# 使用AutoGen生成代码框架

你提供一个**业务C++头文件**和一个**目的目录**，AutoGen它就会根据这个头文件在目的目录中生成框架代码。

cd auto\_gen; ./auto\_gen.sh

USAGE : ./auto\_gen.sh <head\_filename> <dest\_dir>

EXAMPLE: ./auto\_gen.sh aoitem\_msg.h ~/my\_test

但是，AutoGen还没有智能到可以分析任意的C++头文件，所以这个业务头文件需要遵循一定的格式。

**文件testservice.h**

#ifndef FCGI\_DEMO\_ECHO\_H

#define FCGI\_DEMO\_ECHO\_H

#include <string>

#include <stdint.h>

#include "help\_macros4web.h"

#include "web\_view.h"

#include "action\_params.h"

#include "web\_svc\_cntl.h"

namespace fcgi

{

namespace demo

{

namespace echo

{

SERVICE class CEcho

{

public:

CEcho();

~CEcho();

int Initialize(const char\* pszVhostRootPath, const char\* pszServiceConfigFile);

void Reset();

public:

EXPORT uint32\_t GetLoc(

const web\_solution::web\_framework::CActionParams& rParams,

web\_solution::web\_framework::CWebViewData& rData,

web\_solution::web\_framework::CWebSvcCntl& rCntl

);

public:

int RenderGetLoc( std::string& sOutputBuffer,

web\_solution::web\_framework::CWebViewData& rData,

web\_solution::web\_framework::CWebSvcCntl& rCntl

);

private:

int VerifyGetLoc(

const web\_solution::web\_framework::CActionParams& rParams);

};

}

}

}

#endif /\* FCGI\_DEMO\_ECHO\_H\*/

**SERVICE**宏用于标识一个class为业务Service（CTestService），它的某些方法需要暴露出来供前端调用；

**EXPORT**宏用于标识Service中的这个方法需要暴露给前端调用，AutoGen4Web会收集这些方法，并在粘接剂代码中生成路由分发代码。

上述代码中，我们指定了一个名为Echo的Action，伴随这个Action还需要键入一个EchoRender和VerifyEchoInput方法，分别用于渲染输出页面和校验用户输入。

完成头文件后保存到~/echo.h，然后执行下面步骤来生成框架：

1）cd auto\_gen

2）sh auto\_gen.sh ~/ echo.h ~/webapps/生成框架，webapps下会生成一个echo目录：



lib——是自动生成的粘接剂代码；

src——下面是用户自己的代码，用户需要把自己编写的h和cpp等放入其中；

cgi——里面的Makefile可构建出FCGI；

websvc——暂时不适用（后续支持so加载）。

3）cd src

4）cp ~/echo.h . 复制业务头文件到src目录下；

接下来开始编写业务逻辑代码，文件echo.cpp。

#include <string>

#include <sstream>

#include <iostream>

#include <map>

#include <vector>

#include "echo.h"

using namespace std;

using namespace web\_solution::web\_framework;

using namespace fcgi::demo::echo;

int CEcho::Initialize(const char\* pszVhostRootPath, const char\* pszServiceConfigFile)

{

int iRet = 0;

return iRet;

}

uint32\_t CEcho::GetLoc(

const web\_solution::web\_framework::CActionParams& rParams,

web\_solution::web\_framework::CWebViewData& rData,

web\_solution::web\_framework::CWebSvcCntl& rCntl

)

{

uint32\_t dwRet = 0;

string areaData = "Hello Echo Demo!";

string gettest = "";

if (rParams.GetVal("gettest", gettest) != 0)

{

gettest = "";

}

string method = "";

if (rParams.GetVal("HttpMethod", method) != 0)

{

method = "";

}

string prid = "";

if (rParams.GetCookieVal("prid", prid) != 0)

{

prid = "";

}

areaData += "<br />gettest get:" + gettest;

areaData += "<br />prid cookie:" + prid;

areaData += "<br />method:" + method;

rData.SetUserData("content", areaData);

cout << "test cout"<<"<br />";

// test cookie

rData.UpdateCookieVal("echo\_test\_cookie", "johnsonzhu", "Sat, 08 Nov 2014 09:26:30 GMT", "yixun.com");

rData.UpdateCookieVal("echo\_test\_cookie2", "johnsonzhu2", "Sat, 08 Nov 2014 09:26:30 GMT", "yixun.com");

rData.DelCookieVal("echo\_test\_cookie2");

// test 302

//string redirectUrl = "http://u.yixun.com/abc?noRedirect=1";

//rCntl.SetLocation(redirectUrl);

//rCntl.SetResult(302);

return dwRet;

}

int CEcho::RenderGetLoc( std::string& sOutputBuffer,

web\_solution::web\_framework::CWebViewData& rData,

web\_solution::web\_framework::CWebSvcCntl& rCntl

)

{

uint32\_t dwRet = 0;

//设置max-age,暂时设置10秒

uint32\_t maxAge = 10;

rCntl.SetBrowserCacheExpireTime(maxAge);

rCntl.SetMimeEncoding("abc");

rCntl.SetMimeType("text/html; charset=utf-8");

string content = "";

string errMsg = "";

rData.GetUserData("errMsg", errMsg);

if (!errMsg.empty())

{

sOutputBuffer = errMsg;

}

else

{

rData.GetUserData("content", content);

sOutputBuffer = content;

}

return dwRet;

}

int CEcho::VerifyGetLoc(

const web\_solution::web\_framework::CActionParams& rParams)

{

uint32\_t dwRet = 0;

return dwRet;

}

CEcho::CEcho()

{

}

CEcho::~CEcho()

{

}

void CEcho::Reset(){}

# 生成CGI

在cgi目录执行make，生成可执行的echo.fcgi。

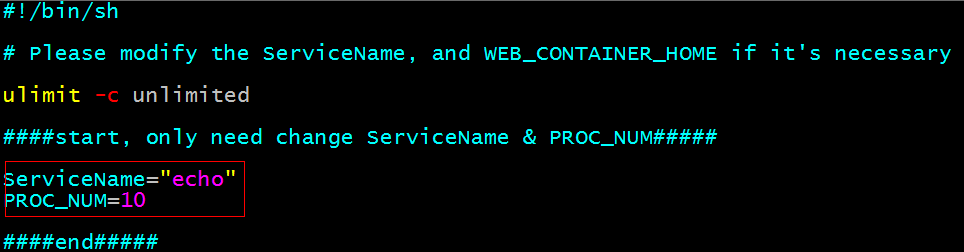
# 部署CGI

## 在webapps目录拷贝新目录echo

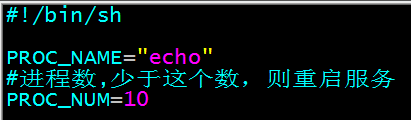
## 在echo目录下的cgi目录部署刚才编译好的echo.fcgi

## 在echo/bin目录下修改配置文件

### service\_manager.sh修改ServerName和进程数PROC\_NUM

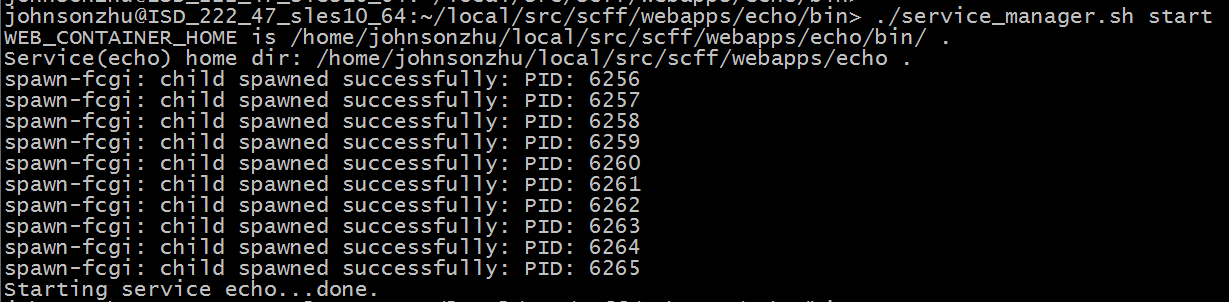


### Protect.sh修改服务名和进程数



## 启动服务

在bin目录执行./service\_manager.sh start



另外可以查看cgi进程情况./service\_manager.sh show



## Nginx配置修改

 location ~/cgi/(\w+)/ {

            fastcgi\_pass unix:/dev/shm/$1.fcgi.sock;

            fastcgi\_param  SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

            include fastcgi\_params;

        }

## 访问服务

访问fcgi，url如：domain.com/cgi/echo/getloc?gettest=abc

