



# 网络项目 HTTP 协议处理

---

千锋iOS 欧阳坚



# 通过本课程，你将掌握

HTTP协议	
HTTP协议服务器	
HTTP协议GET	
HTTP协议POST	
HTTP协议CGI服务器编写	



# HTTP协议原理

---

## HTTP协议原理

## <CR>表示 \r\n

GET /simple.htm HTTP/1.1<CR>

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,  
application/x-shockwave-flash, application/vnd.ms-excel, application/  
vnd.ms-powerpoint, application/msword, \*/\*<CR>

Accept-Language: zh-cn<CR>

Accept-Encoding: gzip, deflate<CR>

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;  
SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)<CR>

Host: localhost:8080<CR>

Connection: Keep-Alive<CR>

<CR>



# HTTP典型响应头和内容

HTTP/1.1 200 OK<CR>

Server: Microsoft-IIS/5.1<CR>

X-Powered-By: ASP.NET<CR>

Date: Fri, 03 Mar 2006 06:34:03 GMT<CR>

Content-Type: text/html<CR>

Accept-Ranges: bytes<CR>

Last-Modified: Fri, 03 Mar 2006 06:33:18 GMT<CR>

ETag: "5ca4f75b8c3ec61:9ee"<CR>

Content-Length: 37<CR>

<CR>

<html><body>hello world</body></html>



# 发送给服务器

- 发送给服务器有2种方式 GET和POST
- 而POST又分为2种方式:
  - application/x-www-form-urlencoded
  - multipart/form-data
- 比如客服端发送给服务器端参数如下:  
**myUserName=123&myPassWord=456&myMessage=89**
- 这里有向服务器提交了3个参数分别以&隔开



## GET方式

- GET方式只发送HTTP消息头，没有消息体，也就是除了要GET的基本信息之外不向服务器提供其他信息，网页表单（FROM）的默认提交方式就是用GET方式，它会把所有向服务器提交的信息都作为URL后面的参数，如a.asp?a=1&b=2这样的方式。而当要提交的数据量很大，或者所提交内容不希望别人直接看到时，GET方式就会存在很大的不足，应该使用POST方式。
- GET方式会存放在网页URL链接上(下列是一行)
- `http://localhost/cgi-bin/get.cgi?  
myUserName=123&myPassWord=456&myMessage=89`



# POST方式

- POST方式提交的数据是作为HTTP消息体存在的
- 根据POST方式不一样又分为2种方式:
- application/x-www-form-urlencoded
- 这种方式只是把  
**myUserName=123&myPassWord=456&myMessage=89**字符串作为HTTP消息体存在，和GET方式类型
- multipart/form-data
- 这种格式最为复杂，如果要上传文件必须使用这种方式





## POST2中方式理论知识

- 关于application/x-www-form-urlencoded等字符编码的解释说明
- 在Form元素的语法中，EncType表明提交数据的格式 用Enctype 属性指定将数据回发到服务器时浏览器使用的编码类型。**application/x-www-form-urlencoded**: 窗体数据被编码为名称/值对。这是标准的编码格式。**multipart/form-data**: 窗体数据被编码为一条消息，页上的每个控件对应消息中的一个部分。**text/plain**: 窗体数据以纯文本形式进行编码，其中不含任何控件或格式字符。
- form的enctype属性为编码方式，常用有两种：**application/x-www-form-urlencoded**和**multipart/form-data**，默认为**application/x-www-form-urlencoded**。当action为get时候，浏览器用x-www-form-urlencoded的编码方式把form数据转换成一个字串（**name1=value1&name2=value2...**），然后把这个字串**append**到url后面，用?分割，加载这个新的url。 当action为post时候，浏览器把form数据封装到http body中，然后发送到server。 如果没有**type=file**的控件，用默认的**application/x-www-form-urlencoded**就可以了。但是如果有**type=file**的话，就要用到**multipart/form-data**了。浏览器会把整个表单以控件为单位分割，并为每个部分加上**Content-Disposition(form-data或者file)**,**Content-Type**(默认为**text/plain**),**name**(控件name)等信息，并加上分割符(**boundary**)。



# multipart/form-data参数格式

```
-----WebKitFormBoundaryAXt2otnPpXnm6evf\r\n
Content-Disposition: form-data; name="myUserName"\r\n
\r\n
12
\r\n
-----WebKitFormBoundaryAXt2otnPpXnm6evf\r\n
Content-Disposition: form-data; name="myPassWord"\r\n
\r\n
333
\r\n
-----WebKitFormBoundaryAXt2otnPpXnm6evf\r\n
Content-Disposition: form-data; name="myMessage"\r\n
\r\n
123
\r\n
-----WebKitFormBoundaryAXt2otnPpXnm6evf--\r\n
```

## 格式解释：

-----  
WebKitFormBoundaryAXt2otnPpXnm6evf

为界限

所有的参数每项以 "--"附加界限  
为开始

参数都是以Content-Disposition:  
开始,name="myUserName"是参  
数名, 在2个空的\r\n之间的内容  
就是参数值

最后以界限以 界限附加一个  
"--"结尾



# HTTP协议

---

服务器响应



# 服务器返回的消息

- 服务器返回的**HTTP**消息也分为消息头和消息体两部分。前面连载的第二篇里已经介绍了返回消息中常见返回代码的含义。对于非正常的返回代码的处理比较简单，只要照着要求去做就好了，而对于正常的返回代码（**200**），其处理方式就多种多样了。

# Content-Type

- **Content-Type**是返回消息中非常重要的内容，它标识出这个返回内容的类型，其值为“主类型/子类型”的格式，例如最常见的就是 **text/html**，它的意思是说返回的内容是文本类型，这个文本又是**HTML**格式的。原则上浏览器会根据**Content-Type**来决定如何显示返回的消息体内容。常见的内容类型有：
  - **text/html** HTML文本
  - **image/jpeg** JPG图片
  - **image/gif** GIF图片
  - **application/xml** XML文档
  - **audio/x-mpegurl** MP3文件列表，如果安装了Winamp，则可以直接把它当面M3U文件来打开

# Content-Disposition

- 如果用AddHeader的方法在HTTP消息头中加入Content-Disposition段，并指定其值为“attachment”，那么无论这个文件是何类型，浏览器都会提示我们下载此文件，因为此时它认为后面的消息体是一个“附件”，不需要由浏览器来处理了。例如，在ASP.Net中 写入如下语句：
- Response.AddHeader("Content-Disposition: attachment");
- 请求此页面是得到的结果如：
- HTTP/1.1 200 OK
- Server: Microsoft-IIS/5.1
- Date: Thu, 23 Mar 2006 07:54:53 GMT
- Content-Disposition: attachment
- Cache-Control: private
- Content-Type: text/html; charset=utf-8
- .....



## 4 Cache

- 返回消息中的**Cache**用于指定网页缓存。我们经常可以看到这样的情况，打开一个网页时速度不快，但再次打开时就会快很多，原因是浏览器已经对此页面进行了缓存，那么在同一浏览器窗口中再次打开此页时不会重新从服务器端获取。网页的缓存是由HTTP消息头中的“**Cache-control**”来控制，常见的取值有**private**、**no-cache**、**max-age**、**must-revalidate**等，默认为**private**。其作用根据不同的重新浏览方式分为以下几种情况：
  - （1） 打开新窗口
    - 如果指定**cache-control**的值为**private**、**no-cache**、**must-revalidate**，那么打开新窗口访问时都会重新访问服务器。而如果指定了**max-age**值，那么在此值内的时间里就不会重新访问服务器，例如：
      - **Cache-control: max-age=5**
        - 表示当访问此网页后的5秒内再次访问不会去服务器
  - （2） 在地址栏回车
    - 如果值为**private**或**must-revalidate**（和网上说的不一样），则只有第一次访问时会访问服务器，以后就不再访问。如果值为**no-cache**，那么每次都会访问。如果值为**max-age**，则在过期之前不会重复访问。
  - （3） 按后退按钮
    - 如果值为**private**、**must-revalidate**、**max-age**，则不会重访问，而如果为**no-cache**，则每次都重复访问
  - （4） 按刷新按钮
    - 无论为何值，都会重复访问



# POST上传例子

千锋3G学院





## 比如浏览器POST上传例子

序列号(可选):

机器信息文件(必填):

Browse...

提交获得认证文件

查看剩余安装个数

重置



# HTML源代码

```
<form method="POST" enctype="multipart/form-data" action="/cgi-bin/rt/  
requestkey.cgi">
```

```
  <p><span lang="zh-cn"><font size="5">序列号</font></span><font size="5">  
(<span lang="zh-cn">可选</span>):</font></p>
```

```
  <p><input type="text" name="userkey" size="48"></p>
```

```
  <p><font size="5"><span lang="zh-cn">机器信息文件</span>(<span lang="zh-  
cn">必填</span>):</font></p>
```

```
  <p><input type="file" name="hwfile" size="47"></p>
```

```
  <p><input type="submit" value="提交获得认证文件" name="getkey">
```

```
  <input type="submit" value="查看剩余安装个数" name="getallkeys">;
```

```
  <input type="reset" value="重置" name="B1"> </p>
```

```
</form>
```



# POST头信息

```
POST /cgi-bin/rt/requestkey.cgi HTTP/1.1\r\n
HOST: localhost\r\n
User-Agent: Mozilla/5.0\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8\r\n
Content-Type: multipart/form-data; boundary=-----123456\r\n
Content-Length: 467\r\n
\r\n
```



# HTTP内容体

-----123456\r\n

Content-Disposition: form-data; name="userkey"\r\n

\r\n

1234

\r\n

-----123456\r\n

Content-Disposition: form-data; name="hwfile"; filename="a.c"\r\n

Content-Type: text/x-csrc\r\n

\r\n

a.C的内容

\r\n



# HTTP内容体(续)

-----123456\r\n

Content-Disposition: form-data; name="getkey"\r\n

\r\n

Getkey按键上的内容

\r\n

-----123456\r\n



# HTTP内容体(图片信息)

Content-Disposition: form-data; name="hwfile";  
filename="common.jpg"\r\n

Content-Type: image/jpeg\r\n  
\r\n

Common.jpg的二进制内容

\r\n

-----123456\r\n



# 下传例子

千锋3G学院



HTTP/1.1 200OK\r\n

Date: Fri, 28 Oct 2011 15:13:42 GMT \r\n

Server: Apache\r\n

Accept-Ranges: bytes\r\n

Content-Length: 86601\r\n

Content-Type: image/jpeg\r\n

\r\n

具体的jpeg图片二进制内容





# Range请求头

- Range:bytes=554554-
- 表示头500个字节: bytes=0-499
- 表示第二个500字节: bytes=500-999
- 表示最后500个字节: bytes=-500
- 表示500字节以后的范围: bytes=500-
- 第一个和最后一个字节: bytes=0-0,-1
- 同时指定几个范围: bytes=500-600,601-999



- Content-Rangs: 0-10000
- Range: bytes=26214400-
- Content-Range: bytes 20000-40000/47000
- 即从第20000字节请求到第40000个字节,(文件长度是47000字节).知道了这一点之后, 请求数据就非常容易了,



# 服务器搭建

---

在mac下搭建CGI服务器

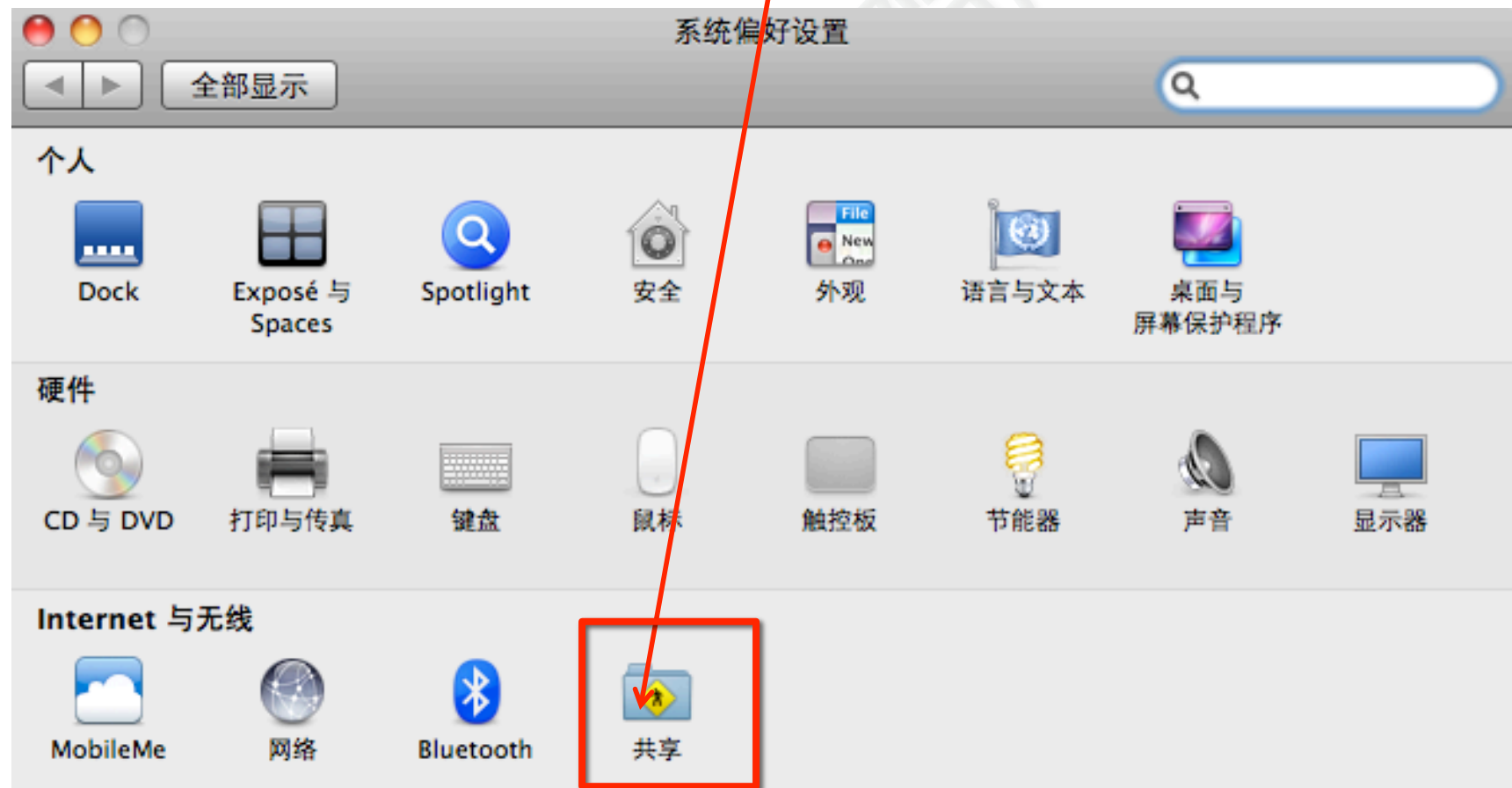


# 写服务器端程序

- iOS程序员不需要写服务器程序
- 但是需要了解服务器都做了什么

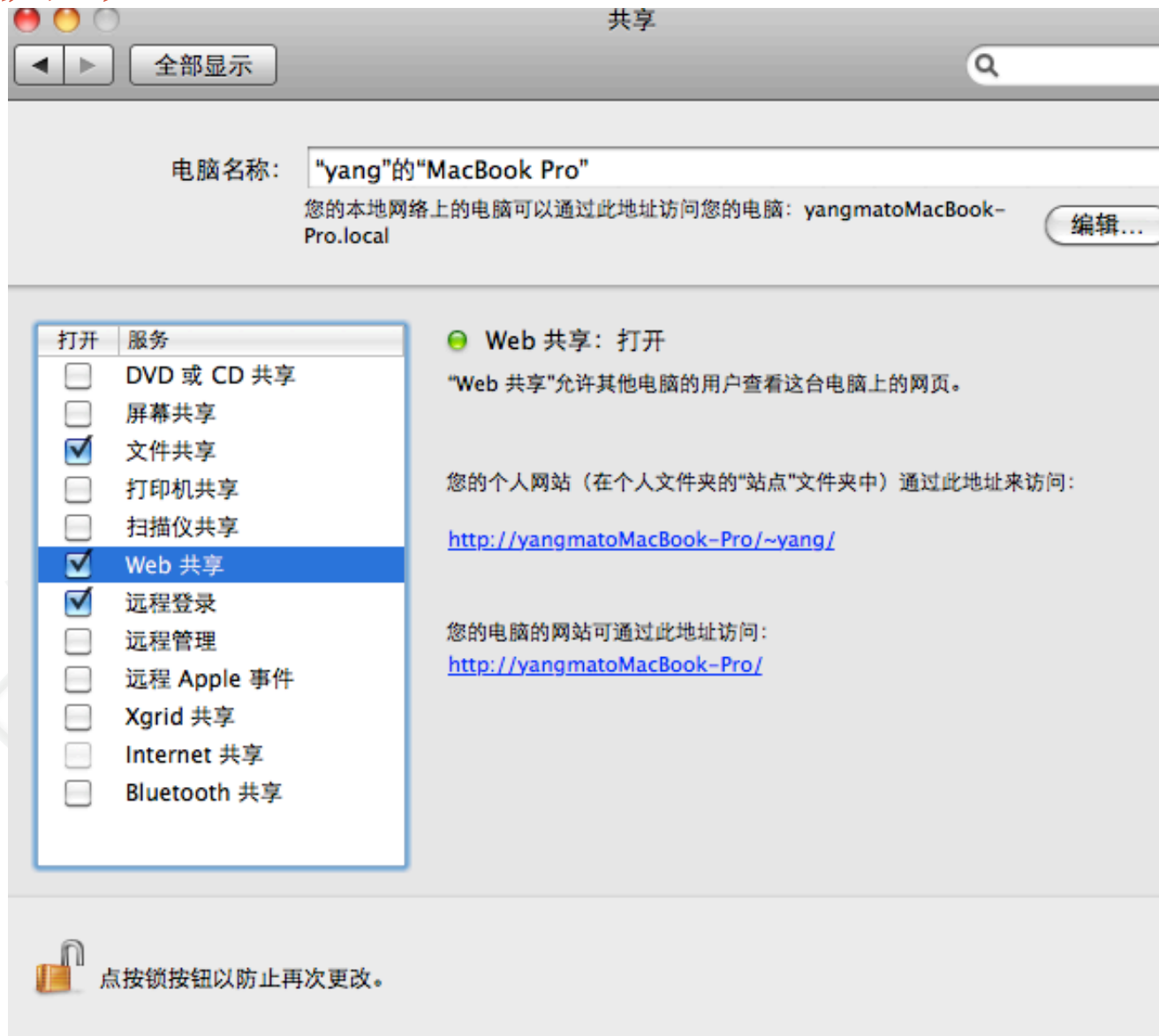


# 开启HTTP服务 点击共享



# 开启HTTP协议

- 确保“Web 共享”选项打开
- 测试方法
- `http://localhost/`
- 看看是否有内容



# 服务器CGI目录介绍

- MacOS的服务器地址是: **/Library/WebServer**
- 目录结构如下:
- CGI执行程序目录: CGI-Executables
- html网页目录: Documents
- 其它文档目录: share



# HTTP GET协议

---

处理直接网页请求





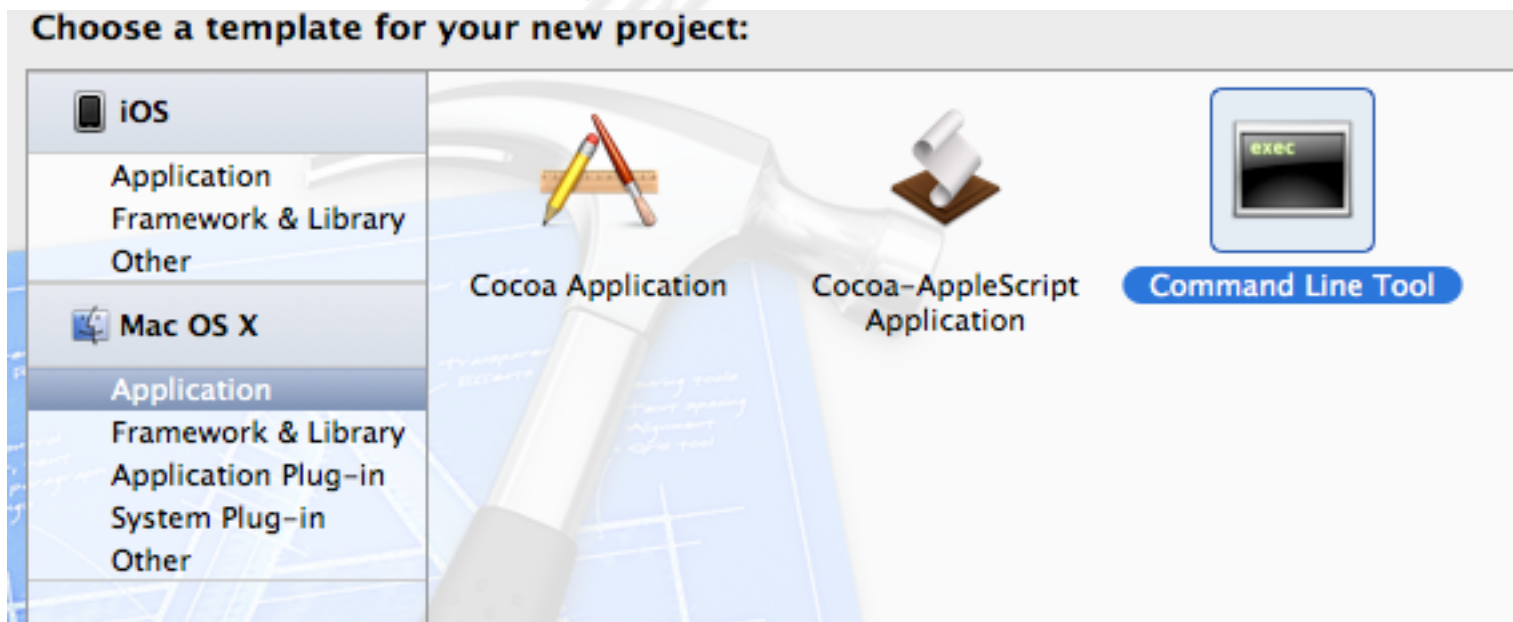
# 开发/cgi-bin/get.cgi程序

- 什么是CGI程序？
- 其实就是C程序的可执行程序，CGI程序可以是Objective-C，C/C++程序，可以是PHP，Bash，Python等程序，这里以C语言为例来开发服务器CGI程序
- 这里程序使用Objective-C进行开发



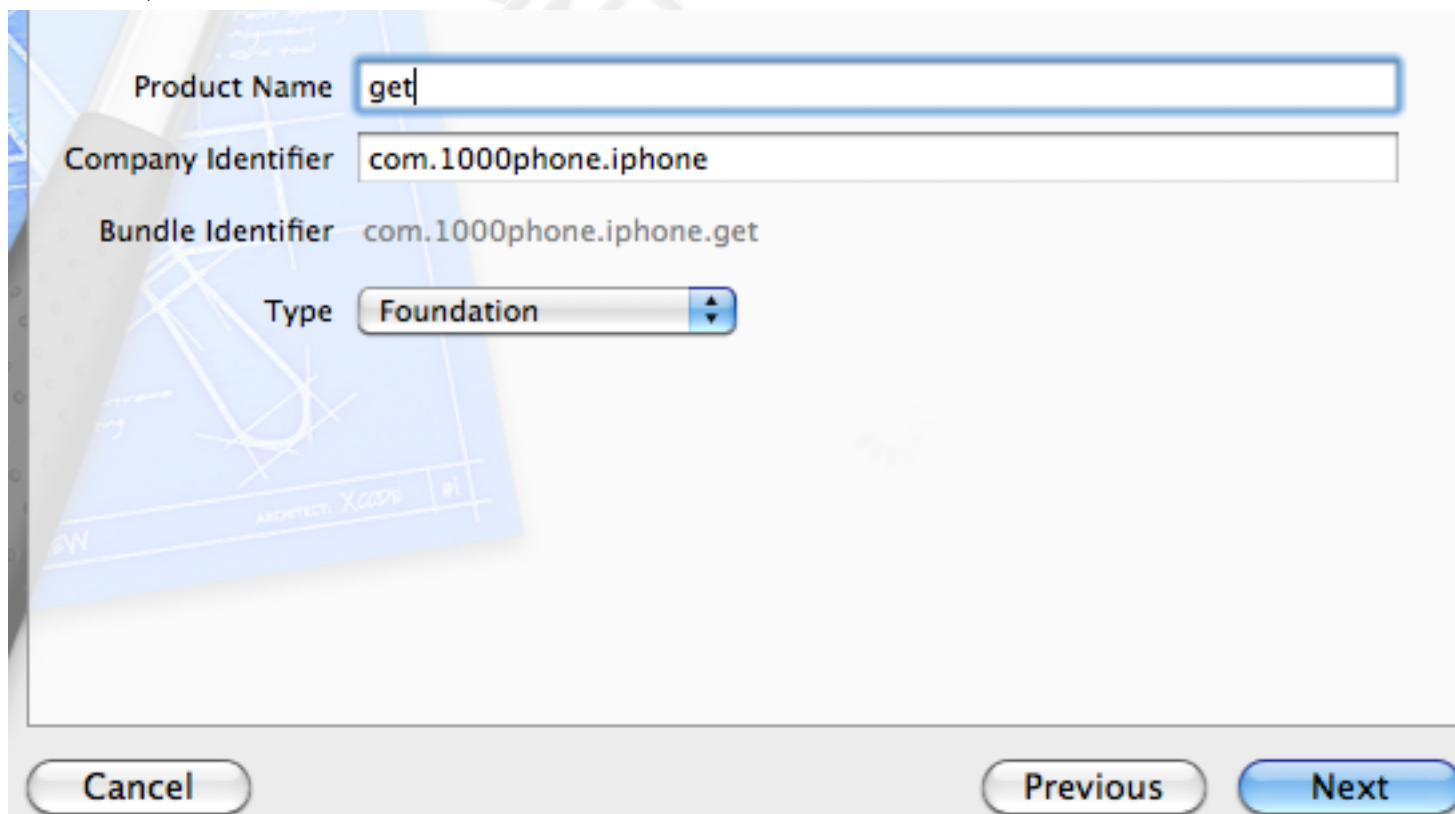
# 创建服务器CGI工程

- 创建Xcode “Command Line Tool”工程
- 该程序是作为CGI程序



# 选择Foundation类型

- 确保选择
- Foundation
- 因为Foundation是
- OC类型



Product Name

Company Identifier

Bundle Identifier

Type



## main.m 文件

```
#import <Foundation/Foundation.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define _START \
{ \
    FILE *dout; \
    dout = fopen("/tmp/ios_debug", "a"); \

#define _END \
    fclose(dout); \
}

void test(void);
void doGet(char *string);
void doCommon(char *string);
```



## main.m 文件(续)

```
void doCommon(char *string) {  
    int i;  
    NSString *requestString = [NSString stringWithUTF8String:string];  
    NSArray *allVars = [requestString componentsSeparatedByString:@"&"];  
    printf("{");  
  
    for (i = 0; i < [allVars count]; i++) {  
        NSString *s = [allVars objectAtIndex:i];  
        NSArray *eq = [s componentsSeparatedByString:@"="];  
        NSString *key = [eq objectAtIndex:0];  
        NSString *value = [eq objectAtIndex:1];  
        printf("\'%s\':\'%s\'",  
            [key UTF8String],  
            [value UTF8String]);  
        if (i != [allVars count]-1) {  
            printf(",");  
        }  
    }  
    printf("}");  
}
```



## main.m 文件(续)

```
void doGet(char *string)
{
    return doCommon(string);
}
```



## main.m 文件(续)

```
void test(void) {  
    char *requestContentType = getenv("CONTENT_TYPE");  
    char *requestMethod = getenv("REQUEST_METHOD");  
    char *scriptName=getenv( "SCRIPT_NAME");  
  
    printf("Content-Type: text/plain\r\n");  
    printf("\r\n");  
  
    _START  
    fprintf(dout, "REQUEST_METHOD: %s\n", requestMethod);  
    fprintf(dout, "CONTENT_TYPE: %s\n", requestContentType);  
    fprintf(dout, "SCRIPT_NAME: %s\n", scriptName);  
    _END
```



## main.m 文件(续)

```
if (strcmp(requestMethod, "GET") == 0) {  
    char *queryString = getenv("QUERY_STRING");  
    _START  
    fprintf(dout, "QUERY_STRING: %s\n", queryString);  
    _END  
  
    doGet(queryString);  
    return;  
}  
}
```





## main.m 文件(续)

```
int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    test();
    [pool release];
    return 0;
}
```



# 编译get.cgi过程

- gcc -o get.cgi main.m -framework Foundation
- clang -o get.cgi main.m -framework Foundation
- 使用gcc或者clang来编译，优先选择clang编译器编译
- 从这里可以看出get其实就是一个可执行程序



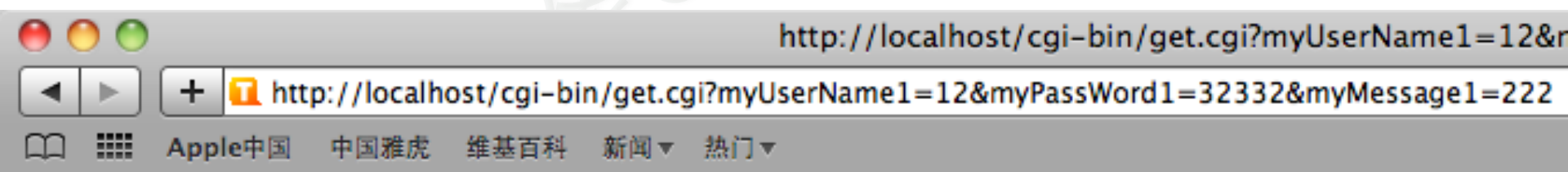
## 部署get.cgi程序

- 将get.cgi程序拷贝到/Library/WebServer/CGI-Executables中即可
- `cp get.cgi /Library/WebServer/CGI-Executables/`



# GET方式测试

- `http://localhost/cgi-bin/get.cgi?`  
`myUserName1=12&myPassWord1=32332&myMessage1`  
`=222`
- 然后查看 `/tmp/ios_debug` 文件，该文件是服务器后台输出文件



OK `myUserName1=12&myPassWord1=32332&myMessage1=222`



## 练习 2人一组相互测试

- `http://localhost/cgi-bin/get.cgi?`  
`myUserName1=12&myPassWord1=32332&myMessage1`  
`=222`
- 两人一组，将localhost换成对方ip地址来进行测试
- 查看ip地址命令
- ifconfig



# HTTP POST 1

---

x-www-form-urlencoded 普通表单方式和GET  
类型，内容要重新编码



# POST方式1 x-www-form-urlencoded

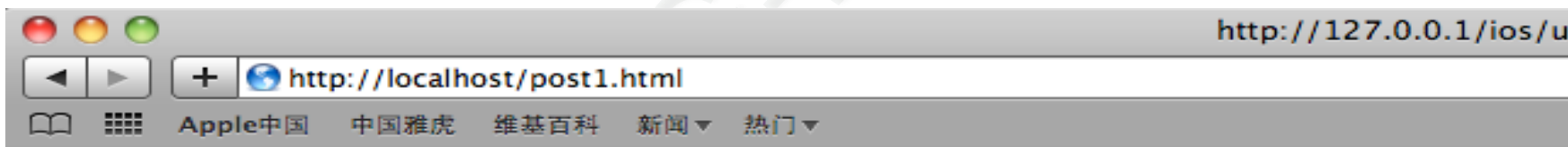
- 该方式

千锋3G学院



# 1. 制作网页

- 下图是一个最后的网页显示结果，右图结果是显示了一个html网页中最常用的一个表单提交方式
- 表单提交方式有2种(下面是第1种)
- **x-www-form-urlencoded**和**multipart/form-data**方式



**you are testing http x-www-form-urlencoded !**

use default enctype=application/x-www-form-urlencoded

username:   
password:   
message:





## 制作上图的网页post1.html

```
<html>
<body>
  <h1>you are testing http x-www-form-urlencoded !</h1>
  <p>use default enctype=application/x-www-form-urlencoded
  <form method=post enctype=application/x-www-form-urlencoded
    action="/cgi-bin/post1.cgi">
    <p>username:
    <input name="myUserName">
    <br>password:
    <input name="myPassWord" type=password>
    <br>message:<textarea name="myMessage"></textarea>
  </p>
  <p>
    <input type="submit" value="Submit">
    <input type="reset" value="Cancel">
  </form>
</body>
</html>
```

use default enctype=application/x-www-form-urlencoded

username:

password:

message:

友情提示:

使用vi存放, 不要使用mac的文本编辑器存放(有问题)

建立一个文件然后存为post1.html页面即可

绿色的部分就是form的执行程序

三个变量分别表示这3个input控件



## 部署post1.html网页

- 讲post1.html网页拷贝在固定路径中即可 /Library/WebServer/Documents
- 友情提示可以使用
- `cp post1.html /Library/WebServer/Documents`



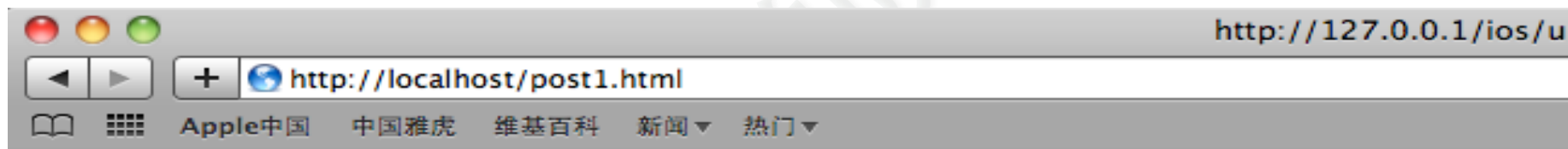
# 测试网页

- 打开任何浏览器，比如Safari，Firefox，IE，iOS/Safari
- 然后输入如下网址
- `http://localhost/post1.html`
- 注意如果在2台机器上测试，把localhost换成相应的ip地址即可



## 运行结果如下

- 如果能显示右图结果
- 表明部署成功



**you are testing http x-www-form-urlencoded !**

use default enctype=application/x-www-form-urlencoded

username:

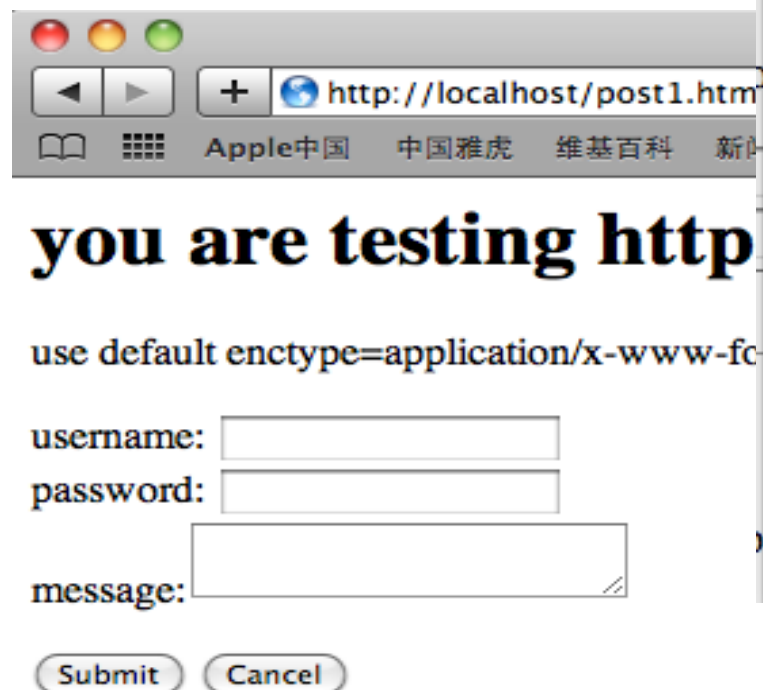
password:

message:



## 网页显示乱码

- 如果出现乱码





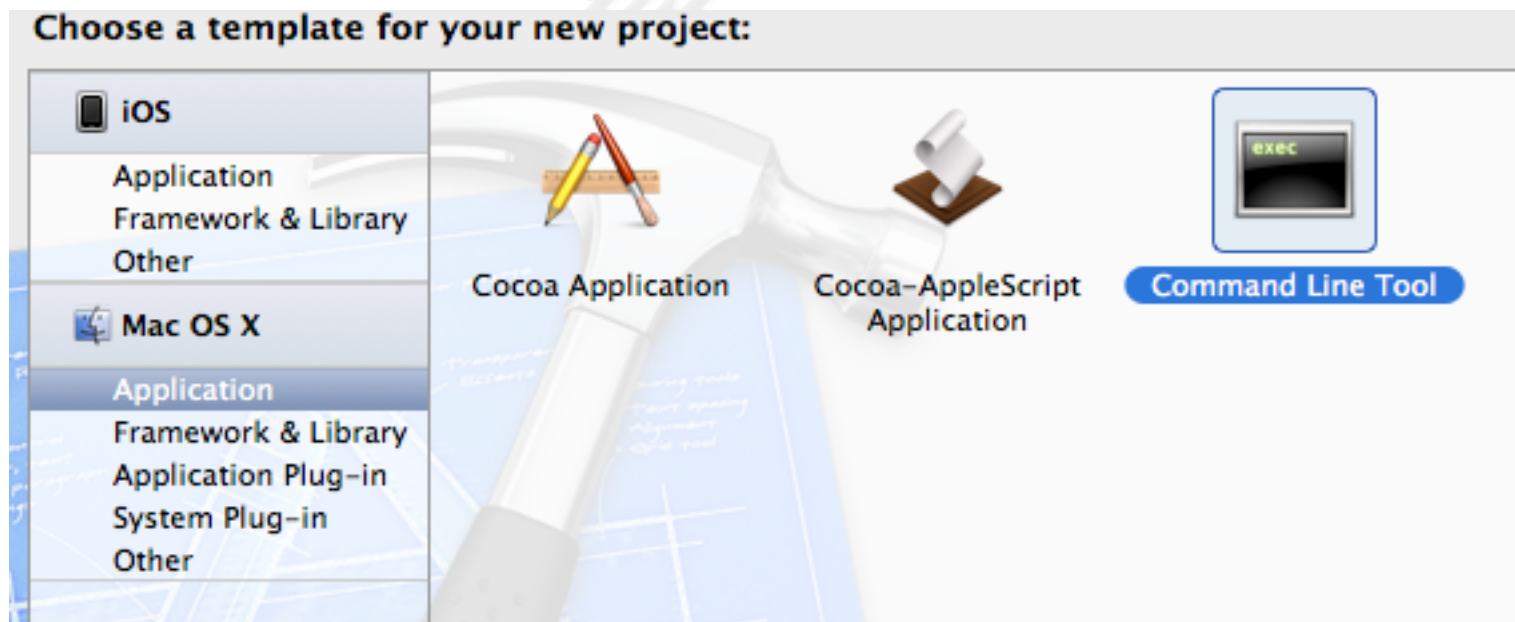
## 2. 编写服务器端程序

千锋3G学院



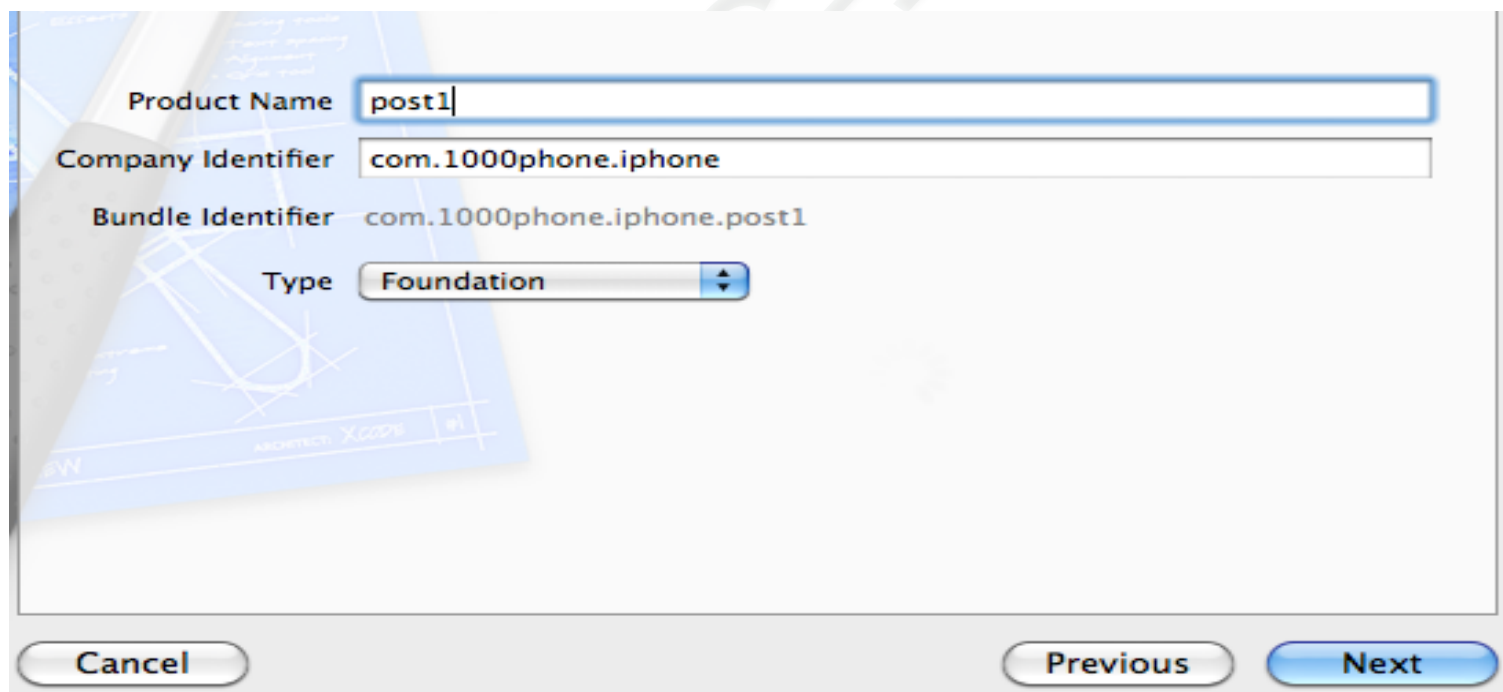
# 创建服务器CGI工程

- 创建Xcode “Command Line Tool”工程
- 该程序是作为CGI程序



# 选择Foundation类型

- 确保选择
- Foundation
- 因为Foundation是
- OC类型







## main.m 文件

```
#import <Foundation/Foundation.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define _START \
{ \
FILE *dout; \
dout = fopen("/tmp/ios_debug", "a"); \

#define _END \
fclose(dout); \
}

void test(void);
void doPostUrlEncoded(char *string);
```



# main.m 文件(续集) doCommon函数

```
void doPostUrlEncoded (char *string) {  
    int i;  
    NSString *requestString = [NSString stringWithUTF8String:string];  
    NSArray *allVars = [requestString componentsSeparatedByString:@"&"];  
    printf("{");  
    for (i = 0; i < [allVars count]; i++) {  
        NSString *s = [allVars objectAtIndex:i];  
        NSArray *eq = [s componentsSeparatedByString:@"="];  
        NSString *key = [eq objectAtIndex:0];  
        NSString *value = [eq objectAtIndex:1];  
        printf("\\""%s\\":\\""%s\\\"",  
            [key UTF8String],  
            [value UTF8String]);  
        if (i != [allVars count]-1) {  
            printf(",");  
        }  
    }  
    printf("}");  
}
```



# main.m 文件(续集) test取得环境变量

```
void test(void) {  
  
    char *requestContentType = getenv("CONTENT_TYPE");  
    char *requestMethod = getenv("REQUEST_METHOD");  
  
    printf("Content-Type: text/plain\r\n");  
    printf("\r\n");  
  
    _START  
    fprintf(dout, "REQUEST_METHOD: %s\n", requestMethod);  
    fprintf(dout, "CONTENT_TYPE: %s\n", requestContentType);  
    _END  
    size_t requestContentLength = atoi(getenv("CONTENT_LENGTH"));  
    char *input = (char *) malloc(requestContentLength+1);  
    memset(input, '\0', requestContentLength+1);  
}
```



## main.m 文件(续集)

```
int haveRead = 0;
while (1) {
    ssize_t ret = read(0, input+haveRead, requestContentLength);
    haveRead += ret;
    if (haveRead == requestContentLength) {
        break;
    }
}
_START
fprintf(dout, "CONTENT_LENGTH: %s\n", getenv("CONTENT_LENGTH"));
fprintf(dout, "CONTENT_TYPE: %s\n", requestContentType);
fprintf(dout, "CONTENT: %s\n", input);
_END
/* POST */
if (strcmp(requestContentType, "application/x-www-form-urlencoded") == 0) {
    doPostUrlEncoded(input);
}
free(input);
}
```



## main.m 主函数

```
int main (int argc, const char * argv[])
{
    @autoreleasepool {
        test();
    }
    return 0;
}
```



## 编译post1.cgi过程

- gcc -o post1.cgi main.m -framework Foundation
- clang -o post1.cgi main.m -framework Foundation
- 使用gcc或者clang来编译，优先选择clang编译器编译
- 从这里可以看出post1.cgi其实就是一个可执行程序

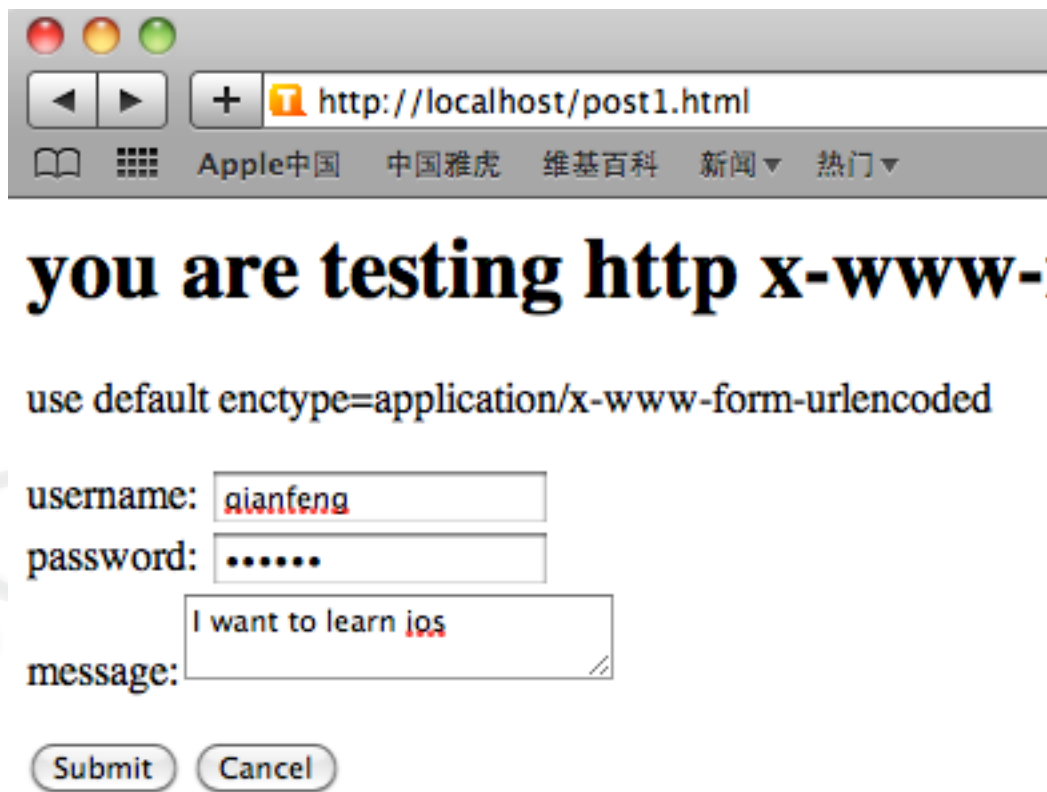


## 部署post1.cgi程序

- 将post1.cgi程序拷贝到/Library/WebServer/CGI-Executables中即可
- `cp post1.cgi /Library/WebServer/CGI-Executables/`

# POST测试

- 在刚才的网页中，输入字符串
- 然后点击“执行”，然后显示结果如下：
- 显示了JSON数据格式
- 同时查看/tmp/ios\_debug文件



you are testing http x-www-

use default enctype=application/x-www-form-urlencoded

username:

password:

message:







# HTTP POST 2

---

multipart/form-data 复杂表单方式(不包括任何文件)



## POST方式2 multipart/form-data 文本提交

- 目前服务器仅仅只是文本提交方式，文件(文本文件,二进制文件)之外的格式



# 1. 制作网页

- 下图是一个最后的网页显示结果，右图结果是显示了一个html网页中最常用的一个表单提交方式
- 表单提交方式有2种(下面是第2种)
- **x-www-form-urlencoded**和**multipart/form-data**方式



**you are accessing multipart/form-data method=post !**

username:

password:

message:



## 制作上图的网页post2.html

```
<html>
<body>
  <h1>you are accessing multipart/form-data method=post !</h1>
  <form enctype="multipart/form-data" method=post
    action="/cgi-bin/post2.cgi">
    <p> username:
    <input name="myUserName" id="user">
    <br> password:
    <input name="myPassWord" type=password id="pasw">
    <br> message:<textarea name="myMessage"></textarea>
  </p>
  <p>
    <input type="submit" value="Submit">
    <input type="reset" value="Cancel">
  </p>
</form>
</body>
</html>
```

友情提示：

使用vi存放，不要使用mac的文本编辑器存放(有问题)

建立一个文件然后存为post2.html页面即可

绿色的部分就是form的执行程序

三个变量分别表示这3个input控件

you are accessing multipart/form-data method=post !

username:

password:

message:



## 部署post2.html网页

- 讲post2.html网页拷贝在固定路径中即可 /Library/WebServer/Documents
- 友情提示可以使用
- `cp post2.html /Library/WebServer/Documents`



# 测试网页

- 打开任何浏览器，比如Safari，Firefox，IE，iOS/Safari
- 然后输入如下网址
- `http://localhost/post2.html`
- 注意如果在2台机器上测试，把localhost换成相应的ip地址即可



## 运行结果如下

- 如果能显示右图结果
- 表明部署成功



**you are accessing multipart/form-data method=post !**

username:

password:

message:



## 2. 编写服务器后端程序

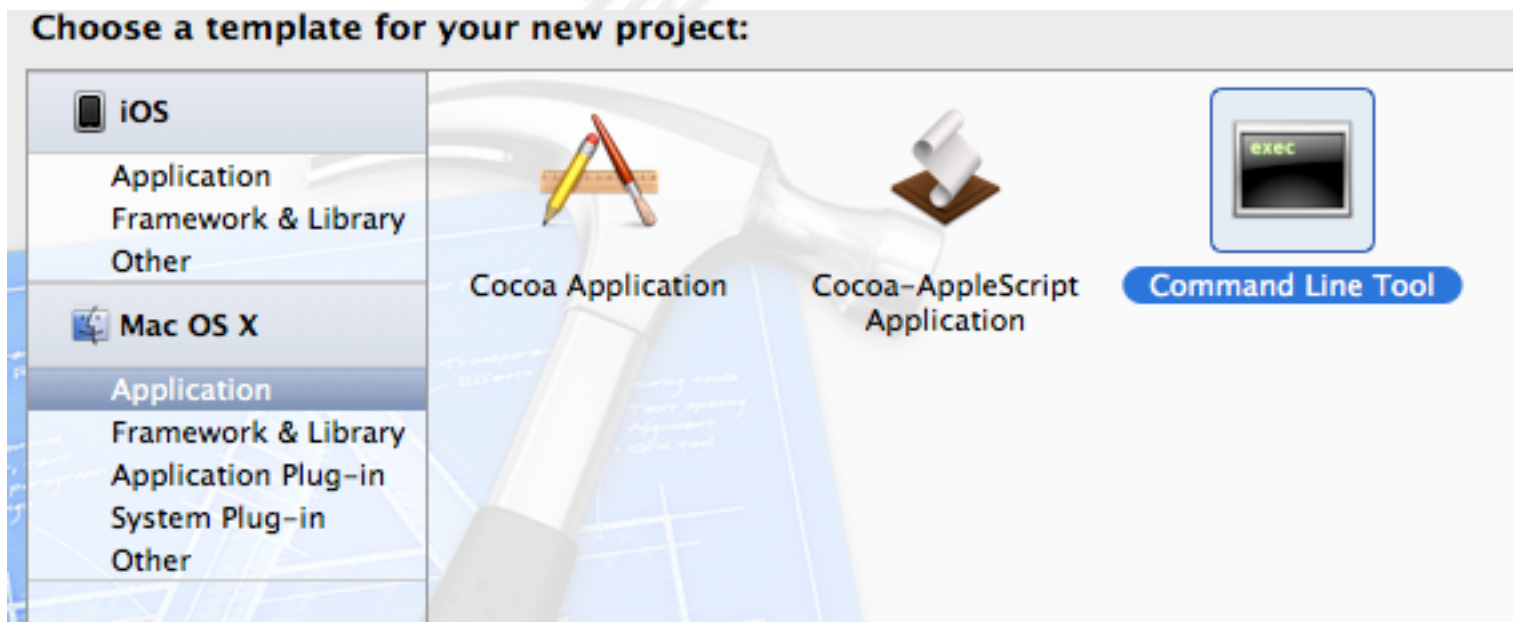
千锋3G学院





# 创建服务器CGI工程

- 创建Xcode “Command Line Tool”工程
- 该程序是作为CGI程序



# 选择Foundation类型

- 确保选择
- Foundation
- 因为Foundation是
- OC类型





## main.m 文件

```
#import <Foundation/Foundation.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define _START \
{ \
FILE *dout; \
dout = fopen("/tmp/ios_debug", "a"); \

#define _END \
fclose(dout); \
}
void test(void);
void doPostFormData(char *string, NSString *boundaryString);
void doCommon(char *string);
NSString * getName(NSString *string);
NSString * getValue(NSString *string);
```



## main.m 文件(续集) getName函数

```
NSString * getName(NSString *string)
{
    NSRange beginRange = [string rangeOfString:@"name="];
    NSString *s = [string substringFromIndex:(beginRange.location)+beginRange.length];
    NSRange endRange = [s rangeOfString:@""];
    NSString *s2 = [s substringToIndex:endRange.location];

    return s2;
}
```



# getName函数解释

```
/*  
    Content-Disposition: form-data; name="myUserName2"\r\n  
    \r\n  
    12  
    \r\n  
  
    \r\n  
    \r\n  
    12  
    \r\n  
  
    myUserName2  
*/
```

## main.m 文件(续集) getValue函数

```
NSString * getValue(NSString *string)
{
    NSRange endRange = [string rangeOfString:@"\r\n"
options:NSBackwardsSearch];
    string = [string substringToIndex:endRange.location];

    NSRange beginRange = [string rangeOfString:@"\r\n\r\n"];
    NSString *s2 = [string substringFromIndex:(beginRange.location
+beginRange.length)];
    return s2;
}
```



## getValue函数解释

```
/*  
    Content-Disposition: form-data;  
name="myUserName2"\r\n  
    \r\n  
    12  
    \r\n  
    Content-Disposition: form-data;  
name="myUserName2"\r\n  
    \r\n  
    12  
*/
```



# main.m doPostFormData

```
void doPostFormData(char *string, NSString *boundaryString) {
    NSString *formString = [NSString stringWithUTF8String:string];
    NSMutableString *mutableString = [NSMutableString stringWithString:formString];

    /* 删除结尾 */
    NSString *endString = [NSString stringWithFormat:@"--%@--\r\n",
boundaryString];
    NSRange endRange = [mutableString rangeOfString:endString];
    [mutableString deleteCharactersInRange:endRange];

    /* 以界限分割 */
    NSArray *array = [mutableString componentsSeparatedByString:
[NSString stringWithFormat:@"--%@\\r\\n", boundaryString]];

    NSInteger len = [array count];
```





## main.m doPostFormData(续集)

```
printf("{");  
for (int i = 1; i < len; i++) {  
    NSString *s = [array objectAtIndex:i];  
    NSString *name = getName(s);  
    NSString *value = getValue(s);  
  
    printf("\"%s\":\"%s\"",  
           [name UTF8String],  
           [value UTF8String]);  
    if (i != len-1) {  
        printf(",");  
    }  
}  
printf("}");  
}
```



# main.m test 处理Post2协议

```
void test(void) {  
    char *requestContentType = getenv("CONTENT_TYPE");  
    char *requestMethod = getenv("REQUEST_METHOD");  
    char *serverName = getenv("SERVER_NAME");  
  
    printf("Content-Type: text/plain\r\n");  
    printf("\r\n");  
  
    _START  
    fprintf(dout, "REQUEST_METHOD: %s\n", requestMethod);  
    fprintf(dout, "SERVER_NAME: %s\n", serverName);  
    _END
```



## main.m test 处理Post2协议(续)

```
size_t requestContentLength = atoi(getenv("CONTENT_LENGTH"));
char *input = (char *) malloc(requestContentLength+1);
memset(input, '\0', requestContentLength+1);
int haveRead = 0;
while (1) {
    ssize_t ret = read(0, input+haveRead, requestContentLength);
    haveRead += ret;
    if (haveRead == requestContentLength) {
        break;
    }
}
_START
fprintf(dout, "CONTENT_LENGTH: %s\n", getenv("CONTENT_LENGTH"));
fprintf(dout, "CONTENT_TYPE: %s\n", requestContentType);
fprintf(dout, "CONTENT: %s\n", input);
_END
```



## main.m test 处理Post2协议(续)

```
if (strstr(requestContentType, "multipart/form-data") >= 0) {  
    /* 这里只是处理文本文件，不能处理二进制文件  
       请加上二进制的处理 提示用NSData或者直接在char *上处理 */  
    NSString *boundaryString = [NSString  
stringWithUTF8String:requestContentType];  
    NSRange range = [boundaryString rangeOfString:@"multipart/form-data;  
boundary="];  
    boundaryString = [boundaryString substringFromIndex:(range.location  
+range.length)];  
  
    doPostFormData(input, boundaryString);  
}  
free(input);  
}
```



## main.m 主函数

```
int main (int argc, const char * argv[])
{
    @autoreleasepool {
        test();
    }
    return 0;
}
```



## 编译post2.cgi过程

- gcc -o post2.cgi main.m -framework Foundation
- clang -o post2.cgi main.m -framework Foundation
- 使用gcc或者clang来编译，优先选择clang编译器编译
- 从这里可以看出post2.cgi其实就是一个可执行程序

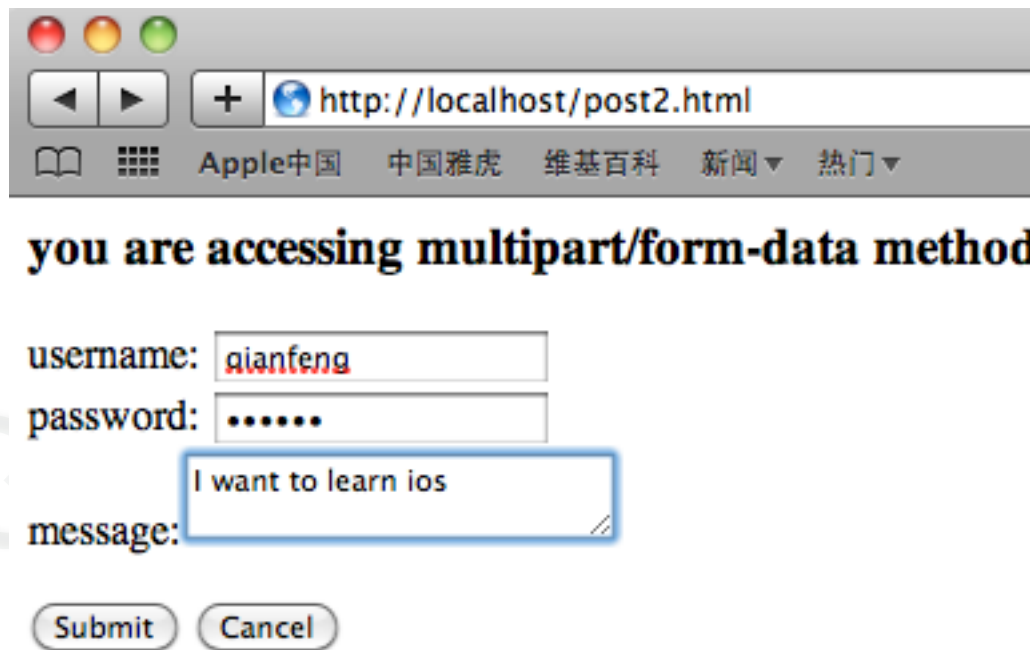


## 部署post2.cgi程序

- 将post2.cgi程序拷贝到/Library/WebServer/CGI-Executables中即可
- `cp post2.cgi /Library/WebServer/CGI-Executables/`

# POST测试

- 在刚才的网页中，输入字符串
- 然后点击“执行”，然后显示结果如下：
- 显示了JSON数据格式
- 同时查看/tmp/ios\_debug文件



you are accessing multipart/form-data method

username:

password:

message:



```
{ "myUserName": "qianfeng", "myPassWord": "123456", "myMessage": "I want to learn ios" }
```





## 查看/tmp/ios\_debug调试文件

- 查看/tmp/ios\_debug文件，然后看看POST传过来的协议

千锋3G学院



# 服务器收到的multipart/form-data数据

multipart/form-data; boundary=----WebKitFormBoundaryWwx4AnQnaM5WWwxb

POST

-----WebKitFormBoundaryWwx4AnQnaM5WWwxb

Content-Disposition: form-data; name="myUserName"

qianfeng

-----WebKitFormBoundaryWwx4AnQnaM5WWwxb

Content-Disposition: form-data; name="myPassWord"

aaa

-----WebKitFormBoundaryWwx4AnQnaM5WWwxb

Content-Disposition: form-data; name="myMessage"

I want to learn ios

-----WebKitFormBoundaryWwx4AnQnaM5WWwxb--

localhost

Apache/2.2.17 (Unix) mod\_ssl/2.2.17 OpenSSL/0.9.8r DAV/2

HTTP/1.1

80

/cgi-bin/ios1.cgi



## boundary分隔符

- Content-Type", "multipart/form-data; boundary=ABCD");
- 然后，将每个字段用“--分隔符”分隔，最后一个“-分隔符--”表示结束。



# HTTP POST 3

---

multipart/form-data 文本文件方式



## POST方式2 multipart/form-data 提交文本文件

- 提交文本文件

千锋3G学院



# 传输2个文件http协议文本

```
-----WebKitFormBoundary7fumcaAtuFAfxd0C\r\n
```

```
Content-Disposition: form-data; name="myMessage"; filename="a.c"\r\n
```

```
Content-Type: application/octet-stream\r\n
```

```
\r\n
```

```
#include <stdio.h>
```

```
\r\n
```

```
-----WebKitFormBoundary7fumcaAtuFAfxd0C\r\n
```

```
Content-Disposition: form-data; name="myMessage2"; filename="get.m"\r\n
```

```
Content-Type: application/octet-stream\r\n
```

```
\r\n
```

```
#import <Foundation/Foundation.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
\r\n
```

```
-----WebKitFormBoundary7fumcaAtuFAfxd0C--
```



# HTTP POST 4

---

multipart/form-data 二进制文件方式

## POST方式2 multipart/form-data 提交二进制文件

- 提交二进制文件，图片，压缩文件，所有二进制文件
- 这里为了方便服务器程序编写，二进制文件一次只是上传一个，并且存在服务器 /tmp/app 目录中
- 这里使用纯C写二进制图片存放





## post3.c 文件

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>

#define FILEPATH "/tmp/app/"

int main(int argc, char *argv[]) {
    printf("Content-Type: text/html\r\n");
    printf("\r\n");

    mkdir(FILEPATH, 0777);
```



## post3.c 文件(续)

```
char *pMethod = getenv("REQUEST_METHOD");
if(strcmp(pMethod, "POST") == 0) {
    char *pCntLen = getenv("CONTENT_LENGTH");
    int StrLen = atoi(pCntLen);
    char *readstr=(char *)malloc(StrLen+1);
    fread(readstr,StrLen,1,stdin);
    {
        // write to debug file
        FILE *dout;
        dout = fopen("/tmp/ios_debug", "a");
        fwrite(readstr, StrLen, 1, dout);
        fclose(dout);
    }
}
```



## post3.c 文件(续)

```
char fname[80]="";
char temp1[80]="";
char *temp2=NULL;
char *p=NULL;
int strl=0;
if(strstr(readstr,"filename=")) {
    p=strstr(readstr,"filename=");
    strl = strcspn(p+10,"\\");

    strncpy(temp1,p+10,strl);
    if((temp2 = strchr(temp1,"\\")) != 0) {
        strcpy(fname,temp2+1);
    } else {
        strcpy(fname,temp1);
    }
} else {
    printf("error\n");
    return 0;
}
```



## post3.c 文件(续)

```
int n=0;
int firstLineMark=0;
int firstLineCount=0;
int headCount=0;
while(n<4) {
    if(*(readstr++)=='\n') {
        firstLineMark++;
        n++;
    }
    if(firstLineMark==0) {
        firstLineCount++;
    }
    headCount++;
}
```



## post3.c 文件(续)

```
StrLen = StrLen-headCount;
char filename[100]="";
sprintf(filename,"%s%s",FILEPATH,fname);
FILE *fp;
if((fp=fopen(filename,"wb"))==NULL) {
    printf("error open file\n");
    return 0;
}
fwrite(readstr,StrLen-firstLineCount-5,1,fp);
fclose(fp);
printf("ok");
}
return 0;
}
```

# 编译c程序post3.cgi

- `gcc -o /Library/WebServer/CGI-Executables/post3.cgi post3.c`
- 然后
- 程序运行后会自动创建一个 `/tmp/app` 目录
- 这里 `/tmp/app` 是存放图片的目录



# post3.html 实现

```
<html>
  <head>
    <title>upload</title>
  </head>
  <body>
    <p>use multipart/form-data method=post file
    <form enctype="multipart/form-data" action="/cgi-bin/post3.cgi" method="post">
      <br> message:
      <input type="file" name="filename" size="220">
      <p>
        <input type="submit" value="Submit">
        <input type="reset" value="Cancel">
      </form>
    </body>
</html>
```

将post3.html 拷贝到/Library/WebServer/Documents中



## 测试一下

- 在浏览器中输入网址
- `http://localhost/post3.html`
- 然后上传一个图片
- 查看是否 服务器上的 `/tmp/app`有文件存在，并且文件是否完成一致





# iOS客户端

---

iOS程序访问服务器

模拟浏览器访问服务器资源



# 创建一个空的iPhone工程

- 对于iOS5来说，创建一个Empty Application
- 对于之前的，创建一个Window Based Application
- 并且创建一个基于UIViewController的类  
RootViewController



# RootViewController.h 类定义

```
#import <UIKit/UIKit.h>
```

```
@interface RootViewController : UIViewController  
{  
    NSURLConnection *requestConnection;  
    NSMutableData *allData;  
}
```

```
@end
```



# RootViewController.m 网络3函数

```
- (void)connectionDidFinishLoading:(NSURLConnection *)connection{
    [requestConnection release];
    /* 把allData转化成NSString对象 */
    NSString *string = [[NSString alloc] initWithData:allData encoding:NSUTF8StringEncoding];

    NSLog(@"string is %@", string);
    [allData release];
    [string release];
}

/* 函数3 每接受一小段数据就会调用 */
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
    NSLog(@"functiono %s is calling", __func__);
    [allData appendData:data];
    /* 把data追加到 allData最后 */
}
```



# RootViewController.m 网络3函数(续)

```
/* 函数2 接收完HTTP协议头的函数 真正数据开始接受时候调用 */
- (void)connection:(NSURLConnection *)connection
    didReceiveResponse:(NSURLResponse *)response {
    allData = [[NSMutableData alloc] initWithLength:0];

    NSHTTPURLResponse *res = (NSHTTPURLResponse *)response;
    NSInteger statusCode = [res statusCode];
    if (statusCode >= 400 ) {
        NSLog(@"HTTP ERRORR CODE %d", statusCode);
    }
    NSDictionary *dic = [res allHeaderFields];
    NSLog(@"all Header Fields");
    NSLog(@"%@", dic);
    int len = [[dic objectForKey:@"Content-Length"] intValue];
    NSLog(@"server data len is %d", len);
}
```

# RootViewController.m GET访问

```
- (void) testGet
{
    NSString *urlString = @"http://localhost/cgi-bin/get.cgi?
myUserName=1233&myPassWord=111&myMessage=323";
    NSURL *url = [NSURL URLWithString:urlString];
    NSURLRequest *request = [[NSURLRequest alloc] initWithURL:url];
    requestConnection = [[NSURLConnection alloc] initWithRequest:request
delegate:self];
    [request release];
}
```



## 调用GET函数，并且运行一下

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    [self testGet];
}
```



# 编写GET方法第二种方法ASI

- ASIHTTPRequest
- 加入头文件 `#import "ASIHTTPRequest.h"`
- 加入如下库framework
- `CoreGraphics.framework`
- `CFNetwork.framework`
- `libz.dylib`
- `SystemConfiguration.framework`
- `MobileCoreServices.framework`





## RootViewController.m 编写testGetASI

```
- (void) testGetASI {  
    NSString *urlString = @"http://localhost/cgi-bin/get.cgi?  
myUserName1=1233&myPassWord1=111&myMessage1=  
323";  
    NSURL *url = [NSURL URLWithString:urlString];  
    ASIHTTPRequest *request = [ASIHTTPRequest  
requestWithURL:url];  
    [request setTag:100];  
    [request setDelegate:self];  
    [request startAsynchronous];  
}
```



# RootViewController.m 完成回调函数

```
- (void)requestFinished:(ASIHTTPRequest *)request
{
    if (request.tag == 100) {
        NSString *s = [request responseString];
        NSLog(@"string is %@", s);
    }
}
```



## RootViewController.m 调用testGetASI

```
- (void)viewDidLoad  
{  
    [super viewDidLoad];  
  
    [self testGetASI];  
}
```



## 运行一下

- 比较2中运行。这是原生的http get请求和ASIHTTP请求

千锋3G学院



# 继续增加POST1方法

千锋3G学院

# RootViewController.m POST urlencoded

```
- (void) testPostUrl
{
    NSString *body =
@"myUserName=12333323&myPassWord=11321&myMessage=332323";

    NSString *urlString = @"http://localhost/cgi-bin/post1.cgi";
    NSURL *url = [NSURL URLWithString:urlString];

    NSMutableURLRequest* req;
    req = [NSMutableURLRequest requestWithURL:url
                                   cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
                                   timeoutInterval:20.0f];
```

## POST urlencoded(续)

```
[req setHTTPMethod:@"POST"];
[req setHTTPShouldHandleCookies:NO];
[req setValue:@"application/x-www-form-urlencoded"
    forHTTPHeaderField:@"Content-Type"];
int contentLength = [body
    lengthOfBytesUsingEncoding:NSUTF8StringEncoding];

[req setValue:[NSString stringWithFormat:@"%d", contentLength]
    forHTTPHeaderField:@"Content-Length"];
[req setHTTPBody:[body dataUsingEncoding:NSUTF8StringEncoding]];
requestConnection = [[NSURLConnection alloc]
    initWithRequest:req delegate:self];
}
```



## 运行一下 调用POST函数

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // [self testGet];
    [self testPostUrl];
}
```

目前在讲多网络连接之前，上面三个函数只能2选1调用，一次只能调用一个，请注释相应代码





# POST1 ASI第二种方式

千锋3G学院



```
- (void) testPostUrlASI
{
    NSString *urlString = @"http://localhost/cgi-bin/post1.cgi";
    NSURL *url = [NSURL URLWithString:urlString];

    ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:url];
    [request setPostValue:@"12333323" forKey:@"myUserName"];
    [request setPostValue:@"11321" forKey:@"myPassWord"];
    [request setPostValue:@"332323" forKey:@"myMessage"];

    [request setTag:101];
    [request setDelegate:self];
    [request startAsynchronous];
}
```



# 继续增加POST2方法

千锋3G学院



# POST multipart

```
#define TEST_FORM_BOUNDARY @"123QianFeng12345678"

- (void) testPostFormData:(NSData*)data {
    NSString *urlString = @"http://localhost/cgi-bin/post1.cgi";
    NSURL *url = [NSURL URLWithString:urlString];
    NSMutableURLRequest* req;
    req = [NSMutableURLRequest requestWithURL:url
                                     cachePolicy:NSURLRequestReloadIgnoringLocalAndRemoteCacheData
                                     timeoutInterval:20.0f];
    NSString *contentType = [NSString stringWithFormat:
                              @"multipart/form-data; boundary=%@", TEST_FORM_BOUNDARY];
    [req setHTTPShouldHandleCookies:NO];
    [req setHTTPMethod:@"POST"];
    [req setValue:contentType forHTTPHeaderField:@"Content-Type"];
    [req setValue:[NSString stringWithFormat:@"%d", [data length]]
               forHTTPHeaderField:@"Content-Length"];
    [req setHTTPBody:data];
    requestConnection = [[NSURLConnection alloc] initWithRequest:req delegate:self];
}
```



# nameValString 自定义函数

```
- (NSString*) nameValString: (NSDictionary*) dict {
    NSArray* keys = [dict allKeys];
    NSString* result = [NSString string];
    int i;
    for (i = 0; i < [keys count]; i++) {
        result = [result stringByAppendingString:
            [@"--" stringByAppendingString:
                [TEST_FORM_BOUNDARY stringByAppendingString:
                    [@"\r\nContent-Disposition: form-data; name=\"" stringByAppendingString:
                        [[keys objectAtIndex: i] stringByAppendingString:
                            [@"\"\\r\\n\\r\\n" stringByAppendingString:
                                [[dict valueForKey: [keys objectAtIndex: i]] stringByAppendingString: @"\r\n"]]]]]]]];
    }

    return result;
}
```



# testPostFormData 函数

```
- (void) testPostFormData {
    NSDictionary *dic = [NSDictionary dictionaryWithObjectsAndKeys:
        @"1234", @"myUserName1", @"3456", @"myPassWord1", nil];
    NSString *param = [self nameValString:dic];
    NSString *footer = [NSString stringWithFormat:@"\r\n--%@--\r\n",
        TEST_FORM_BOUNDARY];
    param = [param stringByAppendingString:[NSString stringWithFormat:
        @"--%@ \r\n", TEST_FORM_BOUNDARY]];
    param = [param stringByAppendingString:@"Content-Disposition: form-data; name=
    \"myMessage1\" \r\n\r\n"];
    NSString *myMessage1 = @"23223";
    NSMutableData *data = [NSMutableData data];
    [data appendData:[param dataUsingEncoding:NSUTF8StringEncoding]];
    [data appendData:[myMessage1 dataUsingEncoding:NSUTF8StringEncoding]];
    [data appendData:[footer dataUsingEncoding:NSUTF8StringEncoding]];
    NSLog(@"param:%@", param);
    [self testPostFormData:data];
}
```

## 运行一下

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    //[self testGet];
    //[self testPostUrl];
    [self testPostFormData];
}
```

目前在讲多网络连接之前，上面三个函数只能3选1调用，一次只能调用一个，请注释相应代码



# 继续增加POST3方法

千锋3G学院





# 上传文件(文本和二进制)

- 上传文本文本在上一节中讲述过。主要在doPostFormDat  
a中处理
- 下面讲述二进制，尤其是图片的上传过程
- 这里为了方便服务器编写，这里要求图片上传只能有一个表单项。也就是multipart/form-data中只有一个数据项
- 二进制/文本表单提交都必须使用multipart/form-data格式



# iOS客户端实现

- 向工程中加入一个图片比如 `testicon.png` 图片

千锋36学院



# RootViewController.m testPicUpload函数

```
- (void) testPicUpload {
    NSString *picPath = [[NSBundle mainBundle] pathForResource:@"testicon" ofType:@"png"];
    NSData *png = [NSData dataWithContentsOfFile:picPath];
    NSString *param = @"";
    NSString *footer = [NSString stringWithFormat:@"%r\n--%@\r\n", TEST_FORM_BOUNDARY];

    param = [param stringByAppendingString:
        [NSString stringWithFormat:@"--%@\r\n", TEST_FORM_BOUNDARY]];
    param = [param stringByAppendingString:@"Content-Disposition: form-data; name=\"filename
\";filename=\"testicon.png\"r\nContent-Type: image/pngr\nr\n"];

    NSMutableData *data = [NSMutableData data];
    [data appendData:[param dataUsingEncoding:NSUTF8StringEncoding]];
    [data appendData:png];
    [data appendData:[footer dataUsingEncoding:NSUTF8StringEncoding]];
    NSLog(@"param:%@", param);
    [self testPostPicData:data];
}
```



# testPostPicData: 函数实现

```
- (void) testPostPicData:(NSData*)data
{
    NSString *urlString = @"http://localhost/cgi-bin/ios3.cgi";
    NSURL *url = [NSURL URLWithString:urlString];

    NSMutableURLRequest* req;
    req = [NSMutableURLRequest requestWithURL:url
                                   cachePolicy:NSURLRequestReloadIgnoringLocalAndRemoteCacheData
                                   timeoutInterval:20.0f];
    NSString *contentType = [NSString stringWithFormat:
                              @"multipart/form-data; boundary=%@", TEST_FORM_BOUNDARY];
    [req setHTTPShouldHandleCookies:NO];
    [req setHTTPMethod:@"POST"];
    [req setValue:contentType forHTTPHeaderField:@"Content-Type"];
    [req setValue:[NSString stringWithFormat:@"%d",
                              [data length]] forHTTPHeaderField:@"Content-Length"];
    [req setHTTPBody:data];
    requestConnection = [[NSURLConnection alloc] initWithRequest:req delegate:self];
}
```



## 调用testPicUpload函数

```
- (void)viewDidLoad  
{  
    [super viewDidLoad];  
    // Do any additional setup after loading the view from its  
    nib.  
    [self testPicUpload];  
}
```



运行一下

千锋3G学院



# 上传图片ASI第二种方法

千锋3G学院



```
- (void) testPostPicASI {  
    NSString *urlString = @"http://localhost/cgi-bin/post3.cgi";  
    NSURL *url = [NSURL URLWithString:urlString];  
  
    ASIFormDataRequest *request = [ASIFormDataRequest  
requestWithURL:url];  
    NSString *path = [[NSBundle mainBundle]  
pathForResource:@"testicon" ofType:@"png"];  
    [request setFile:path forKey:@"myphoto"];  
    [request setTag:102];  
    [request setDelegate:self];  
    [request startAsynchronous];  
}
```





运行一下

千锋3G学院

# 改进iOS客户端

- 从相机，或者照片程序中取图片，然后上传服务器
- 在进行图片上传之前，先做一个图片预览的一个小例子



# 如何往iPhone模拟器中加入图片

- 在Finder中找到你要放入iPhone模拟器的图片
- 然后在iPhone模拟器中打开Safari浏览器
- 然后在Finder中用鼠标左键选中图片不动然后拖动图片到iPhone模拟器中的Safari浏览器中。如图所示



## 如何往iPhone模拟器中加入图片(续)

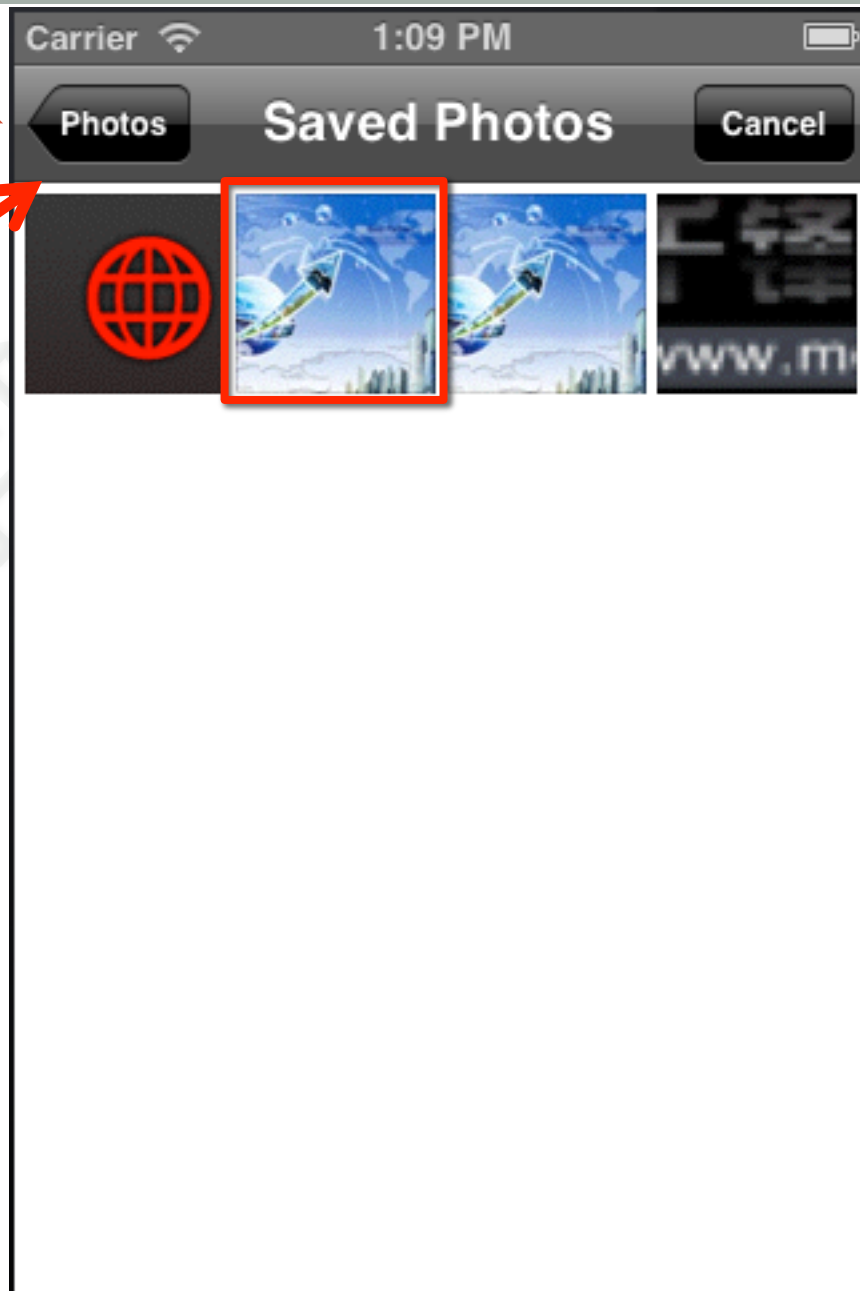
- 然后用鼠标长按iPhone模拟器中Safari浏览器上面的图片，出现如图所示：
- 然后选中“Save Image”就可以把该图片保存在iPhone模拟器中的相册里面





## 图片、相机选取原理

- 点击按钮启动相册或者相机





# 创建iOS工程

- 创建一个空的工程
- 创建一个基于UIViewController的类RootViewController



# 在AppDelegate.m中添加

```
#import "RootViewController.h"
```

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
    ...
    RootViewController *rvc = [[[RootViewController alloc] initWithNibName:
        @"RootViewController" bundle:nil] autorelease];
    self.window.rootViewController = rvc;
    [self.window makeKeyAndVisible];
    return YES;
}
```



# RootViewController.h 申明

```
#import <UIKit/UIKit.h>

@interface RootViewController : UIViewController
    <UIImagePickerControllerDelegate, UINavigationControllerDelegate>
{
    UIImageView *myImageView;

}
@property (nonatomic, retain) IBOutlet UIImageView *myImageView;

- (IBAction) selectImage:(id)sender;

@end
```





# RootViewController.xib

- 连接IBOutlet和IBAction， 包含一个按钮和一个图片





# RootViewController.m 按钮点击

```
@implementation RootViewController
@synthesize myImageView;
- (void)dealloc {
    self.myImageView = nil;
    [super dealloc];
}
- (IBAction) selectImage:(id)sender {
    /* 启动相机或者相框 */
    UIImagePickerController *imagePickerController =
        [[UIImagePickerController alloc] init];
    imagePickerController.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    /* 使用相框 */
    //imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera;
    imagePickerController.delegate = self;
    [self presentViewController:imagePickerController animated:YES];
    /* 把imagePickerController展现在屏幕上 */
    [imagePickerController release];
}
```



# RootViewController.m 代理实现

```
#pragma mark -
#pragma mark UIImagePickerController
- (void)imagePickerController:(UIImagePickerController *)picker
    didFinishPickingMediaWithInfo:(NSDictionary *)info {
    NSLog(@"image select");
    /* 1. 获得图片 */
    UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];

    /* 2. 把图片放在myImageView中 */
    myImageView.image = image;

    /* 3. 关掉picker */
    [picker dismissModalViewControllerAnimated:YES];
}
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
    NSLog(@"image cancel");
    [picker dismissModalViewControllerAnimated:YES];
    /* 退出picker */
}
```



# 运行一下

- 选取照片和相机

```
imagePickerController.sourceType =  
    UIImagePickerControllerSourceTypePhotoLibrary;  
/* 使用相框 */  
//imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera;
```



# 作业，把图片选取后然后上传服务器

- 提示函数UIImagePNGRepresentation
- 把一个UIImage图片对象转化为NSData对象
- UIImage \*image = UIImage \*image = [UIImage imageNamed:@"testicon.png"];
- NSData \*jpeg = UIImagePNGRepresentation(image);



# 附：Wireshark使用

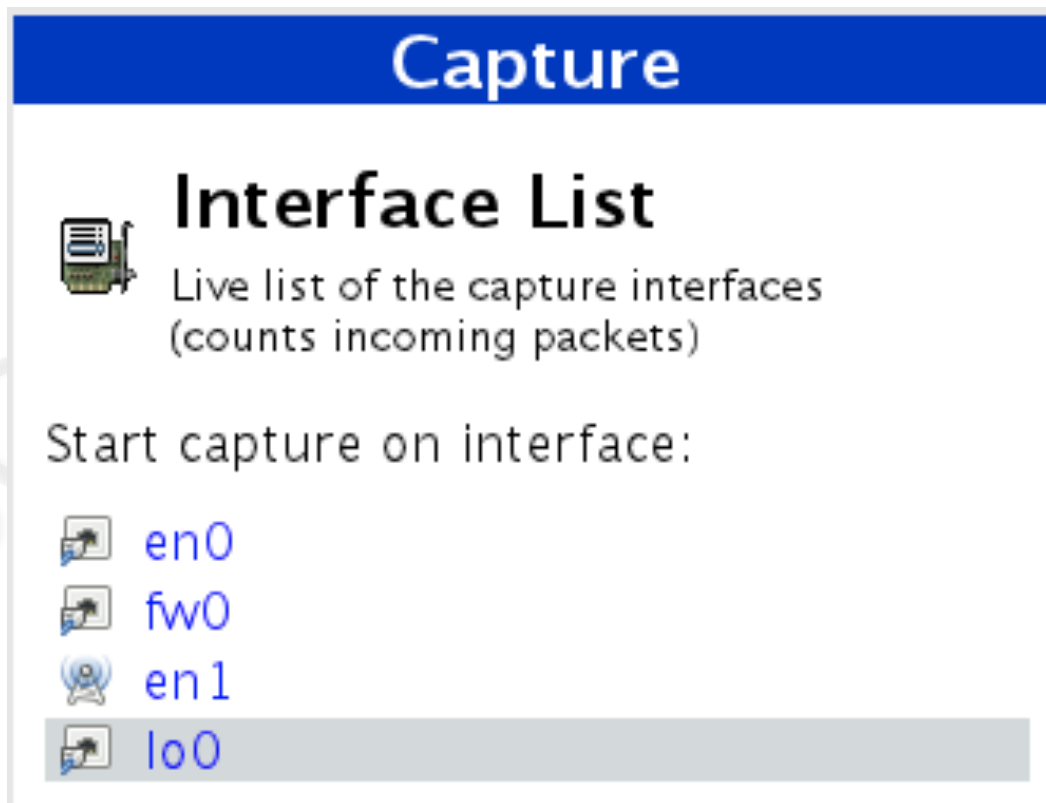
---

抓包工具



# wireshark

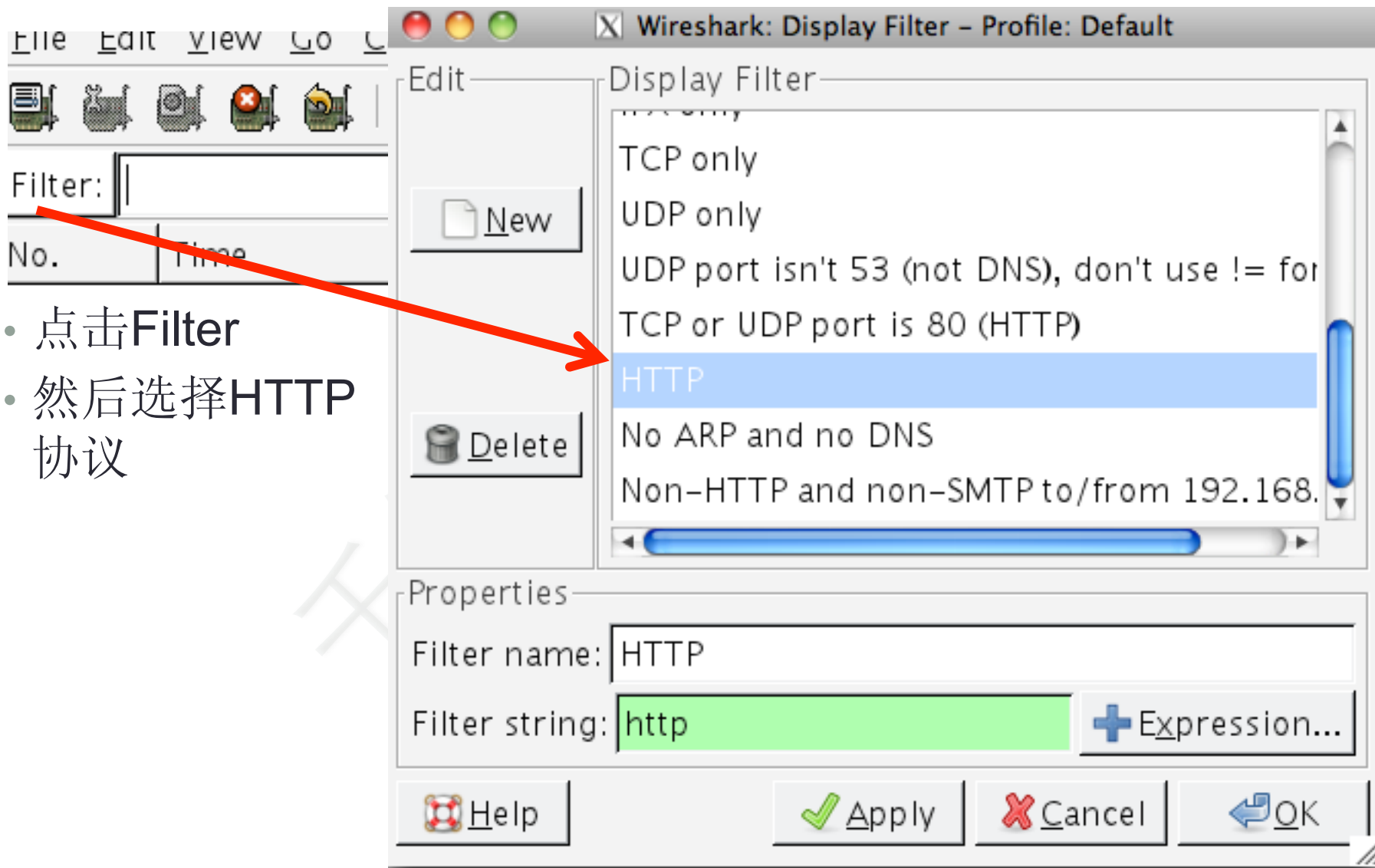
- 这里捕捉lo0循环设备





# 过滤只是抓包HTTP协议

- 点击Filter
- 然后选择HTTP协议





# 开始抓包

Filter: http Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
13	3.096369	127.0.0.1	127.0.0.1	HTTP	102	POST /cgi-bin/ios1.cgi HTTP/1.1 (application
15	3.207175	127.0.0.1	127.0.0.1	HTTP	279	HTTP/1.1 200 OK

- 这里有2条记录，一条HTTP请求，一条HTTP响应

# HTTP请求

Filter:  Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
13	3.096369	127.0.0.1	127.0.0.1	HTTP	102	POST /cgi-bin/ios1.cgi HTTP/1.1 (application/octet-st
15	3.207175	127.0.0.1	127.0.0.1	HTTP	279	HTTP/1.1 200 OK

\*\*\*\*\*

▼ Hypertext Transfer Protocol

- ▷ POST /cgi-bin/ios1.cgi HTTP/1.1\r\n
 Host: localhost\r\n
 Origin: http://localhost\r\n
 Content-Length: 215\r\n
 User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10\_6\_8; zh-cn) AppleWebKit/533.21.1 (KHTML, like Gecko) Version\r\n
 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryxt0g9oy9NMnpGaTi\r\n
 Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5\r\n
 Referer: http://localhost/ios1.html\r\n
 Accept-Language: zh-cn\r\n
 Accept-Encoding: gzip, deflate\r\n
 Connection: keep-alive\r\n
 \r\n
 [Full request URI: http://localhost/cgi-bin/ios1.cgi]
- ▷ MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "----WebKitFormBoundaryxt0g9oy9NMnpGaTi"



## HTTP响应头

Filter:  Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
13	3.096369	127.0.0.1	127.0.0.1	HTTP	102	POST /cgi-bin/ios1.cgi HTTP/1.1 (application/octet-stre
15	3.207175	127.0.0.1	127.0.0.1	HTTP	279	HTTP/1.1 200 OK

Frame 15: 279 bytes on wire (2232 bits), 279 bytes captured (2232 bits)

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)

Transmission Control Protocol, Src Port: http (80), Dst Port: 49271 (49271), Seq: 1, Ack: 757, Len: 223

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Date: Fri, 06 Jan 2012 02:46:51 GMT\r\n

Server: Apache/2.2.17 (Unix) mod\_ssl/2.2.17 OpenSSL/0.9.8r DAV/2\r\n

Content-Length: 0\r\n

Keep-Alive: timeout=5, max=100\r\n

Connection: Keep-Alive\r\n

Content-Type: text/plain\r\n

\r\n



• Thanks



## 课后作业1

实验心得

- 我叫艾锋，今天我学习了
  1. \_\_\_\_\_
  2. \_\_\_\_\_
  3. \_\_\_\_\_
- 我觉得最难的部分是\_\_\_\_\_, 因为\_\_\_\_\_。
- 我觉得掌握的最熟练的是\_\_\_\_\_。
- 我相信自己一定能够努力学好iPhone编程！



## 课后作业2

- 整理今天学到的新函数，记录到一个文本文件中



## 课后作业3

- 尽可能重复整个实验
- 次数不限



# 放飞梦想 大有希望!

---

千锋寄语