OSS SDK for C++ 使用说明

baocai zhang
www.giser.net

1 简介

Aliyun OSS C++ SDK 使用 c++实现了 Aliyun OSS 提供的功能，主要包括 Bucket 的创建、删除、浏览。Object 的创建、删除浏览以及多点上传等功能，关于 OSS 提供的服务请参考 OSS API 说明文档。

2 主要内容

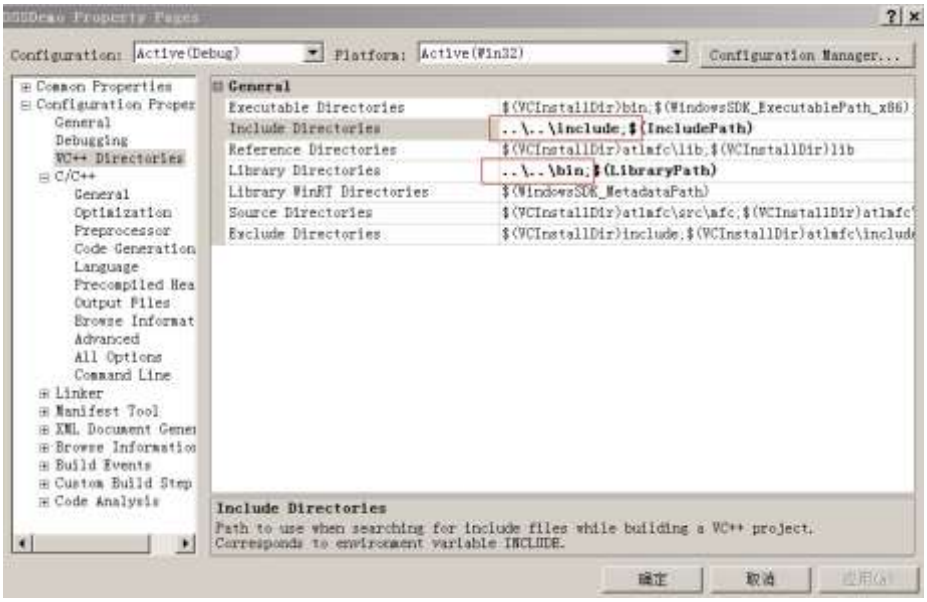
Aliyun OSS C++ SDK 主要包括源代码（BSD 协议开源）、说明文档、示例工程、帮助文档等。
该 SDK 开发环境为 vs2012，提供了 vs2012 的工程文件，目前只支持 windows 平台，暂不支持跨平台使用。
该 SDK 提供头文件和动态库和 lib 文件用于开发。
在线帮助文档：
http://osscppsdk.sinaapp.com/html/index.html

3 第三方库

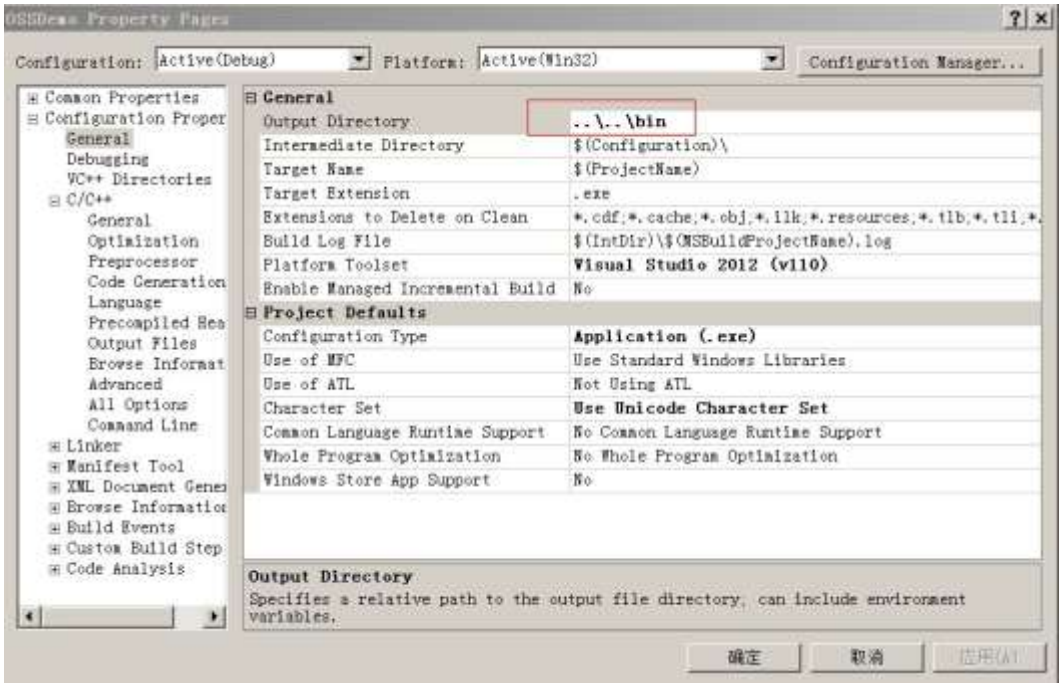
本 SDK 使用到的第三方类库包括以下部分：
Libcurl
Xmlparser

4 使用步骤

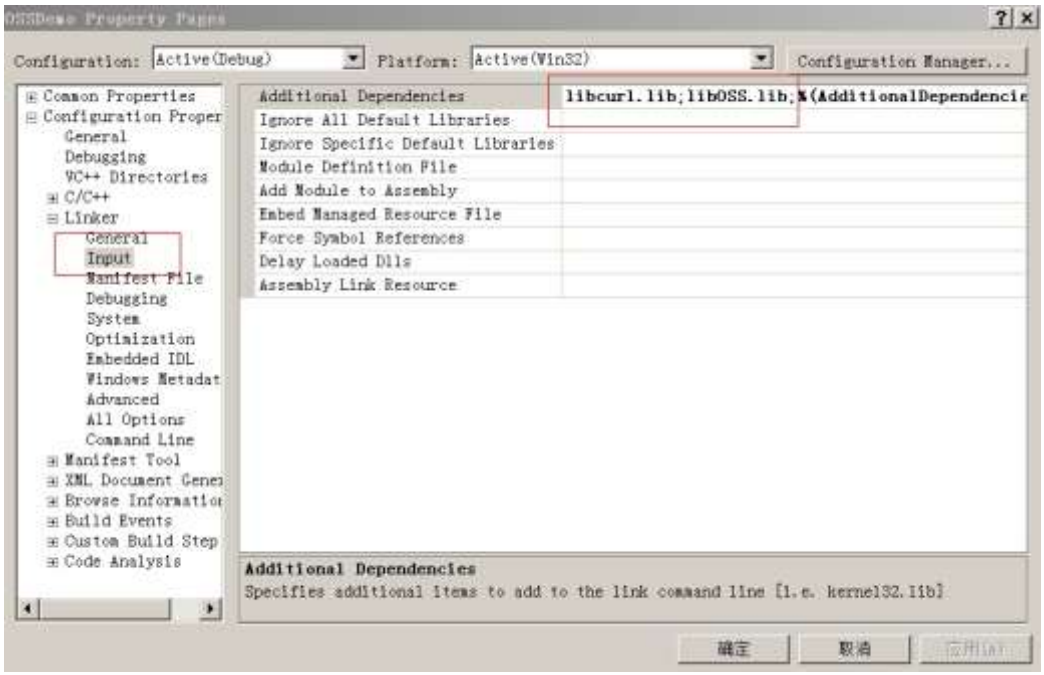
4.1 新建 vs2012 工程，将 vc include 目录设为 SDK 目录下的 include 目录，Library 目录设为 SDK 目录下的 bin 目录。



4.2 工程 Output 目录设为 SDK 目录下的 bin 目录。



4.3 在 Linker 属性下 Input 中添加 libcurl.lib 和 libOSS.lib



4.4 在代码中包含头文件 OSSClient.h

```
#include "OSSClient.h"
```

4.5 定义 OSSClient 对象，并使用 accessID 和 accessKey 初始化。

```
OSS::OSSClient *_ptClient = new  OSSClient(aID,akey,config);
```

4.6 使用 OSSClient 对 OSS 对象操作，OSS 对象提供了四类对象的操作函数，包括 Bucket，Object，MultiPart 和 ObjectGroup。OSSClient 还分别提供了同步和异步的方式进行调用。

4.6.1 同步方式调用

直接使用 OSSClient 对象调用即可。

```
_ptClient->ListBuckets(buckets);
```

4.6.2 异步方式调用

异步方式是 web 请求的主流请求方式，当请求返回的时候响应相关消息。

1）使用异步方式首先要实现代理类，该代理类必须继承 OSS::OSSClientCallback 这个抽象基类，并实现抽象基类中定义的回调方法。

```
class OSSClientTest: OSS::OSSClientCallback
{
public:
    OSSClientTest(void);
    ~OSSClientTest(void);
public:
    virtual  void OnCreateBucketComplete(OSSClient *client,Bucket &bucket);
    virtual  void OnCreateBucketFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnDeleteBucketComplete(OSSClient *client,string &bucketName);
    virtual  void OnDeleteBucketFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnListBucketsComplete(OSSClient *client,Buckets &buckets);
    virtual  void OnListBucketsFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnSetBucketACLComplete(OSSClient *client);
    virtual  void OnSetBucketACLFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnGetBucketACLComplete(OSSClient
*client,CannedAccessControlList &aclList);
    virtual  void OnGetBucketACLFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnIsBucketExistComplete(OSSClient *client,bool &isExist);
    virtual  void OnIsBucketExistFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnListObjectsComplete(OSSClient *client,ObjectListing
&objectListing);
    virtual  void OnListObjectsFailed(OSSClient *client,OSS_RESULTCODE code);
    //Object Op
    virtual  void OnPutObjectComplete(OSSClient *client,PutObjectResult &result);
    virtual  void OnPutObjectFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnFetchObjectComplete(OSSClient *client,OSSObject &object);
    virtual  void OnFetchObjectFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnFetchObjectToFileComplete(OSSClient *client,string
&fileName);
    virtual  void OnFetchObjectToFileFailed(OSSClient *client,OSS_RESULTCODE
code);
    virtual  void OnGetObjectMetadataComplete(OSSClient *client,ObjectMetadata
&objectMetadata);
    virtual  void OnGetObjectMetadataFailed(OSSClient *client,OSS_RESULTCODE
code);
    virtual  void OnCopyObjectComplete(OSSClient *client,CopyObjectResult
&result);
```

```
    virtual  void OnCopyObjectFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnDeleteObjectComplete(OSSClient *client);
    virtual  void OnDeleteObjectFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnDeleteMultipleObjectsComplete(OSSClient
*client,DeleteObjectsResult &result);
    virtual  void OnDeleteMultipleObjectsFailed(OSSClient *client,OSS_RESULTCODE
code);
    virtual  void OnPostObjectGroupComplete(OSSClient
*client,PostObjectGroupResult &result);
    virtual  void OnPostObjectGroupFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnGetObjectGroupIndexComplete(OSSClient
*client,FetchObjectGroupIndexResult &result);
    virtual  void OnGetObjectGroupIndexFailed(OSSClient *client,OSS_RESULTCODE
code);
    //multipart Op
    virtual  void OnUploadPartComplete(OSSClient *client,UploadPartResult
&result);
    virtual  void OnUploadPartFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnListPartsComplete(OSSClient *client,PartListing &result);
    virtual  void OnListPartsFailed(OSSClient *client,OSS_RESULTCODE code);
    virtual  void OnListMultipartUploadsComplete(OSSClient
*client,MultipartUploadListing &result);
    virtual  void OnListMultipartUploadsFailed(OSSClient *client,OSS_RESULTCODE
code);
    virtual  void OnInitiateMultipartUploadComplete(OSSClient
*client,InitiateMultipartUploadResult &result);
    virtual  void OnInitiateMultipartUploadFailed(OSSClient
*client,OSS_RESULTCODE code);
    virtual  void OnCompleteMultipartUploadComplete(OSSClient
*client,CompleteMultipartUploadResult &result);
    virtual  void OnCompleteMultipartUploadFailed(OSSClient
*client,OSS_RESULTCODE code);
    virtual  void OnAbortMultipartUploadComplete(OSSClient *client,string
&puloadId);
    virtual  void OnAbortMultipartUploadFailed(OSSClient *client,OSS_RESULTCODE
code);
    virtual  void OnNetworkFailed(OSSClient *client,OSS_RESULTCODE code);
};
```

2） 定义回调代理类

```
OSSClientTest *ptOSSClientTest = new OSSClientTest();
```

3）将代理类的指针赋给OSSClient 的delegate对象

```
 _ptClient->delegate =(OSSClientCallback *)ptOSSClientTest;
```

4） 调用以_Async 结尾的相关方法对 OSS 对象操作

```
_ptClient->ListBuckets_Async();
```