Parsing Dates Without Times

This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator.

108

Parsing fixed-format date strings is tricky. For an explanation as to why, see QA1480 NSDateFormatter and Internet Dates. However, there's an extra gotcha when you try to parse fixed-format date strings that don't include a time. The posts below explain this gotcha and how to get around it.

If you have questions about any of this, please start a new thread here on DevForums. Make sure to tag it with *Foundation* so that I notice your post.

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@apple.com"`

Foundation

Asked 7 months ago by eskimo

Reply to this question

Replies

Parsing fixed-format date strings is tricky. For an explanation as to why, see QA1480 NSDateFormatter and Internet Dates. However, there's an extra wrinkle when you try to parse fixed-format date strings that don't include a time. Consider this code:

```
func startOfDateInSaoPaulo(_ dateStr: String) -> Date? {
    let df = DateFormatter()
    df.locale = Locale(identifier: "en_US_POSIX")
    df.timeZone = TimeZone(identifier: "America/Sao_Paulo")!
    df.dateFormat = "yyyyMMdd"
    return df.date(from: dateStr)
}
```

The goal here is to parse a string of the form `yyyyMMdd` and return the start of that day in São Paulo, Brazil. And the code seems to work. For example:

```
print(startOfDateInSaoPaulo("20200722")?.description ?? "nil")
// -> 2020-07-22 03:00:00 +0000
```

It even handles daylight saving time changes. São Paulo set the clocks forward on 4 Nov 2018, and you can see this change when you map 3 Nov and 5 Nov:

```
print(startOfDateInSaoPaulo("20181103")?.description ?? "nil")
// -> 2018-11-03 03:00:00 +0000
print(startOfDateInSaoPaulo("20181105")?.description ?? "nil")
// -> 2018-11-05 02:00:00 +0000
```

Note Time zones in Brazil are very exciting. If you're curious, read Time in Brazil and Daylight saving time in Brazil. Also, for those Northern hemisphere folks out there, keep in mind that São Paulo is in the southern hemisphere and thus the daylight saving "spring forward" happens in the second half of the year.

However, consider this:

```
print(startOfDateInSaoPaulo("20181104")?.description ?? "nil")
// -> nil
```

Whoah!?! This is failing because, internally, the date formatter maps the date string to a set of date components (`DateComponents`). The year, month, and day come from the date string, but the hour, minute, and second default to 0. In Brazil, daylight saving time starts at midnight, and thus the date components `year: 2018, month: 11, day: 4, hour: 0, minute: 0, second: 0` don't exist in the São Paulo time zone. When the date formatter runs these components through the calendar, it returns `nil`.

Note Making daylight saving time changes at midnight is weird but Brazil is not the only country that does this.

Posted 7 months ago by eskimo

The solution here is to set the `defaultDate` property on the date formatter to something in the middle of the day. For example:

```
func middleOfDateInSaoPaulo(_ dateStr: String) -> Date? {
    let df = DateFormatter()
    // This translates to midday, Sao Paulo time, on 2001-01-01.
    df.defaultDate = Date(timeIntervalSinceReferenceDate: 50400.0)
    df.locale = Locale(identifier: "en_US_POSIX")
    df.timeZone = TimeZone(identifier: "America/Sao_Paulo")!
    df.dateFormat = "yyyyMMdd"
    return df.date(from: dateStr)
}
```

Now all the dates centred around 4 Nov 2018 work just fine:

```
print(middleOfDateInSaoPaulo("20181103")?.description ?? "nil")
// -> 2018-11-03 15:00:00 +0000
print(middleOfDateInSaoPaulo("20181104")?.description ?? "nil")
// -> 2018-11-04 14:00:00 +0000
print(middleOfDateInSaoPaulo("20181105")?.description ?? "nil")
// -> 2018-11-05 14:00:00 +0000
```

If necessary, you can call startOfDay(for:) to get the a `Date` value that represents the start of the day.

In many cases, however, it's better to avoid this problem by working with a set of date components rather than a date. Remember that a `Date` value represents an absolute point in time, and that's not always the best model for a day. For example, if you're storing someone's birthday, it's better to store date components rather than a date.

And this brings us back to a key piece of advice from QA1480:

> look at solutions outside of the Cocoa space

If your final goal is to get a set of date components, you don't need a date formatter at all. Rather, simply parse the fixed-format string yourself:

```
func dateComponentsOfDate(_ dateStr: String) -> DateComponents? {
    let yearStr = dateStr.prefix(4)
    let monthStr = dateStr.dropFirst(4).prefix(2)
    let dayStr = dateStr.dropFirst(6).prefix(2)
    guard
        let year = Int(yearStr),
        let month = Int(monthStr),
        let day = Int(dayStr)
    else { return nil }
    return DateComponents(year: year, month: month, day: day)
}

print(dateComponentsOfDateInSaoPaulo("20181104")?.description ?? "nil")
// -> year: 2018 month: 11 day: 4 …
```

So, to summarise:

- Converting fixed-format strings that represent a date without a time is tricky.
- Unless you can *guarantee* that you're working in a time zone that has not, **and will never**, do daylight saving time changes at midnight, you must use the `defaultDate` technique shown above.
- Alternatively, you can avoid this problem entirely by working with date components rather than dates.

Posted 7 months ago by eskimo