iOS Dev Accelerator
Week 7 Day 4

- App Submission
- Class Prefixes
- Typedef
- Enumerations
- Instruments & Exception Breakpoint

App Distribution

Bundle ID

- Your Bundle ID is how both Apple and your device recognizes your app.
- Your app's Bundle Identifier must be unique to be registered with Apple
- Usually written out in reverse DNS notation (ie com.myCompany.myApp)
- The Bundle ID you have set in your App's Xcode project MUST match the Bundle ID you have assigned to the App on the iOS Dev Center.
- In Xcode, the Bundle ID is stored in the Info.plist, and is later copied into your app's bundle when you build.
- In Member center, you create an App ID that matches the app's bundle ID.
- In iTunes Connect, you enter the Bundle ID to identify your app, after your first version is available on the store, you cannot change your bundle ID EVER AGAIN.

Teams!

- Each Xcode project is associated with a single team.
- If you enroll as an individual, you're considered a one-person team.
- The team account is used to store the certificates, identifiers, and profiles needed to provision your app.
- All iOS apps needs to be provisioned to run on a device.

Team Provisioning Profile

- When you set your team, Xcode ‘may’ attempt to create your code signing identity and development provisioning profile.
- Xcode creates a specialized development provisioning profile called a team provisioning profile that it manages for you.
- A team provisioning profile allows an app to be signed and run by all team members on all their devices.

Provisioning Profile Creation

- Here are the steps Xcode takes when creating your provisioning profile:

1. Requests your development certificate
2. Registers the iOS device chosen in the Scheme popup menu
3. Creates an App ID that matches your app's bundle ID and enables services
4. Creates a team provisioning profile that contains these assets
5. Sets your project's code signing build settings accordingly

Version Number & Build String

- The version number of an app is 3 positive integers separated by periods (ex: 1.0.4)
- The first digit is a major release, the 2nd is a minor release, and the third is a maintenance release.
- Build String represents an iteration of the bundle and contain letters and numbers. Change it whenever you distribute a new build of your app for testing.

Code Signing

- “Code Signing your app lets users trust your app has been created by a source known to Apple and that it hasn't been tampered with”
- The **signing identity** is a public-private key pair that Apple issues. The private key is stored in your keychain and used to generate a signature. The certificate contains the public key and identifies you as the owner of the key pair.
- To sign an app, you also need an intermediate certificate, which is automatically installed in your keychain when you install Xcode.
- You use Xcode to create your signing identity and sign your app. Your signing identity is added to your keychain after creation and the corresponding certificate is stored in the member center.
- A **development certificate** identifies you, as a team member, in a development provisioning profile that allows your apps signed by you to launch on devices.
- A **distribution certificate** identifies your team or organization in a distribution provisioning profile and allows you to submit your app to the store.
- You can view your Signing Identifies and Provisioning Profiles in Xcode if you need to troubleshoot them! Everything should match what you see in Member Center.

Submitting your app

1. Create a distribution certificate for your app in Xcode.
2. Create a store distribution provisioning profile on Member Center.
3. Archive and Validate your app in Xcode.
4. Create your App on iTunes Connect
5. Submit your app binary using Xcode or application Loader.
6. Finalize all the info on iTunes Connect and add the build you submitted from Xcode

Class Prefixes

Class Naming

- In Objective-C, class names must be unique not only across your own code, but also any framework/libaries and bundles you might be including in your project.
- This is because Objective-C does not support name-spacing (Swift does).
- So you should avoid using generic class names, like MenuCell and DetailViewController, since another framework or bundle you include might have the same class name.
- So in order to best ensure this uniqueness, the convention is to use prefixes on all your classes.

Class Prefixes

- You can already see prefixes in play with apple provided classes that start with UI or NS.
- Those two letter prefixes are reserved for Apple. Other Apple prefixes: AB, CA,CI, etc
- Your own class prefixes should be 3 letters, to avoid overlapping with Apples.
- Your prefixes are usually an abbreviation of your company name or your full name. (ex BPJ or CFS —CodeFellowsStackoverflow)

Class Prefixes Note

- Prior to Xcode 6, Xcode would ask you what you want your class prefix should be for each project, and would insert the letters into the beginning of all of your class names.
- Xcode 6 no longer does this.
- Someone filed a radar about this, and Apple responded by saying they no longer recommend class prefixes unless you are specifically creating a framework. Your regular app classes don't need them anymore.
- Regardless, its good to know about this concept.

Demo

Enums

Enums

- An enum, or enumerator, is just a list of predefined variables.
- Enums are a data type, just like ints and doubles.
- An enum variable can only hold values defined in the enum.
- To define an enum, you use the keyword enum
- Values in an enum are ints under the hood.

Enums

- Defining an enum is as simple as using the enum keyword, and then defining a list of comma separated options
- to create a variable with your enum type, also use the enum keyword
- same idea if you want to have it as a parameter

```
enum genre {
  comedy,
  romance,
  action,
  sci_fi
};
```

```
enum genre favoriteGenre = sci_fi;
```

```
-(void)findMoviesForGenre:(enum genre)genre {
  //find movies
}
```

Enums

- Enums default by starting at the value 0, but you can give each option a custom number.
- If you leave the option after a custom number without a custom number, it just takes the next number
- So sci_fi here is 13

```
enum genre {
    comedy,
    romance = 11,
    action = 12,
    sci_fi
};
```

Demo

Typedef

Typedefs

- Objective-C (and also Swift) provides a keyword called typedef, which lets you give types a new name.
- It can be something as simple as giving NSString a new nickname:

```
typedef NSString word;
```

- Or as complicated as a block:

```
typedef void (^downloadCallback)(NSArray *, NSError *);
```

refer to fuckingblocksyntax.com when you need to do this

Typedefs and enums

- Typedefs are great to use with enums.
- You can declare an anonymous enum, and then give that enum a typedef
- this way when you declare variables or want to use the enum as a parameter or return type, you dont have to use the keyword enum:

```
typedef enum {
  apple,
  samsung,
  HTC,
  Nokia
} brand;
```

Demo

Instruments

- “Instruments is a performance, analysis, and testing tool for dynamically tracing and profiling OS X and iOS Code”
- Instruments allows to you to view data from largely different aspects of your app and view them side by side, helping you identify trends and bottlenecks.
- Instruments is a set of modules, each one a template that collects and displays different types of information, such as file IO, memory usage, CPU, etc.

Trace Document

- All the data and work done in instruments is done in a trace document.
- The trace documents displays the set of instruments you are using and the data they have collected.
- Each document will typically only hold a single session worth of data and is considered a ‘single trace’. They can be saved for future viewing.
- There are a few instruments that can even simulate a user interface interactions.

Trace Document

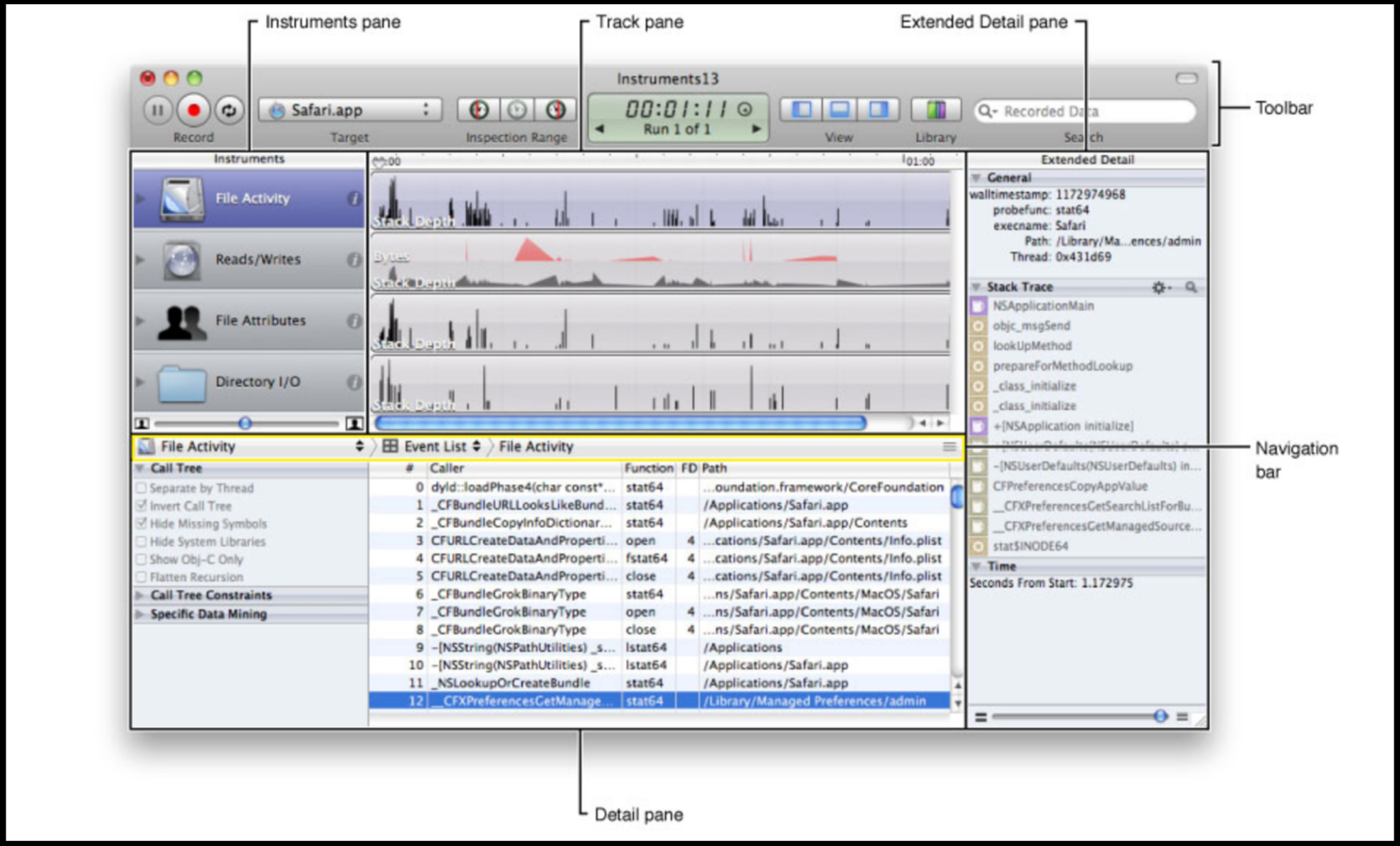

Table 2-3 Trace document toolbar controls

Control	Description
Pause/Resume button	Pauses the gathering of data during a recording. This button does not actually stop recording, it simply stops Instruments from gathering data while a recording is under way. In the track pane, pauses show up as a gap in the trace data.
Record/Stop button	Starts and stops the recording process. Use this button to begin gathering trace data.
Loop button	Sets whether the user interface recorder should loop during play back to repeat the recorded steps continuously. Use this setting to gather multiple runs of a given set of steps.
Target menu	Selects the target for the document. The target is the process (or processes) for which data is gathered.
Inspection Range control	Selects a time range in the track pane. When set, Instruments displays only data collected within the specified time period. Use the buttons of this control to set the start and end points of the inspection range and to clear the inspection range.
Time/Run control	Shows the elapsed time of the current trace. If your trace document has multiple data runs associated with it, you can use the arrow controls to choose the run whose data you want to display in the track pane.
View control	Hides or shows the Instruments pane, Detail pane, and Extended Detail pane. This control makes it easier to focus on the area of interest.
Library button	Hides or shows the instrument library. For information on using the Library window, see "Adding and Configuring Instruments."
Search field	Filters information in the Detail pane based on a search term that you provide. Use the search field's menu to select search options.

Allocations

- The allocations instrument captures and displaying information about memory allocations in your app.
- Allocations can help you recover memory you have abandoned by using its Mark generation feature.
- Demo

Leaks

- The Leaks instrument measures your general memory usage, and marks any leaked memory.
- If a leak is an object, it will be reported by its Class name. If its not an object, it will just say Malloc-size.
- Demo

Zombies

- The Zombies instrument will help you detect overreleased objects, aka zombies aka dangling pointers.
- Zombies works by replacing any object that has its retain count set to 0 with an NSZombie object, and then detects whenever these zombie objects are executed on.
- Demo

Time Profiler

- Time profilers help you provide a great user experience by helping you improve launch times, FPS, and overall smoothness in your UI.
- It’s goal is to tell you how much time is spent in each method.
- Demo

Exception Breakpoint

- Sometimes you will get an exception and the callstack/Xcode will be very unhelpful with telling you where or what happened.
- You can create an exception breakpoint, which basically put a breakpoint on the line that this mysterious exception was raised.
- Demo