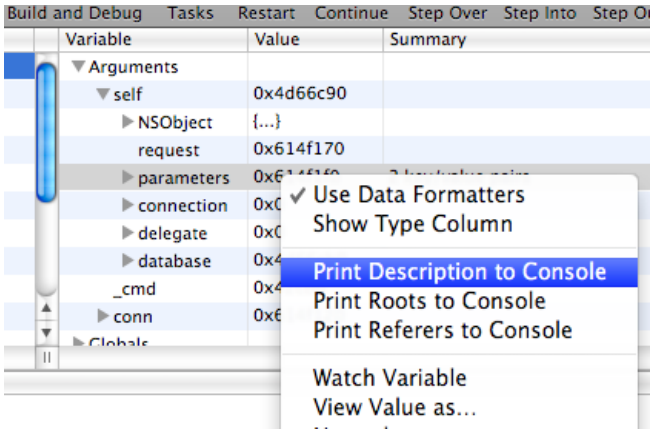
Xcode的Debug中查看数据细节的方法

Xcode的Debug中查看数据细节的方法

热度 3已有 1946 次阅读 2012-9-25 11:23 |个人分类:调试技巧|系统分类:ios| xcode, 查看数据细节

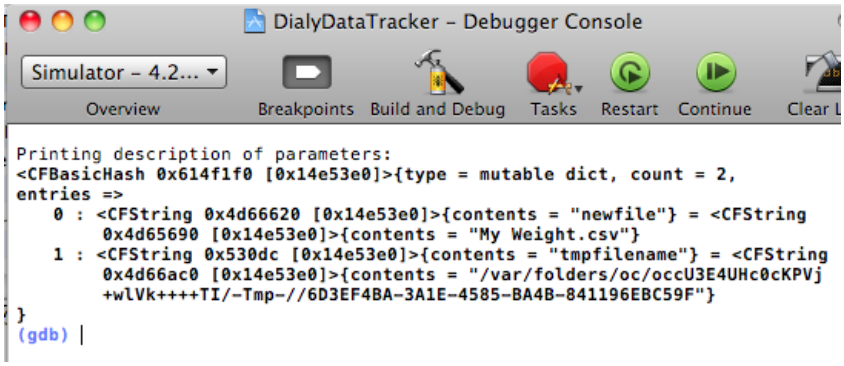
众所周知，在Xcode中的IDE环境中编译和调试程序十分方便，我们在某行代码中设置好断点，当程序执行到该处时，只需要将鼠标放到代码段中的字符串等变量名上面，Xcode就能显示出变量的内容。但如果是一些稍微复杂的变量类型，诸如NSDictionary，还是无法看到字典里的全部字段的内容。如果用NSLog去自己写代码输出的话，又嫌有些麻烦。Google了一下，在伟大的StackOverFlow网站找到一个好方法。

这个时候，我们可以开启Debuger，来追踪到具体的内容。在代码的断点处使用快捷键command+shift+Y，调出Debuger窗口，在左侧选择你的代码文件，右侧会列出该代码段的变量列表，选择你所需要追踪的数据，这里我找到了要查看的变量parameters，这是一个NSDictionary类型。点击右键，选择Print Description to Console。如下图所示：



找到需要查看的变量名

然后打开Xcode的控制台(command+shift+R)，就能看到详细的信息了。很简单吧，高手可以无视了，对于我这个Xcode菜鸟来说，真的很方便了。



数据显示在控制台了

Xcode的调试器为用户提供了一个GDB的界面，GDB是GNU组织的开放源代码调试器。您可以在Xcode的图形界面里做任何事情；但是，如果您需要您可以在命令行里使用GDB的命令。 要在一个调试的任务里输入GDB命令行命令：在工具栏里点击Console Drawer （控制台抽屉窗口）按钮打开控制台。 您可以在控制台里查看Xcode调试器发送给GDB的命令，或者您可以直接在控制台里输入GDB命令。在控制台窗口里点击然后在gdb提示符后面输入命令。

Java代码

1. 命令	解释
2. break NUM	**在指定的行上设置断点。**
3. bt	显示所有的调用栈帧。该命令可用来显示函数的调用顺序。
4. clear	删除设置在特定源文件、特定行上的断点。其用法为：clear FILENAME:NUM。
5. continue	**继续执行正在调试的程序。该命令用在程序由于处理信号或断点而**
6.	导致停止运行时。
7. display EXPR	每次程序停止后显示表达式的值。表达式由程序定义的变量组成。
8. file FILE	装载指定的可执行文件进行调试。
9. help NAME	显示指定命令的帮助信息。
10. info **break**	显示当前断点清单，包括到达断点处的次数等。

```
11. info files               显示被调试文件的详细信息。
12. info func                显示所有的函数名称。
13. info local               显示当函数中的局部变量信息。
14. info prog                显示被调试程序的执行状态。
15. info var                 显示所有的全局和静态变量名称。
16. kill                     终止正被调试的程序。
17. list                     显示源代码段。
18. make                     在不退出 gdb 的情况下运行 make 工具。
19. next                     在不单步执行进入其他函数的情况下，向前执行一行源代码。
20. print EXPR               显示表达式 EXPR 的值。
21.
22. print-object            打印一个对象
23. print (int) name       打印一个类型
24. print-object [artist description]    调用一个函数
25. set artist = @"test"     设置变量值
26. whatis                       查看变理的数据类型
```

在Xcode中，Debug时，不能像eclipse ,或VS那些集成开发那样，能直接查看变量的值。那怎么在调试的时候查看XCode的变量呢？有一些方法的。

1、新建一个Single View App

在viewDidLoad里添加些代码：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    NSDictionary*dic = [NSDictionary dictionaryWithObjectsAndKeys:@"value1",@"key1",
@"28",@"age",@"rongfzh",@"name",nil];
    UILabel*label =[[UILabel alloc] init];
    label.frame= CGRectMake(20,40,250,60);
    label.text= [dic objectForKey:@"name"];
    [self.view addSubview:label];
}
```

在最后一行打上断点。

2、"po" : print object 命令 打印出对象

Command+R调试运行,在 Debug Console 上lldb上输入

```
po dic
```

回车,就可以把字典内容打印出来，

再打印label试试。

```
(lldb)po label
  (UILabel *) $3 = 0x06a8bdd0 <UILabel: 0x6a8bdd0; frame = (20 40; 250 60); text = 'rongfzh';
clipsToBounds = YES; userInteractionEnabled = NO; layer = <CALayer: 0x6a8be90>>
```

label的信息也打印出来了。

3、print命令

```
print (char*)[[dic description] cString]
 (char *) $4 = 0x06d79760 "{\n age = 28;\n key1 = value1;\n name = rongfzh;\n}"
```

打印对象的retainCount，但对象被回收

```
(lldb) print (int)[label retainCount]
(int) $2 = 1
```

GDB可以很方便的帮我们查看变量的值。

当执行到某断点时，在GDB窗口中使用po就可以查看变量.(po = print object)<wbr></wbr>

1）查看String 或其它变量。

```
po 变量名
```

2）查看某个Property。比如要查看item变量的name属性。

po [item name] <wbr><wbr>注意，po item.name是不工作的。</wbr></wbr>

3）查看数组

po [myArray objectAtIndex:index]

如果需查看内存数据：可以在输出窗口采用gdb命令：x /nfu <addr>

n表示要显示的内存单元的个数

－－

f表示显示方式，可取如下值：

x 按十六进制格式显示变量

d 按十进制格式显示变量

u 按十进制格式显示无符号整型

o 按八进制格式显示变量

t 按二进制格式显示变量

a 按十六进制格式显示变量

i 指令地址格式

c 按字符格式显示变量

f 按浮点数格式显示变量

－－

u表示一个地址单元的长度：

b表示单字节

h表示双字节

w表示四字节

g表示八字节

－－－

例如x/16xb self

会显示self指针地址内容，16个字节，16进制