(NSTimeZone *)*aTimeZone* **locale:**(NSDictionary *)*localeDictionary*

Returns a string representation of the receiver. The representation conforms to *formatString* (a **strftime**-style date-conversion string) and is adjusted to *aTimeZone*. Included are the keys and values that represent the locale data from *localeDictionary*. For information on creating a locale dictionary, see the class description in NSUserDefaults.

– (NSString *)**descriptionWithLocale:**(NSDictionary *)*localeDictionary*

Returns a string representation of receiver (see **description**). Included are the key and values that represent the locale data from *localeDictionary*. For information on creating a locale dictionary, see the class description in NSUserDefaults.

Adding and Getting Intervals

+ (NSTimeInterval)**timeIntervalSinceReferenceDate**	Returns the interval between the system's absolute reference date and the current date and time. This value is less than zero until the first instant of 1 January 2001.
– **addTimeInterval:**(NSTimeInterval)*seconds*	Returns an NSDate that's set to a specified number of seconds relative to the receiver.
– (NSTimeInterval)**timeIntervalSince1970**	Returns the interval between the receiver and the reference date used by UNIX® systems.
– (NSTimeInterval)**timeIntervalSinceDate:**(NSDate *)*anotherDate*	Returns the interval between the receiver and *anotherDate*.
– (NSTimeInterval)**timeIntervalSinceNow**	Returns the interval between the receiver and the current date and time.
– (NSTimeInterval)**timeIntervalSinceReferenceDate**	Returns the interval between the receiver and the system's absolute reference date. This value is less than zero until the first instant of 1 January 2001.

Comparing Dates

– (NSComparisonResult)**compare:**(NSDate *)*anotherDate*	Compares the receiver's date to that of *anotherDate* and returns NSOrderedDescending if the receiver is temporally later, NSOrderedAscending if it's temporally earlier, and NSOrderedSame if they are equal.
– (NSDate *)**earlierDate:**(NSDate *)*anotherDate*	Compares the receiver's date to *anotherDate* and returns the one that's temporally earlier.
– (BOOL)**isEqualToDate:**(NSDate *)*anotherDate*	Returns YES if *anotherDate* and the receiver are within one second of each other; otherwise, returns NO.
– (NSDate *)**laterDate:**(NSDate *)*anotherDate*	Compares the receiver's date to *anotherDate* and returns the one that's temporally later.