**基于C#的APK反编译api调用代码实例**

代码描述：基于C#的APK反编译api调用代码实例

代码平台：聚合数据

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.IO;

using Xfrog.Net;

using System.Diagnostics;

using System.Web;

//----------------------------------

// APK加密调用示例代码 － 聚合数据

// 在线接口文档：<http://www.juhe.cn/docs/76>

// 代码中JsonObject类下载地址:<http://download.csdn.net/download/gcm3206021155665/7458439>

//----------------------------------

namespace ConsoleAPI

{

    class Program

    {

        static void Main(string[] args)

        {

            string appkey = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"; //配置您申请的appkey

            //1.①根据url上传待加密的apk

            string url1 = "<http://op.juhe.cn/ijiami/upload>";

            var parameters1 = new Dictionary<string, string>();

            parameters1.Add("apkurl" , ""); //需要进行加密的APK下载URL地址

            parameters1.Add("key", appkey);//你申请的key

            parameters1.Add("dtype" , ""); //返回数据的格式,xml或json，默认json

            string result1 = sendPost(url1, parameters1, "get");

            JsonObject newObj1 = new JsonObject(result1);

            String errorCode1 = newObj1["error\_code"].Value;

            if (errorCode1 == "0")

            {

                Debug.WriteLine("成功");

                Debug.WriteLine(newObj1);

            }

            else

            {

                //Debug.WriteLine("失败");

                Debug.WriteLine(newObj1["error\_code"].Value+":"+newObj1["reason"].Value);

            }

            //2.②提交加密应用加密

            string url2 = "<http://op.juhe.cn/ijiami/encrypt>";

            var parameters2 = new Dictionary<string, string>();

            parameters2.Add("appid" , ""); //步骤①返回的appid

            parameters2.Add("key", appkey);//你申请的key

            parameters2.Add("dtype" , ""); //返回数据的格式,xml或json，默认json

            string result2 = sendPost(url2, parameters2, "get");

            JsonObject newObj2 = new JsonObject(result2);

            String errorCode2 = newObj2["error\_code"].Value;

            if (errorCode2 == "0")

            {

                Debug.WriteLine("成功");

                Debug.WriteLine(newObj2);

            }

            else

            {

                //Debug.WriteLine("失败");

                Debug.WriteLine(newObj2["error\_code"].Value+":"+newObj2["reason"].Value);

            }

            //3.③查询加密结果

            string url3 = "<http://op.juhe.cn/ijiami/result>";

            var parameters3 = new Dictionary<string, string>();

            parameters3.Add("appid" , ""); //步骤②返回的appid

            parameters3.Add("key", appkey);//你申请的key

            parameters3.Add("dtype" , ""); //返回数据的格式,xml或json，默认json

            string result3 = sendPost(url3, parameters3, "get");

            JsonObject newObj3 = new JsonObject(result3);

            String errorCode3 = newObj3["error\_code"].Value;

            if (errorCode3 == "0")

            {

                Debug.WriteLine("成功");

                Debug.WriteLine(newObj3);

            }

            else

            {

                //Debug.WriteLine("失败");

                Debug.WriteLine(newObj3["error\_code"].Value+":"+newObj3["reason"].Value);

            }

        }

        /// <summary>

        /// Http (GET/POST)

        /// </summary>

        /// <param name="url">请求URL</param>

        /// <param name="parameters">请求参数</param>

        /// <param name="method">请求方法</param>

        /// <returns>响应内容</returns>

        static string sendPost(string url, IDictionary<string, string> parameters, string method)

        {

            if (method.ToLower() == "post")

            {

                HttpWebRequest req = null;

                HttpWebResponse rsp = null;

                System.IO.Stream reqStream = null;

                try

                {

                    req = (HttpWebRequest)WebRequest.Create(url);

                    req.Method = method;

                    req.KeepAlive = false;

                    req.ProtocolVersion = HttpVersion.Version10;

                    req.Timeout = 5000;

                    req.ContentType = "application/x-www-form-urlencoded;charset=utf-8";

                    byte[] postData = Encoding.UTF8.GetBytes(BuildQuery(parameters, "utf8"));

                    reqStream = req.GetRequestStream();

                    reqStream.Write(postData, 0, postData.Length);

                    rsp = (HttpWebResponse)req.GetResponse();

                    Encoding encoding = Encoding.GetEncoding(rsp.CharacterSet);

                    return GetResponseAsString(rsp, encoding);

                }

                catch (Exception ex)

                {

                    return ex.Message;

                }

                finally

                {

                    if (reqStream != null) reqStream.Close();

                    if (rsp != null) rsp.Close();

                }

            }

            else

            {

                //创建请求

                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url + "?" + BuildQuery(parameters, "utf8"));

                //GET请求

                request.Method = "GET";

                request.ReadWriteTimeout = 5000;

                request.ContentType = "text/html;charset=UTF-8";

                HttpWebResponse response = (HttpWebResponse)request.GetResponse();

                Stream myResponseStream = response.GetResponseStream();

                StreamReader myStreamReader = new StreamReader(myResponseStream, Encoding.GetEncoding("utf-8"));

                //返回内容

                string retString = myStreamReader.ReadToEnd();

                return retString;

            }

        }

        /// <summary>

        /// 组装普通文本请求参数。

        /// </summary>

        /// <param name="parameters">Key-Value形式请求参数字典</param>

        /// <returns>URL编码后的请求数据</returns>

        static string BuildQuery(IDictionary<string, string> parameters, string encode)

        {

            StringBuilder postData = new StringBuilder();

            bool hasParam = false;

            IEnumerator<KeyValuePair<string, string>> dem = parameters.GetEnumerator();

            while (dem.MoveNext())

            {

                string name = dem.Current.Key;

                string value = dem.Current.Value;

                // 忽略参数名或参数值为空的参数

                if (!string.IsNullOrEmpty(name))//&& !string.IsNullOrEmpty(value)

                {

                    if (hasParam)

                    {

                        postData.Append("&");

                    }

                    postData.Append(name);

                    postData.Append("=");

                    if (encode == "gb2312")

                    {

                        postData.Append(HttpUtility.UrlEncode(value, Encoding.GetEncoding("gb2312")));

                    }

                    else if (encode == "utf8")

                    {

                        postData.Append(HttpUtility.UrlEncode(value, Encoding.UTF8));

                    }

                    else

                    {

                        postData.Append(value);

                    }

                    hasParam = true;

                }

            }

            return postData.ToString();

        }

        /// <summary>

        /// 把响应流转换为文本。

        /// </summary>

        /// <param name="rsp">响应流对象</param>

        /// <param name="encoding">编码方式</param>

        /// <returns>响应文本</returns>

        static string GetResponseAsString(HttpWebResponse rsp, Encoding encoding)

        {

            System.IO.Stream stream = null;

            StreamReader reader = null;

            try

            {

                // 以字符流的方式读取HTTP响应

                stream = rsp.GetResponseStream();

                reader = new StreamReader(stream, encoding);

                return reader.ReadToEnd();

            }

            finally

            {

                // 释放资源

                if (reader != null) reader.Close();

                if (stream != null) stream.Close();

                if (rsp != null) rsp.Close();

            }

        }

    }

}