Complications and Time Travel

CS193W - Spring 2016 - Lecture 5

Clock Faces


X-LARGE
10
:09
UTILITY
ROM 4:09
30°
NO UPCOMING EVENTS
MOTION
10:09
11
MICKEY
ROM 4:09
30°
NO UPCOMING EVENTS
SIMPLE
30°
ROM 4:09
CHRONOGRAPH
MON 11
ROM 4:09
SOLAR
MON 11
10:09
COLOR
11
15:00
ASTRONOMY
MON 11
10:09
MODULAR
MON
11
10:09
Calendar
No upcoming
events
30°
ROM
4:09

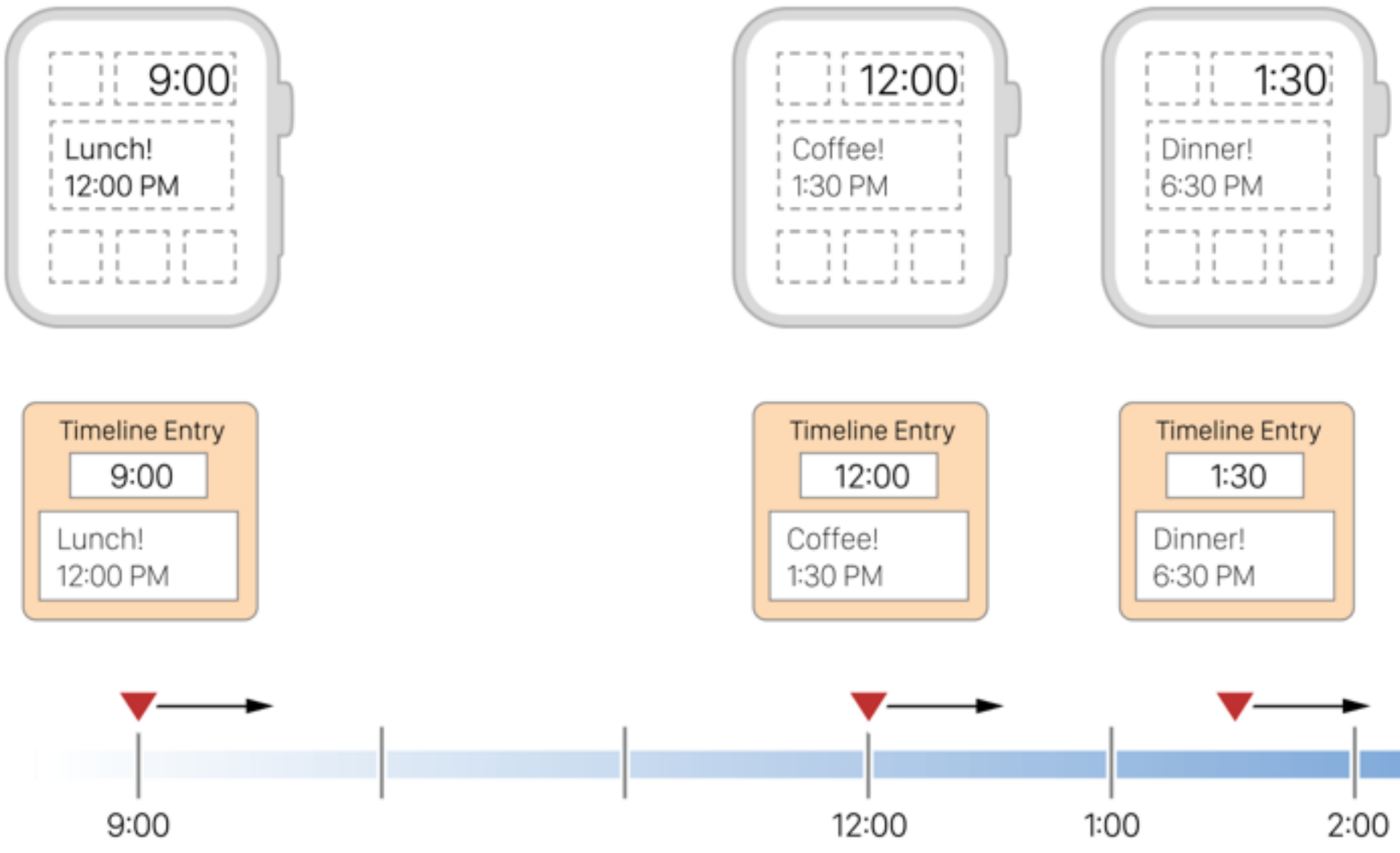
Complications

Time Travel

- By turning the digital crown, users can go forward and backward in time; the complications change to reflect their value at the selected time.

- For example, the user could see the weather forecast for a certain time of the day, or view upcoming meetings.

Time Travel Example



UI Responsiveness

- Complications must be shown immediately when the user raises her wrist

- Time Travel requires that the complications change immediately in sync with the time

- But, complications may depend on server data, and network access is slow

- Solution: complication data is prefetched by the OS periodically

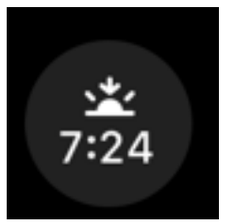
Complication Templates

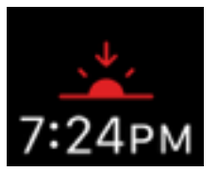
- Rather than using storyboards, complications use templates, where each template is a subclass of `CLKComplicationTemplate`.

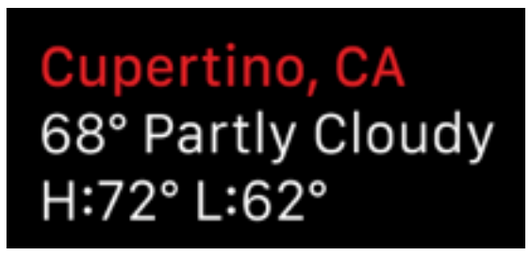
- There are 22 subclasses of `CLKComplicationTemplate`.

- The 22 subclasses can be grouped into 5 families.

Complication Template Families

- Circular 

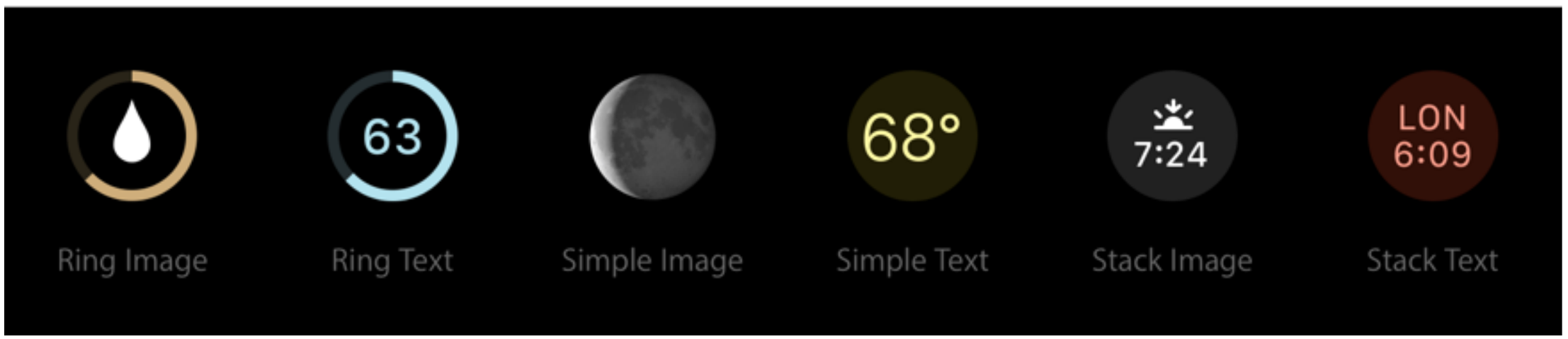
- Modular small 

- Modular large 

- Utility small 

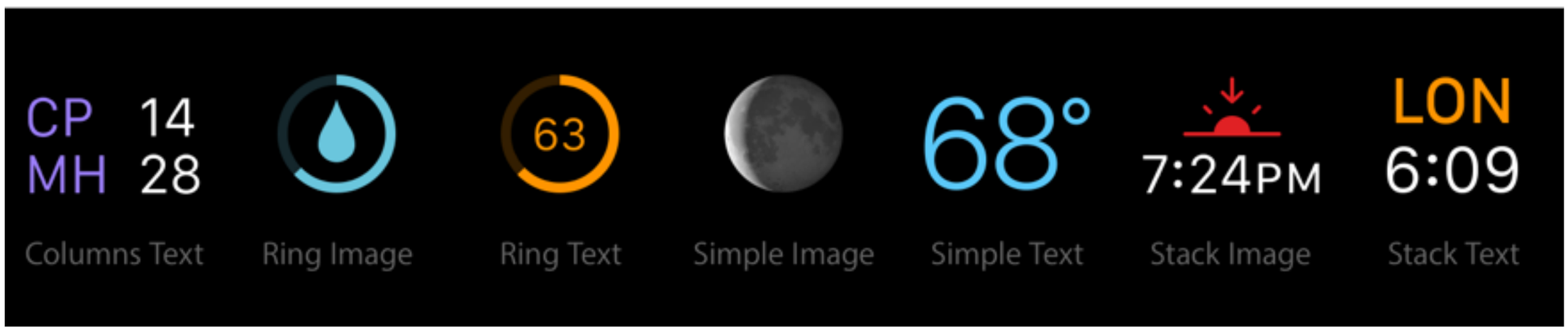
- Utility large 

Circular Small



`CLKComplicationTemplateCircularSmallRingImage`, etc.

Modular Small



Modular Large

CAL 396/660
MIN 13/30
HOUR 3/12

Columns

Cupertino, CA
68° Partly Cloudy
H:72° L:62°

Standard Body

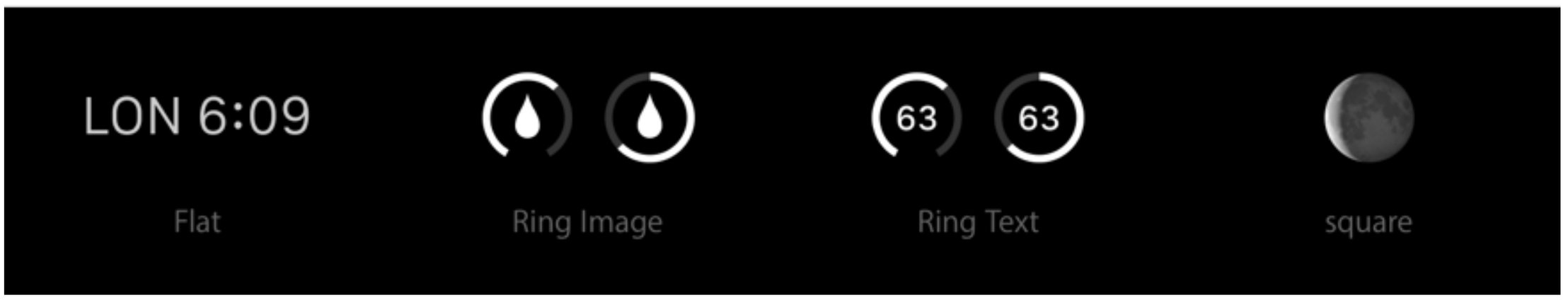
Final Score
14 Central Prep
28 Mission High

Table

Wednesday
Mar 9

Tall Body

Utility Small

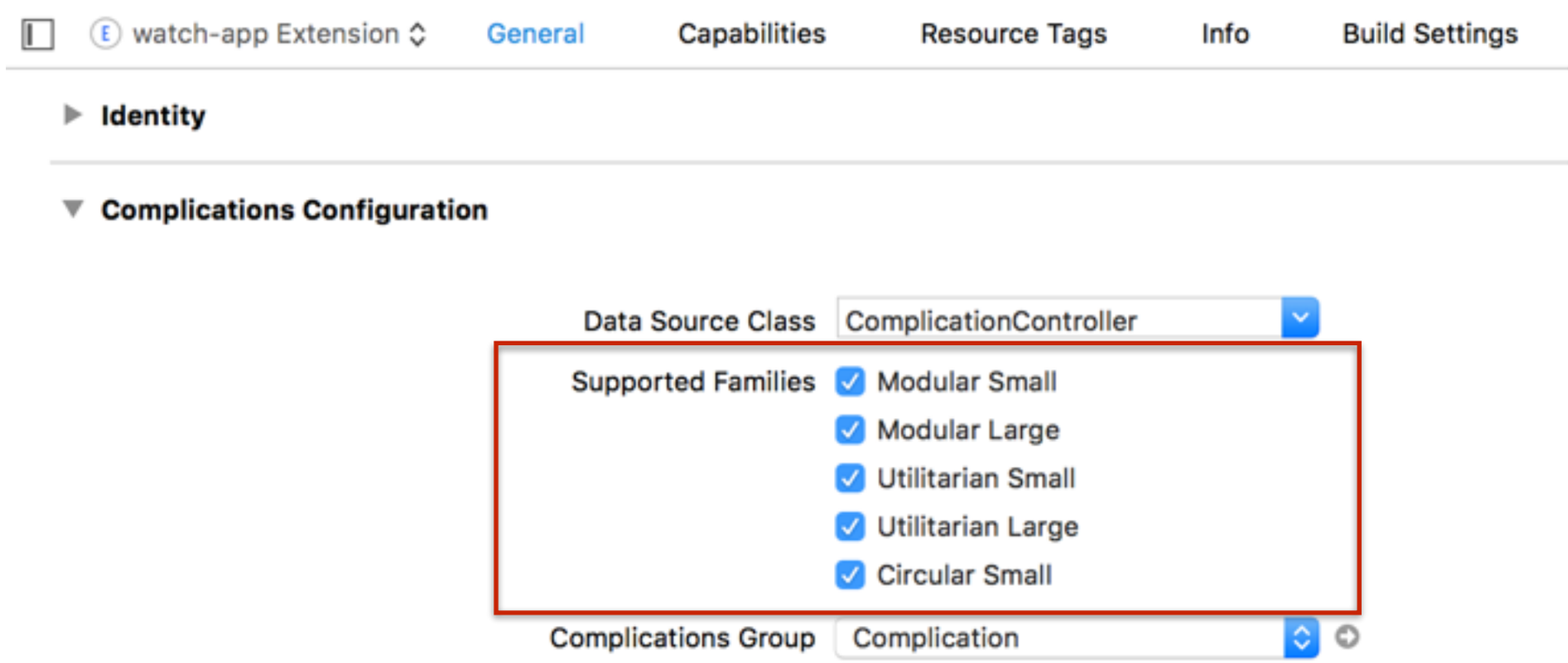


Utility Large

11:00AM PHOTO SHOOT

Large Flat

Specifying Supported Families



CLKComplication

- Represents a complication
- Has just one exposed property, `family`

```
var family: CLKComplicationFamily

enum CLKComplicationFamily : Int {
    case ModularSmall
    case ModularLarge
    case UtilitarianSmall
    case UtilitarianLarge
    case CircularSmall
}
```

Providers

- ClockKit uses text and image providers to provide data for complications
- The providers are smart about how to display their information in a small area

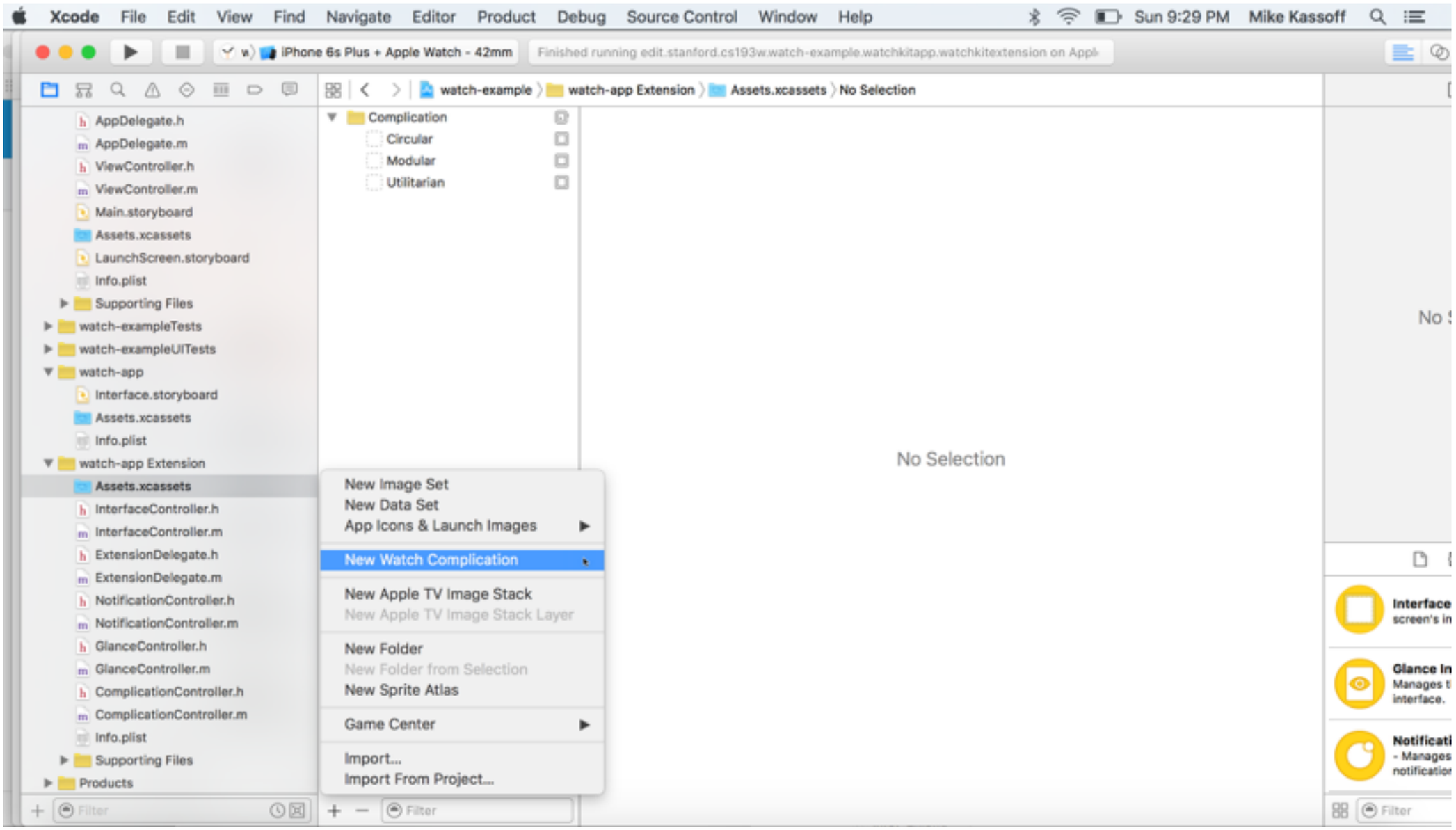
Thursday, November 23
Thur, November 23
Thur, Nov 23
Nov 23
23

CLKImageProvider

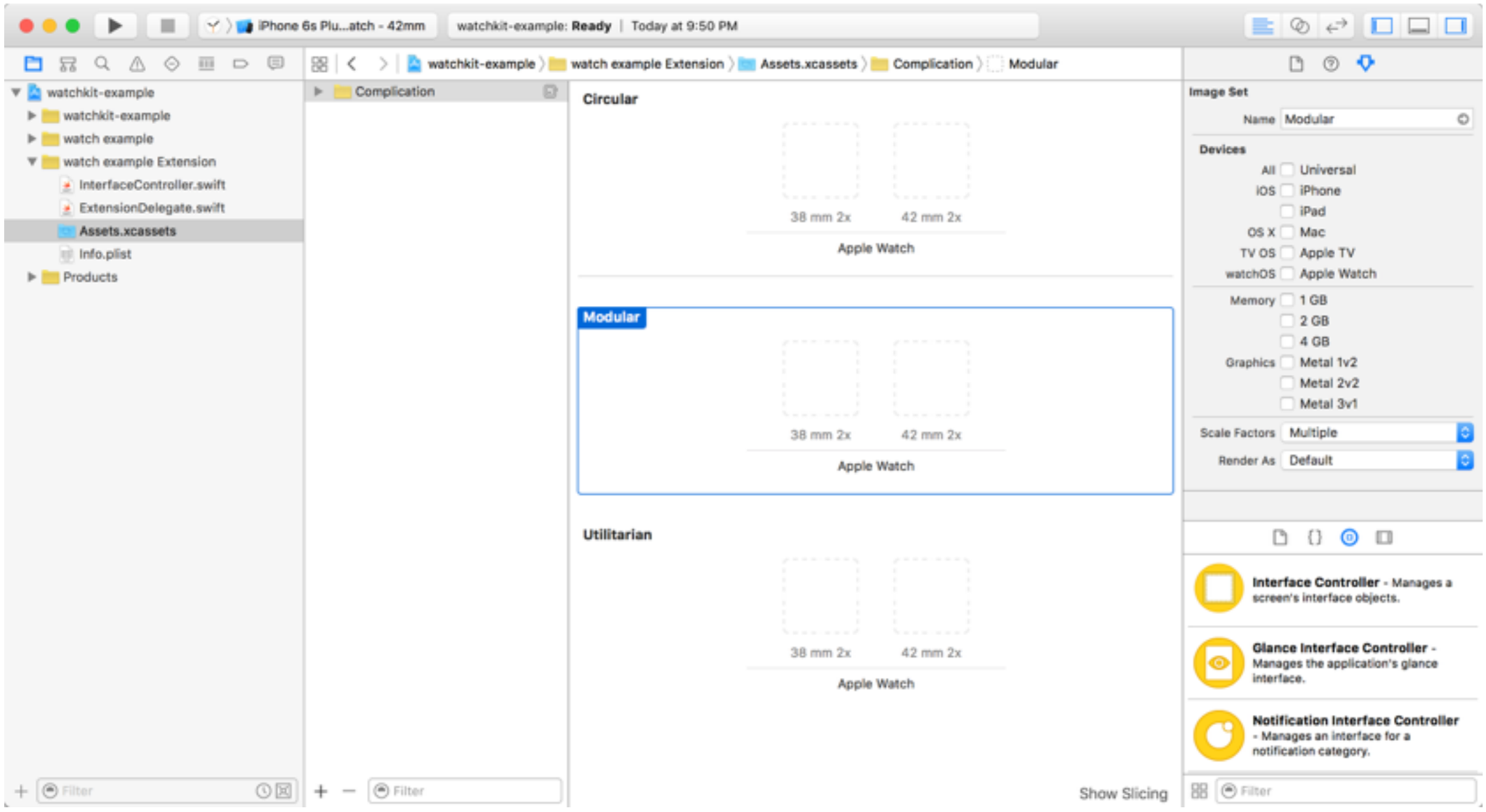
```
convenience init(onePieceImage onePieceImage: UIImage,
twoPieceImageBackground twoPieceImageBackground: UIImage?,
twoPieceImageForeground twoPieceImageForeground: UIImage?)

@property(nonatomic) UIColor *tintColor
```

- Images are tinted based on their alpha channel
- For two piece images, the background is tinted and the foreground is always white
- In monochrome environments, the one piece image is used and it is tinted to the watch face's color

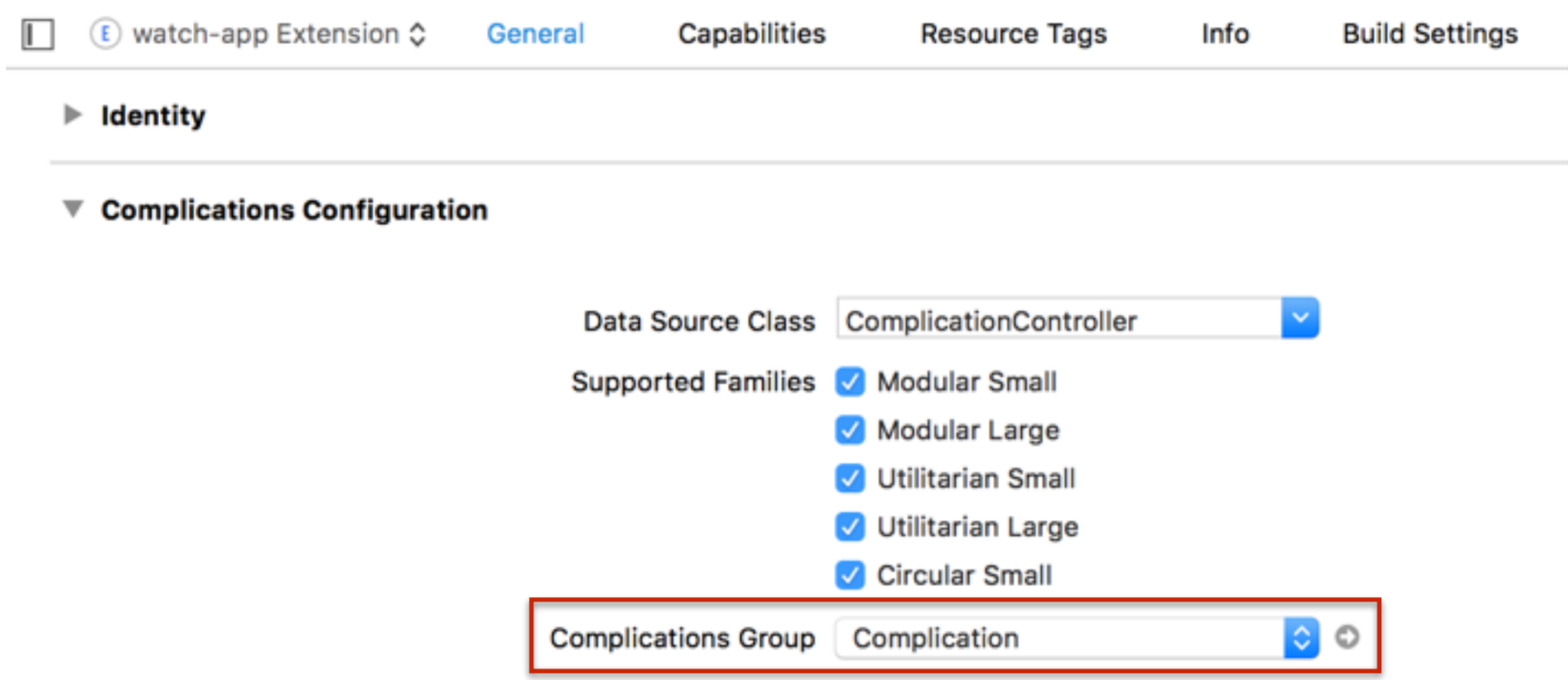
Adding Complication Assets



Compilation Assets



Set the Complications Group



CLKTextProvider

- The superclass of several other providers:

```
CLKSimpleTextProvider
CLKDateTextProvider
CLKRelativeDateTextProvider
CLKTimeIntervalTextProvider
CLKTimeTextProvider
```

- It has one property, `tintColor`

CLK**Simple**TextProvider

```
convenience init(text text: String,
                 shortText shortText: String?)
```

Shows `text` if it fits,
otherwise `shortText`,
otherwise a truncated version of `text`

CLK**Time**TextProvider

```
convenience init(date date: NSDate,
                 timeZone timeZone: NSTimeZone?)
```

Like all the other date/time-related providers, it is localized.

```
10:09AM
10:09
```

CLK**Date**TextProvider

```
convenience init(date date: NSDate,
                 units calendarUnits: NSCalendarUnit,
                 timeZone timeZone: NSTimeZone?)
```

`calendarUnits` can contain one or more of the following:

```
NSDayCalendarUnit
NSMonthCalendarUnit
NSWeekdayCalendarUnit
NSYearCalendarUnit
```

```
Saturday, December 28, 2015
Saturday, December 28
Saturday, Dec 28
Sat, Dec 28
Dec 28
28
```

CLK**TimeInterval**TextProvider

```
convenience init(startDate startDate: NSDate,
                 endDate endDate: NSDate,
                timeZone timeZone: NSTimeZone?)
```

Appropriate for large templates, e.g. modular large and utilitarian large.
In small templates only the beginning date will be shown.

```
9:30AM – 3:30PM
9:30 – 10:30AM
Jan 1 – Jan 7
1/1 – 1/7
```

CLK**RelativeDate**TextProvider

```
convenience init(date date: NSDate,
                 style style: CLKRelativeDateStyle,
                 units calendarUnits: NSCalendarUnit)
```

Instead of using a timeline, shows the time differential between the current date and the given date.

`Natural`
Displays units broken up
`10hrs 9min`

`Offset`
Can display a single time unit
`10 hours`

`Timer`
Shows number of hours, minutes, and seconds between dates
`10:09`

CLKComplicationServer

Use `CLKComplicationServer.sharedInstance()` to manage the active complications in an app

```
var activeComplications: [CLKComplication]
```

Returns the complications on the current clock face

CLKComplicationServer: Time Travel Date Range

```
var earliestTimeTravelDate: NSDate
var latestTimeTravelDate: NSDate
```

Returns the earliest and latest dates for which you need to provide time travel entries

CLKComplicationServer: Managing Timelines

```
func reloadTimelineForComplication(_ complication: CLKComplication!)
func extendTimelineForComplication(_ complication: CLKComplication!)
```

Tell ClockKit that the timeline has changed

Daily Time Allotment

- To save battery life, each complication is given a daily time allotment for computation

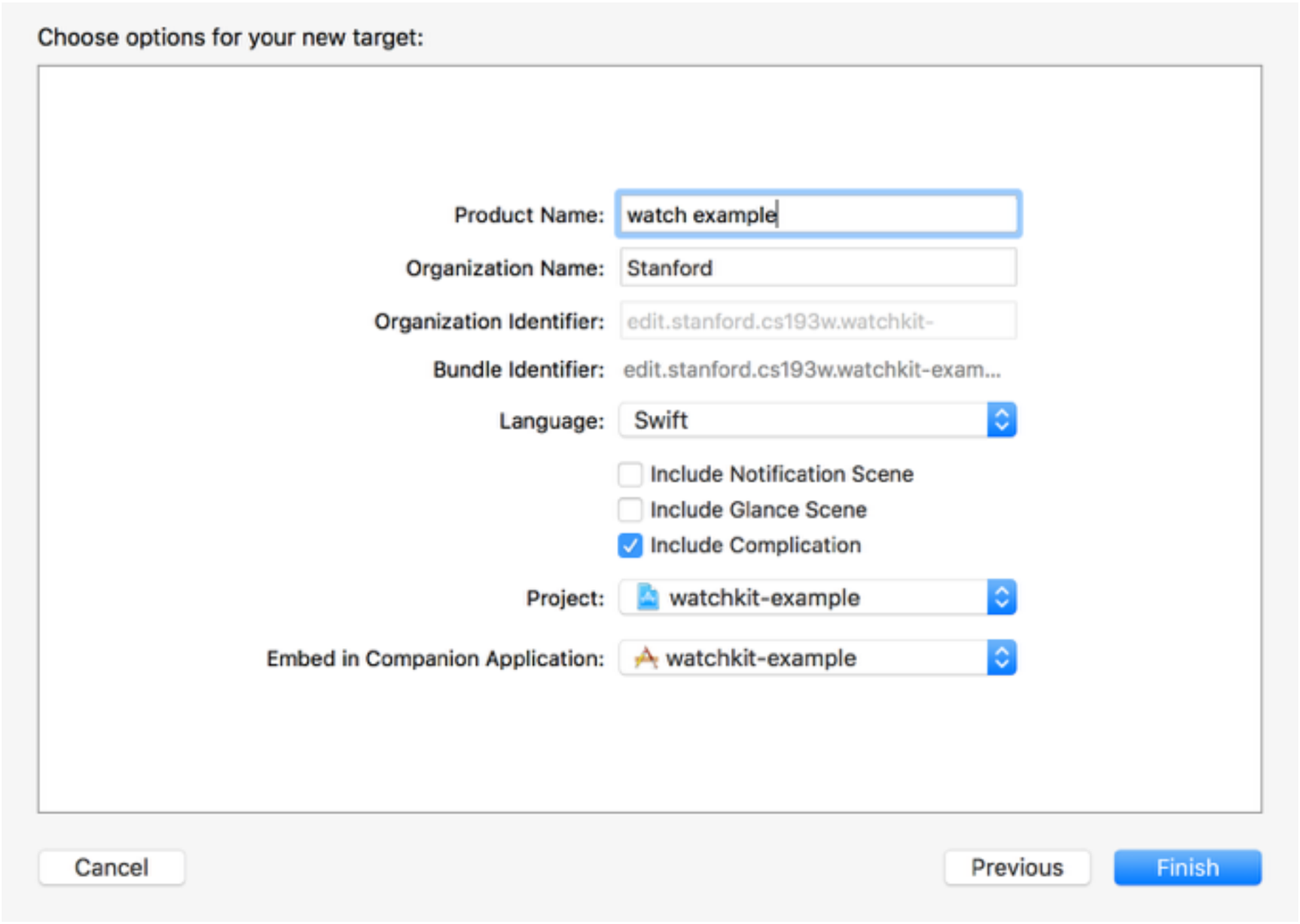
- In short: don't change your timeline too many times during the day. Update it every few hours or once a day, not every few minutes.

- Note that your timeline can still have many entries during the day; just don't call `reloadTimelineForComplication` and `extendTimelineForComplication` timeline too often.

CLKComplicationDataSource

- Provides data for your populating your complication's timeline
- Provides methods for scheduling future timeline updates

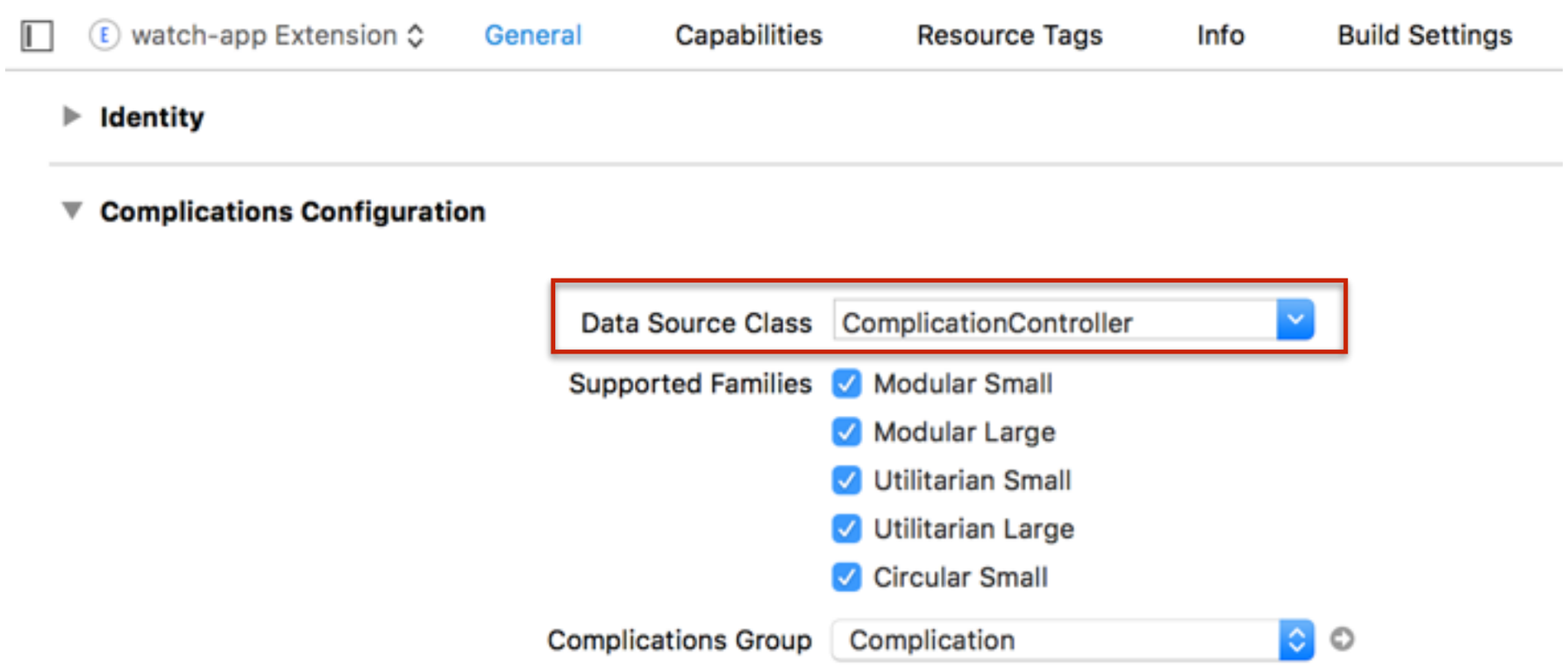
Adding a Complication Data Source to your App



Or, Add One Manually Later

1. Create a class that implements the `CLKComplicationDataSource` protocol

2. Specify the name of that class in your watch app target settings



Time Travel Directions

```
    func getSupportedTimeTravelDirectionsForComplication(complication: CLKComplication,
withHandler handler: (CLKComplicationTimeTravelDirections) -> Void) {
        handler([.Forward, .Backward])
    }
```

Timeline Start and End Dates

```
    func getTimelineStartDateForComplication(complication: CLKComplication,
withHandler handler: (NSDate?) -> Void) {
        handler(NSDate(timeInterval: -60*60*24, sinceDate: NSDate()))
    }
```

```
    func getTimelineEndDateForComplication(complication: CLKComplication,
withHandler handler: (NSDate?) -> Void) {
        handler(NSDate(timeInterval: 60*60*24, sinceDate: NSDate()))
    }
```

You can call `earliestTimeTravelDate` / `latestTimeTravelDate` on `CLKComplicationServer.sharedInstance` to determine the maximum range you need to care about

CLKComplicationTimelineEntry

`CLKComplicationDataSource` uses `CLKComplicationTimelineEntry` to specify timeline entries

```
convenience init(date date: NSDate,
complicationTemplate complicationTemplate: CLKComplicationTemplate)
```

`date`

The date to start showing the timeline entry

`complicationTemplate`

The template with the data to display

Getting the Timeline Entries

Getting current entry

```
func getCurrentTimelineEntryForComplication(_ complication: CLKComplication,
                                            withHandler handler: (CLKComplicationTimelineEntry?) -> Void)
```

Getting past entries

```
optional func getTimelineEntriesForComplication(_ complication: CLKComplication,
                                              beforeDate date: NSDate,
                                                   limit limit: Int,
                                             withHandler handler: ([CLKComplicationTimelineEntry]?) -> Void)
```

Getting future entries

```
optional func getTimelineEntriesForComplication(_ complication: CLKComplication,
                                               afterDate date: NSDate,
                                                   limit limit: Int,
                                             withHandler handler: ([CLKComplicationTimelineEntry]?) -> Void)
```

Scheduling Timeline Updates

To determine when to schedule a timeline update in the future, ClockKit calls:

```
optional func getNextRequestedUpdateDateWithHandler(_ handler: (NSDate?) -> Void)
```

Sometime after that date, ClockKit will call either:

```
optional func requestedUpdateDidBegin()
```

or, in the case of your last chance to update your timeline due to budget constraints:

```
optional func requestedUpdateBudgetExhausted()
```

If the timeline needs updating, you must call `reloadTimelineForComplication:` or `extendTimelineForComplication:` from these methods.

ClockKit will then call `getNextRequestedUpdateDateWithHandler:` and repeat.

Specifying a Placeholder Template

```
func getPlaceholderTemplateForComplication(_ complication: CLKComplication,
                                           withHandler handler: (CLKComplicationTemplate?) -> Void)
```

This template is used in the customization screen of your watch face and shows dummy data for your complication that indicates to the user what sort of data your complication conveys.

Specifying Privacy Behavior

```
optional func getPrivacyBehaviorForComplication(_ complication: CLKComplication,
                                        withHandler handler: (CLKComplicationPrivacyBehavior) -> Void)



enum CLKComplicationPrivacyBehavior : UInt {
    case ShowOnLockScreen
    case HideOnLockScreen
}
```

ClockKit Summary

`CLKComplication`
An opaque object representing a complication; you can query its family

`CLKComplicationTemplate`
The base class of the 22 specific kinds of templates

`CLKTextProvider` / `CLKImageProvider`
Wrappers around text and images that display/shorten them as appropriate for a particular complication template family

`CLKComplicationTimelineEntry`
Maps a date to a complication template

`CLKComplicationServer`
Used to extend and reload complication timelines

`CLKComplicationDataSource`
The data source for timeline entries