Instruments

BJ Homer

Day One

Instruments

Instruments

Instruments lets you examine a running application

Instruments lets you examine a running application

... without stopping it.

Things you can analyze:

Memory allocation

CPU performance

Drawing performance

Things you can analyze:

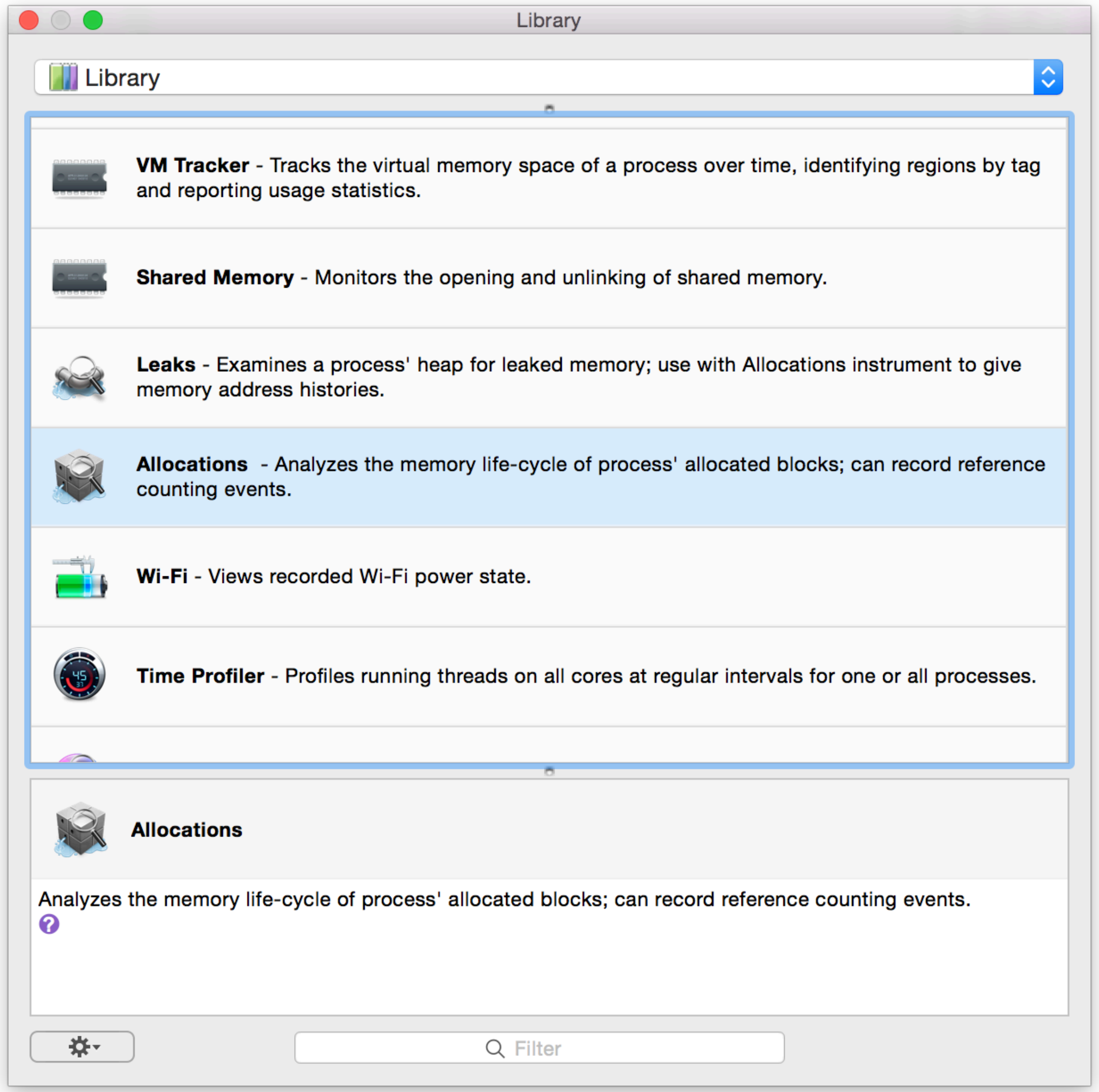
Thread behavior

Energy analysis

Drawing performance

Let's dive in

Instruments



VM Tracker - Tracks the virtual memory space of a process over time, identifying regions by tag and reporting usage statistics.

Shared Memory - Monitors the opening and unlinking of shared memory.

Leaks - Examines a process' heap for leaked memory; use with Allocations instrument to give memory address histories.

Allocations - Analyzes the memory life-cycle of process' allocated blocks; can record reference counting events.

Wi-Fi - Views recorded Wi-Fi power state.

Time Profiler - Profiles running threads on all cores at regular intervals for one or all processes.

Allocations

Analyzes the memory life-cycle of process' allocated blocks; can record reference counting events.

Instruments

Allocations—Memory events

Leaks—Leaked memory allocation history

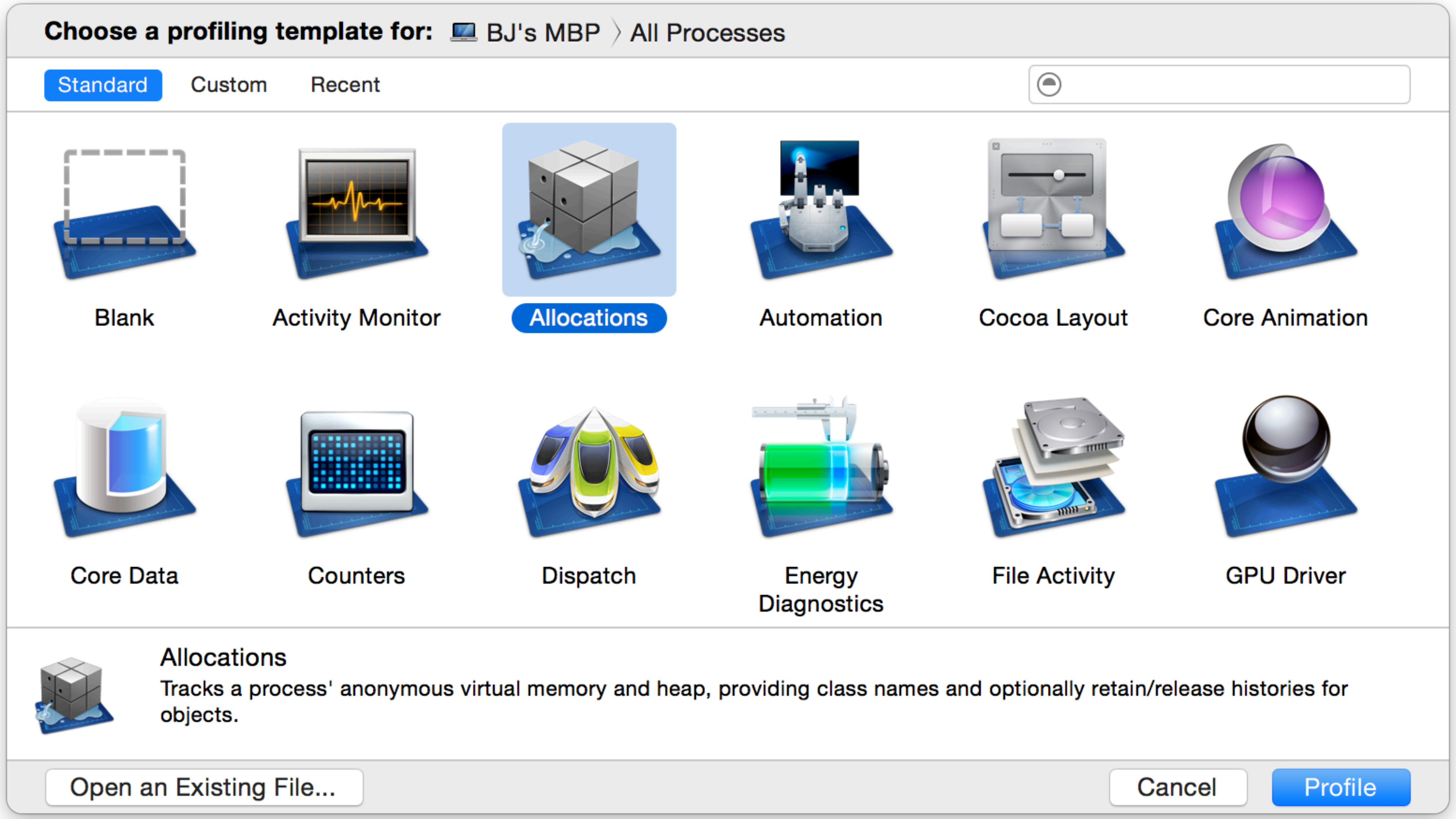
Time Profiler—High-frequency stack traces

Core Animation—Frames/sec, debugging

Core Data—Information on fetch/save/fault times

+ 40 more

Templates



Allocations

Tracks a process' anonymous virtual memory and heap, providing class names and optionally retain/release histories for objects.

Templates

Pre-configured sets of instruments.

Common templates:

- Allocations
- Time Profiler
- Leaks
- Zombies

The best way to learn Instruments

is to use it

DTrace

Super awesome runtime analysis

DTrace

The foundation of Instruments

Zero runtime cost

when disabled

DTrace Probes

provider : module : function : name

DTrace Probes

```
syscall::open:entry
syscall::open:return
```

DTrace Probes

```
objc*:MySpecialView:-drawRect?:entry

pid*:MyAppName:*MySwiftClass*someFunc*:entry
```

What's going on here?

```
objc*            - Provider
MySpecialView    - Module
drawRect?        - Function
entry            - Name
```

What's going on here?

```
pid*                             - Provider
MyAppName                        - Module
MySwiftClass*someFunc*           - Function
entry                            - Name
```

Custom probes

Trace your own data!

Custom probes

```
// MyProbes.d
provider CocoaHeads {
    probe attendeeCount(int);
};
```

Custom Probes

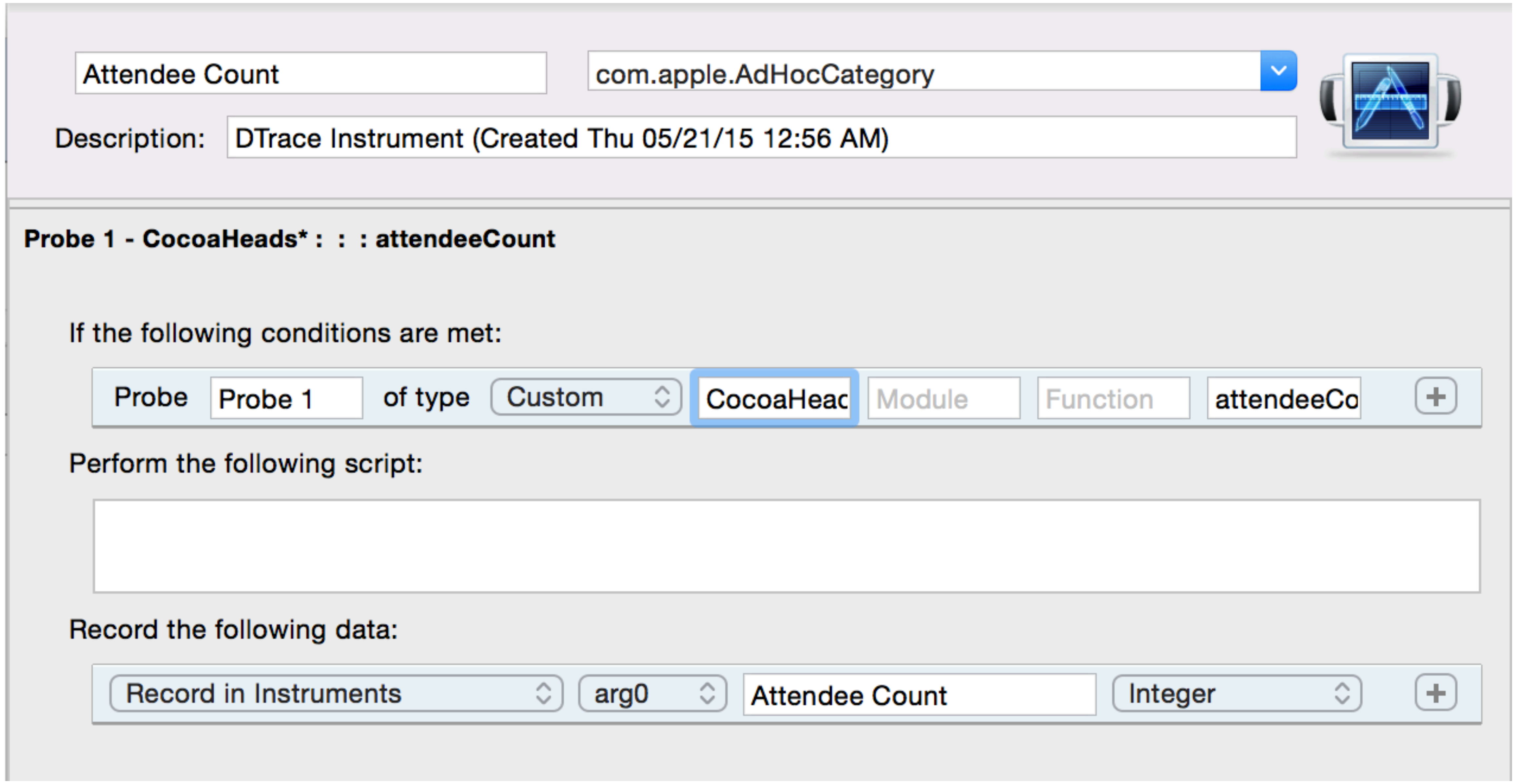
```
// MyProbes.h  (Generated)
#define COCOAHEADS_ATTENDEECOUNT(count) ...
```

Custom Probes

```
#import "MyProbes.h"

void traceAttendeeCount(uint32_t count) {
    COCOAHEADS_ATTENDEECOUNT(count);
}
```

Custom Probes


Attendee Count

com.apple.AdHocCategory

Description: DTrace Instrument (Created Thu 05/21/15 12:56 AM)

Probe 1 - CocoaHeads* : : : attendeeCount

If the following conditions are met:

Probe | Probe 1 | of type | Custom | CocoaHeac | Module | Function | attendeeCo

Perform the following script:

Record the following data:

Record in Instruments | arg0 | Attendee Count | Integer

Custom Probes

In Action

Learn more

objc.io/issue-19/dtrace.html

These slides are online at

bjhomer.github.io

Know your tools

Learn Instruments!