Automating debugDescription

NSObject Protocol

/usr/include/objc/NSObject.h

@property(readonly, copy) NSString *debugDescription

Returns a string that describes the contents of the receiver for presentation in the debugger.

The debugger's print-object (`po`) command invokes this method to produce a textual description of an object.

`NSObject` implements this method by calling through to the `description` method. Thus, by default, an object's debug description is the same as its description. However, you can override `debugDescription` if you want to decouple these.

```
@interface AutoDescription : NSObject


@property (nonatomic, strong) NSString *name;

@property (nonatomic, readwrite) float distance;

@property (nonatomic, readwrite) NSInteger itemCount;

@property (nonatomic, strong) NSString *level;

@property (nonatomic, readwrite) struct Position pos;

@property (nonatomic, strong) NSDictionary *dict;


@end
```

```
(lldb) po foo
<AutoDescription: 0x10010aa60>
```

```
(lldb) po foo
<AutoDescription: 0x10010aa40>
name: Sam
distance: 123.456001
itemCount: 8
level: 5
pos: struct Position {(float) = 1295.000000, (float) = 95.099998, (float) = 0.130000}
dict: {
    a = 1;
    b = 2;
    c = 3;
    d = 4;
    e = 5;
}
```

- Uses Objective-C Runtime to iterate over the properties on a class

- From each property it resolves the Ivar, this is to avoid calling the getter directly (no way to know if a getter will have side effects) and from that get the offset of the Ivar on the object instance

- Also use the property to get the type encoding

- Calculate the size of the Ivar based on the encoded type, use the type to provide the print formatter

- Composite the gathered information into return value

Benefits

- Uses custom Objective-C type decoder (can correctly decode structs)
- Doesn’t use any private APIs (safe/stable to ship)
- Never forget to update classes again

Demo

github.com/samdmarshall/debugDescription