# WebAPI接口调用示例代码

## 1 引入示例工程类库(Net 2.0版本)Step By Step

Step 1: 向MIS 平台服务课申请应用程序接入权限. 得到应用程序身份密钥.

Step 2: 将工程中的Util 工程引入到解决方案中. 将2个配置写入配置文件，如web.config. 其中**WebAPIBaseUrl** 为webAPI地址**，WebAPICallAuth为**身份密钥。接入测试阶段与正式上线后不同。

Step 3: 业务数据拼装，实例化传输类.(自行编写)

Step 4：调用传输方法，处理返回.

附示例工程代码打包（.NET 4.0）：



测试环境地址**WebAPIBaseUrl**: http://nhoaapt01.chn.nexchip/MessageSignBox.API

|  |  |
| --- | --- |
| 接口 | 说明 |
| /api/SignMessage/NewSignMsg | 新增签核箱消息消息 |
| /api/SignMessage/UpdateSignMsg | 更新签核箱消息 |
| /api/SignMessage/UpdateNotifySignMsg | 更新签核箱消息--通知类完成消息  （调用后相应人员之前通知消息状态为已完成） |

## 2 JavaScript 方式.

**直接发送请求，采用CROS方式.**

**///Ajax 方式请求WebAPI**

**function testsendAjax(self,type) {**

**var rowData = exampleTable.row(exampleTable.cell(self.parentNode).index().row).data(); //当前行的值**

**var value = {**

**'appname': rowData.appname,**

**'sendtime': moment().format('YYYY-MM-DDTHH:SS:DD'),**

**'msgbody': rowData**

**};**

**var postData = JSON.stringify(value);**

**$.ajax({**

**type: "POST",**

**contentType: "application/json;charset=utf-8",**

**url: "@ViewBag.WebAPIBaseUrl" + "/SignMessage/UpdateSignMsg",**

**data: postData,**

**crossDomain: true,**

**dataType: "json",**

**//xhrFields: {**

**// withCredentials:true**

**//},**

**beforeSend: function (XMLHttpRequest) {**

**XMLHttpRequest.setRequestHeader("Authorization", "@ViewBag.WebAPICallAuth");**

**},**

**success: function (data) {**

**if (data.success) {**

**//reloadTables()**

**//layer.closeAll();**

**layer.alert("测试成功！");**

**}**

**else {**

**layer.alert("测试失败！" + data.msg || '');**

**}**

**}**

**});**

**}**

**说明： 红色标注实际业务系统数据。 @ViewBag.WebAPIBaseUrl 为提供的API地址，**

**@ViewBag.WebAPICallAuth为身份密钥.**

## 3 C# 方式

**建立了数据传输实体类，以DTO 结尾，相应文件打包和辅助代码在对应目录. 核心代码如下：**

try

{

var postPath = re.postPath;

var postData = Newtonsoft.Json.JsonConvert.DeserializeObject<SignMessageSendDto>(re.dto);

postData.sendtime = DateTime.Now;

var response = HttpClinetHelper.HttpPost(postPath, Newtonsoft.Json.JsonConvert.SerializeObject(postData));

if (response == "")

{

return new BizResult<string>

{

Success = false,

Msg = "网络请求内容为空"

};

}

else

{

var returnD = Newtonsoft.Json.JsonConvert.DeserializeObject<BizResult<List<SignMessageBox>>>(response);

return new BizResult<string>

{

Success =returnD.Success,

Msg=returnD.Msg

};

}

}

catch (Exception ex)

{

return new BizResult<string>

{

Success = false,

Msg = ex.Message

};

}

如上内容总结及注意事项：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 使用方式 | 使用方法步骤&注意事项 | 常见问题解决 |
| 1 | Ajax 方式 | 以jQuery为例，发起Ajax请求.参考示例代码.  - 注意请求头中增加`Authorization`参数. |  |
| 2 | C#方法 | 1. 引入Util类库  - 用到JSON.NET 2. 配置文件中增加APIUrl,AppTokenString中节点. SettingConfig中配置读取方式 3. 拼装业务数据后发送.参考示例代码. | 1. 用到JSON.NET版本与原系统已有冲突. 解决方案：统一版本或者参考修改配置支持多版本. |