对象存储SDK文档（java）

# 第一：安装

1 准备java 环境（maven 项目）  
2 项目引入sdk  jar包：  
 <dependency>  
        <groupId>com.cloopen.sms</groupId>  自定义  
        <artifactId>sms</artifactId>    自定义  
        <version>1.0</version> 自定义  
        <scope>system</scope> system，类似provided，需要显式提供依赖的jar以后，Maven就不会在Repository中查找它  
        <systemPath>${basedir}/src\main\webapp\WEB-INF\lib\CCP\_REST\_SMS\_SDK\_JAVA\_v2.6.3r.jar</systemPath> 项目根目录下的lib文件夹下  
    </dependency>

<dependency>

<groupId>com.xinnet.smart</groupId>

<artifactId>smart-lib-base</artifactId>

<version>1.0</version>

<scope>system</scope>

<systemPath>${basedir}/src\main\webapp\WEB-INF\lib\smart-lib-base-v1.jar</systemPath>

</dependency>

<dependency>

<groupId>dom4j</groupId>

<artifactId>dom4j</artifactId>

<version>1.6.1</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.12</version>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.9</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.2</version>

</dependency>

pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>xinnetoss</groupId>

<artifactId>xinnetoss</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<dependency>

<groupId>com.xinnet.oss</groupId>

<artifactId>xinnet-sdk-oss</artifactId>

<version>1.0</version>

<scope>system</scope>

<systemPath>${basedir}/src\main\webapp\WEB-INF\lib\xinnet-sdk-oss-v1.jar</systemPath>

</dependency>

<dependency>

<groupId>com.xinnet.smart</groupId>

<artifactId>smart-lib-base</artifactId>

<version>1.0</version>

<scope>system</scope>

<systemPath>${basedir}/src\main\webapp\WEB-INF\lib\smart-lib-base-v1.jar</systemPath>

</dependency>

<dependency>

<groupId>dom4j</groupId>

<artifactId>dom4j</artifactId>

<version>1.6.1</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.12</version>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.9</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.2</version>

</dependency>

</dependencies>

</project>

3 项目引入测试实例(BucketOperationsSample，ObjectOperationsSample)进行测试。

BucketOperationsSample.java：

package com.xinnet.oss.samples;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.UUID;

import com.xinnet.oss.OssClient;

import com.xinnet.oss.model.Bucket;

import com.xinnet.smart.base.util.UTrace;

public class BucketOperationsSample {

private static String key = "CHKDE5ZSRD5CW9J6UBHC";

private static String secret = "PO6Hl1Lj7e9hp05ZB2m74HIsDBMFhqykCq6B5ETc";

private static String bucketName = "my-oss-bucket" + UUID.randomUUID();

private static String publicUrl = "http://172.16.110.15:6669/v1/storage/oss";

public static void main(String[] args) throws IOException {

try {

/\*

\*1 查询 Bucket 操作

\* get Bucket Operation

\*/

// doListBuckets();

/\*

\*2查询 Bucket 基本信息操作

\* get Bucket Operation

\*/

// doGetBucketInfo();

/\*

\*3 创建Bucket 操作

\* Put Bucket Operation

\*/

//doCreateBucketOperation();

/\*

\*4 删除 Bucket 操作

\* Delete Bucket Operation

\*/

// doDeleteBucketOperation();

/\*

\*5 删除 Bucket 操作

\* Delete Bucket Operation

\*/

//doDeleteBucketListOperation();

/\*

\* 6查询 Bucket 权限操作

\* get Bucket Operation

\*/

// doGetBucketAcl();

/\*

\*7 设置Bucket 权限操作

\* Put Bucket Operation

\*/

//doSetBucketAcl();

/\*

\*8判断Bucket 权限操作

\* Put Bucket Operation

\*/

doesBucketExistOperation();

} catch (Exception e) {

UTrace.trace(e);

}

}

//ossClient.listBuckets();ossClient.getBucketInfo("<yourBucketName>");

/\*\*

\* 1 列举所有的存储空间

\* 查询bucket

\*/

private static void doListBuckets() {

List<Bucket> bucketList = OssClient.listBuckets(key, secret, publicUrl);

for (Bucket bucket : bucketList) {

System.out.println(" - " + bucket.getName());

}

}

/\*\*

\* 2 获取bucket下对象信息

\* 查询bucket下对象信息

\* 存储空间的信息包括地域（Region或Location）、创建日期（CreationDate）、拥有者（Owner）、权限（Grants）等。

\*/

private static void doGetBucketInfo() {

// Bucket权限: private, public-read, public-read-write

OssClient.getBucketInfo(key, secret, publicUrl, "bucketlijufa");

}

/\*\*

\* 3 创建bucket

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）lijufabucketone3：bucket的名字

\* （第5个参数）private： Bucket权限（private, public-read, public-read-write）

\*/

private static void doCreateBucketOperation() {

// Bucket权限: private, public-read, public-read-write

OssClient.createBucket(key, secret, publicUrl, "lijufabucketone3", "private");

}

/\*\*

\*4 删除bucket

\*/

private static void doDeleteBucketOperation() {

OssClient.deleteBucket(key, secret, publicUrl, "lijufabucketone3");

}

/\*

\* 5.批量删除Bucket

\*

\*/

private static void doDeleteBucketListOperation() {

List<Bucket> bucketList = new ArrayList<Bucket>();

Bucket bucket = new Bucket();

bucket.setName("bucketone1");

bucketList.add(bucket);

OssClient.deleteBucketList(key, secret, publicUrl, bucketList);

}

/\*\*

\* 6 获取存储空间的访问权限。

\* // AccessControlList acl = ossClient.getBucketAcl("<yourBucketName>");

\* //ossAuth:public-read-write，public-read 和 private，

\* GET

\*/

private static void doGetBucketAcl() {

OssClient.getBucketAcl(key, secret, publicUrl, "bucketone1");

}

/\*\*

\* 7 设置存储空间的访问权限为私有。

\* //ossAuth:public-read-write，public-read 和 private，

\* //ossClient.setBucketAcl("<yourBucketName>", CannedAccessControlList.Private);

\* PUT

\*/

private static void doSetBucketAcl() {

OssClient.setBucketAcl(key, secret, publicUrl, "bucketone1", "private");

}

/\*

\* 8 判断bucket是否存在 True or False

\*ossClient.doesBucketExist("<yourBucketName>");

\*/

private static void doesBucketExistOperation() {

OssClient.doesBucketExist(key, secret, publicUrl, "bucketone1");

}

}

ObjectOperationsSample.java:

package com.xinnet.oss.samples;

import com.xinnet.oss.OssClient;

import com.xinnet.smart.base.util.UTrace;

import java.io.\*;

import java.util.UUID;

public class ObjectOperationsSample {

private static String key = "CHKDE5ZSRD5CW9J6UBHC";

private static String secret = "PO6Hl1Lj7e9hp05ZB2m74HIsDBMFhqykCq6B5ETc";

private static String bucketName = "my-oss-bucket" + UUID.randomUUID();

private static String publicUrl = "http://172.16.110.15:6669/v1/storage/oss";

public static void main(String[] args) throws IOException {

try {

//删除 Object 操作?上传字符串。?上上传文件流?

/\*

\*1 上传文件 操作

\* 上传字符串。

\* put Bucket Operation

\*/

//doPutObject() ;

/\*

\*上传文件 操作

\* 上传Byte数组

\* put Bucket Operation

\*/

// doPutObjectByte();

/\*

\* 2 上传本地文件 操作

\*

\* put Bucket Operation

\*/

//doPutObjectInputStream();

/\*

\* 3查询 Object 基本信息操作

\* get Object Operation

\*/

// doGetObject();

/\*

\*4 查询 Object

\* 得到文件原数据信息

\*/

// doGetObjectInfo();

/\*

\*5 设置object 权限 操作

\* Put Bucket Operation

\*/

// doSetObjectAcl();

/\*

\*6 获取object 权限 操作

\* Put Bucket Operation

\*/

// doGetObjectAcl();

/\*

\* 7 删除 Object 操作

\* Delete Bucket Operation

\*/

//doDeleteObjectOperation();

/\*

\* 8 生成以GET方法访问的签名URL，访客可以直接通过浏览器访问相关内容。

\*

\*/

//generatePresignedUrlOperation();

/\*

\* 9 判断object 是否存在。

\*

\*/

//doesObjectExistOperation();

/\*

\* 10 获取object , 并保存到本地

\*

\*/

doGetObjectToLocal();

} catch (Exception e) {

UTrace.trace(e);

}

}

/\*\*

\* // 1上传字符串。

\* ossClient.putObject(putObjectRequest);

\* 查询bucket

\*/

private static void doPutObject() {

// Bucket权限: private, public-read, public-read-write

OssClient.putObject(key, secret, publicUrl,"bucketone1","objectone5","private","hello object lijufa");

}

/\*\*

\* // 1-2上传Byte数组

\* ossClient.putObject(putObjectRequest);

\* 查询bucket

\*/

private static void doPutObjectByte() {

// Bucket权限: private, public-read, public-read-write

OssClient.putObject(key, secret, publicUrl,"bucketone1","objectone4","private","Hello OSS".getBytes());

}

/\*\*

\* // 1-3上传网络流。

\* ossClient.putObject(putObjectRequest);

\* 查询bucket

\*/

/\* private static void doPutObjectInputStream() {

// Bucket权限: private, public-read, public-read-write

InputStream inputStream = new URL("https://www.aliyun.com/").openStream();

OssClient.putObject(key, secret, "bucketone1","objectone3","private","Hello OSS".getBytes());

}\*/

/\*\*

\* // 1-4上传文件流

\* ossClient.putObject(putObjectRequest);

\* 查询bucket

\*/

/\* private static void doPutObjectInputStream() {

// Bucket权限: private, public-read, public-read-write

try {

InputStream inputStream = new FileInputStream("<yourlocalFile>");

} catch (FileNotFoundException e) {

e.printStackTrace();

}

OssClient.putObject(key, secret, "bucketone1","objectone3","private","Hello OSS".getBytes());

}\*/

/\*\*

\* 2上传本地文件

\* ossClient.putObject(putObjectRequest);

\*

\*/

private static void doPutObjectInputStream() {

// Bucket权限: private, public-read, public-read-write

try {

File file=new File("C:\\Users\\Administrator\\Desktop\\1.png");

InputStream inputStream = new FileInputStream(file);

OssClient.putObject(key, secret, publicUrl, "bucketone1","objectone8","private",inputStream);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

/\*\*

\*

\* 3获取object

\* get

\*

\*

\*/

private static void doGetObject() {

// Bucket权限: private, public-read, public-read-write

OssClient.getObject(key, secret, publicUrl,"bucketone1","objectone8");

}

/\*\*

\*

\*4 得到文件原数据信息

\*head

\*

\*/

private static void doGetObjectInfo() {

// Bucket权限: private, public-read, public-read-write

OssClient.getObjectInfo(key, secret, publicUrl,"bucketone1","objectone2");

}

/\*\*

\*5 删除Object

\* ossClient.deleteObject(bucketName, key);

\*/

private static void doDeleteObjectOperation() {

OssClient.deleteObject(key, secret, publicUrl,"bucketone1","objectone2");

}

/\*\*

\* 6 获取存Object的访问权限。

\* // AccessControlList acl = ossClient.getBucketAcl("<yourBucketName>");

\* GET

\*/

private static void doGetObjectAcl() {

OssClient.getObjectAcl(key, secret, publicUrl, "bucketone1","objectone2");

}

/\*\*

\* 7 设置Object的访问权限。

\* //ossAuth:public-read-write，public-read 和 private，

\* //ossClient.setBucketAcl("<yourBucketName>", CannedAccessControlList.Private);

\* PUT

\*/

private static void doSetObjectAcl() {

OssClient.setObjectAcl(key, secret, publicUrl, "bucketone1","objectone2","public-read-write");

}

/\*

\* 8 判断Object是否存在 True or False

\*ossClient.doesBucketExist("<yourBucketName>");

\*/

private static void doesObjectExistOperation() {

OssClient.doesObjectExist(key, secret, publicUrl,"bucketone1","objectone2");

}

/\*\*

\* 9

\* // 生成以GET方法访问的签名URL，访客可以直接通过浏览器访问相关内容。

\*

\*

\*/

private static void generatePresignedUrlOperation() {

OssClient.generatePresignedUrl(key, secret, publicUrl, "bucketone1","objectone8",600);

}

/\*\*

\*

\* 10 获取object , 并保存到本地

\* get

\*

\*

\*/

private static void doGetObjectToLocal(){

OssClient.getObjectToLocal(key, secret, publicUrl,"bucketone1","objectone8","C:\\Users\\Administrator\\Desktop\\");

}

}

4 仿照测试实例进行接口调用。

# 第二：快速入门

## 说明：

权限：包含三种权限，私有：BUCKET\_ACL\_PRIVATE，公共读：BUCKET\_ACL\_PUBLIC\_READ，公共读写：BUCKET\_ACL\_PUBLIC\_READ\_WRITE，默认是私有权限，子账号不能修改权限，只有大账号才能设置和获取权限

暂时不支持中文名称（bucket以及object）

## 关于bucket操作

存储空间: xinnet.Bucket类用于上传、下载、删除文件以及对存储空间进行各种配置。初始化类时，需要指定Endpoint.

存储空间（Bucket）是存储对象（Object）的容器。对象都隶属于存储空间

注：参考示例BucketOperationsSample.java 类。

注：*key：用户key，secret：用户秘钥；publicUrl：项目地址；所有接口都需要这3个参数，每个用户唯一。*

### 2.1 listBuckets（列举本用户所有的存储空间bucket）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | | 列举本用户所有的存储空间bucket | | |
| Method | | *listBuckets* | | |
| 调用 | | OssClient.*listBuckets*(*key*, *secret*,*publicUrl*) | | |
| 必选参数 | | | | |
| Name | | Type | | Description |
| *key* | | string | | *用户key；有厂商提供* |
| *secret* | | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | | string | | *项目url地址；有厂商提供* |
| 可选参数 | | | | |
| Name | | Type | | Description |
| Response JSON example | | | | |
| 返回list集合：List<Bucket> | | | | |
| Response Elements | | | | |
| Bucket对象属性： | | | | |
| name | String | | Bucket名字 | |

事例片段如下：

/\*\*

\* 1 列举所有的存储空间（查询bucket）

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\*

\*/

**private** **static** **void** doListBuckets() {

List<Bucket> bucketList=OssClient.*listBuckets*(*key*, *secret*,*publicUrl*);

**for** (Bucket bucket:bucketList) {

System.***out***.println(" - " + bucket.getName());

}

}

### 2.2 **getBucketInfo（**获取bucket下对象信息**）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能说明 | | 获取bucket下对象信息 | | | |
| Method | | *getBucketInfo* | | | |
| 调用 | | OssClient.*getBucketInfo*(*key*, *secret*, *publicUrl*, "bucketlijufa"); | | | |
| 必选参数 | | | | | |
| Name | | Type | | | Description |
| *key* | | string | | | *用户key；有厂商提供* |
| *secret* | | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | | string | | | *项目url地址；有厂商提供* |
| bucketName | | string | | | Bucket 名字 |
| 可选参数 | | | | | |
| Name | | Type | | | Description |
| Response JSON example | | | | | |
| {success:true,result:"<?xml version=\"1.0\" encoding=\"UTF-8\"?><ListBucketResult xmlns=\"https://test-www.pppcloud.cn/help.html\"><Name>bucketlijufa</Name><Prefix></Prefix><Marker></Marker><MaxKeys>1000</MaxKeys><IsTruncated>false</IsTruncated></ListBucketResult>",code:"success",message:"success"} | | | | | |
| Response Elements | | | | | |
| Name | | Type | | Description | |
| success | | Boolean | | true：成功 false：失败 | |
| code | | String | | Success：成功 false：失败 | |
| message | | String | | 描述信息 | |
| Bucket下对象信息： | | | | | |
| result | String | | Bucket下对象信息 | | |

事例片段如下：

/\*\*

\* 2 获取bucket下对象信息

\* 查询bucket下对象信息

\*（第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketlijufa：bucket的名字

\*/

**private** **static** **void** doGetBucketInfo() {

// Bucket权限: private, public-read, public-read-write

OssClient.*getBucketInfo*(*key*, *secret*, *publicUrl*, "bucketlijufa");

}

### 2.3 **createBucket（创建存储空间bucket）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | **创建存储空间bucket** | | |
| Method | *createBucket* | | |
| 调用 | OssClient.*createBucket*(*key*, *secret*,*publicUrl*,"lijufabucketone3","private"); | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| ossAuth | String | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| {success:true,code:"success",message:"success"} | | | |
| Response Elements | | | |
| Name | Type | Description | |
| success | Boolean | true：成功 false：失败 | |
| code | String | Success：成功 false：失败 | |
| message | String | 描述信息 | |

事例片段如下：

/\*\*

\* 3 创建bucket

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）lijufabucketone3：bucket的名字

\* （第5个参数）private： Bucket权限（private, public-read, public-read-write）

\*/

**private** **static** **void** doCreateBucketOperation() {

// Bucket权限: private, public-read, public-read-write

OssClient.*createBucket*(*key*, *secret*,*publicUrl*,"lijufabucketone3","private");

}

### 2.4 **deleteBucket（删除存储空间bucket）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | **删除存储空间bucket** | | |
| Method | *deleteBucket* | | |
| 调用 | OssClient.*deleteBucket*(*key*, *secret*, *publicUrl*, "lijufabucketone3"); | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| {success:true,code:"success",message:"success"} | | | |
| Response Elements | | | |
| Name | Type | Description | |
| success | Boolean | true：成功 false：失败 | |
| code | String | Success：成功 false：失败 | |
| message | String | 描述信息 | |

事例片段如下：

/\*\*

\*4 删除bucket

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）lijufabucketone3：bucket的名字

\*/

**private** **static** **void** doDeleteBucketOperation() {

OssClient.*deleteBucket*(*key*, *secret*, *publicUrl*, "lijufabucketone3");

}

### 2.5 **deleteBucketList（批量删除存储空间）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | **批量删除存储空间bucket** | | |
| Method | *deleteBucketList* | | |
| 调用 | OssClient.*deleteBucketList*(*key*, *secret*, *publicUrl*, bucketList); | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketList | ArrayList<Bucket> | | Bucket 名字的list集合 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| 返回内容：  “;luanchu1:删除成功;launchu2:删除成功;” | | | |
| Response Elements | | | |
| Name | Type | Description | |
| 返回内容为字符串 | | | |

事例片段如下：

/\*

\* 5.批量删除Bucket

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketList：bucket的名字的list集合

\*

\*/

**private** **static** **void** doDeleteBucketListOperation() {

List<Bucket> bucketList = **new** ArrayList<Bucket>();

Bucket bucket = **new** Bucket();

bucket.setName("luanchu1");

Bucket bucket2 = **new** Bucket();

bucket2.setName("launchu2");

bucketList.add(bucket);

bucketList.add(bucket2);

String str = OssClient.*deleteBucketList*(*key*, *secret*, *publicUrl*, bucketList);

System.***out***.println(str);

}

### 2.6 **getBucketAcl（**获取存储空间的访问权限**）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | 获取存储空间的访问权限 | | |
| Method | *getBucketAcl* | | |
| 调用 | OssClient.*getBucketAcl*(*key*, *secret*, *publicUrl*, "bucketone1"); | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| 返回内容：  “private” | | | |
| Response Elements | | | |
| Name | Type | Description | |
| 返回内容为字符串（注：Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写））） | | | |

事例片段如下：

/\*\*

\* 6 获取存储空间的访问权限。

\* // AccessControlList acl = ossClient.getBucketAcl("<yourBucketName>");

\* //ossAuth:public-read-write，public-read 和 private，

\* （第一个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\*

\*/

**private** **static** **void** doGetBucketAcl() {

OssClient.*getBucketAcl*(*key*, *secret*, *publicUrl*, "bucketone1");

}

### 2.7 **setBucketAcl（**设置存储空间的访问权限**）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | 设置存储空间的访问权限 | | |
| Method | *setBucketAcl* | | |
| 调用 | OssClient.*setBucketAcl*(*key*, *secret*, *publicUrl*, "bucketone1", "private"); | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| ossAuth | String | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| {success:true,code:"success",message:"success"} | | | |
| Response Elements | | | |
| Name | Type | Description | |
| success | Boolean | true：成功 false：失败 | |
| code | String | Success：成功 false：失败 | |
| message | String | 描述信息 | |

事例片段如下：

/\*\*

\* 7 设置存储空间的访问权限为私有。

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）private： Bucket权限（private, public-read, public-read-write）

\*/

**private** **static** **void** doSetBucketAcl() {

OssClient.*setBucketAcl*(*key*, *secret*, *publicUrl*, "bucketone1", "private");

}

### 2.8 **doesBucketExist（**判断bucket是否存在**）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | 判断bucket是否存在 | | |
| Method | *doesBucketExist* | | |
| 调用 | OssClient.*doesBucketExist*(*key*, *secret*, *publicUrl*, "bucketone1") | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| 返回内容：  True 或false | | | |
| Response Elements | | | |
| Name | Type | Description | |
| 返回内容为True 或false（boolean 类型） | | | |

事例片段如下：

/\*\*

\*

\* 8 判断bucket是否存在 True or False

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\*/

**private** **static** **void** doesBucketExistOperation() {

OssClient.*doesBucketExist*(*key*, *secret*, *publicUrl*, "bucketone1");

}

## 关于object操作

在OSS中，操作的基本数据单元是文件（Object）。OSS SDK提供了丰富的文件上传方式；

注：参考示例ObjectOperationsSample.java类。

注：*key：用户key，secret：用户秘钥；publicUrl：项目地址；所有接口都需要这3个参数，每个用户唯一。*

### 3.1 **putObject（往存储空间**bucket上传object对象，内容为字符串**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 往存储空间bucket上传object对象，内容为字符串 | | | |
| Method | *putObject* | | | |
| 调用 | OssClient.*putObject*(*key*, *secret*, *publicUrl*,"bucketone1","objectone5","private","hello object lijufa"); | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| ossAuth | String | | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| content | String | | | object的内容（格式为字符串） |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |

事例片段如下：

/\*\*

\* // 1上传字符串。

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone5：object的名字

\* （第6个参数）private： object权限（private, public-read, public-read-write）

\*（ 第7个参数）hello object lijufa： object的内容（格式为字符串）

\*/

**private** **static** **void** doPutObject() {

// Bucket权限: private, public-read, public-read-write

OssClient.*putObject*(*key*, *secret*, *publicUrl*,"bucketone1","objectone5","private","hello object lijufa");

}

### 3.2 **putObject** （**往存储空间**bucket上传object对象，内容为inputStream）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 往存储空间bucket上传object对象，内容为inputStream | | | |
| Method | *putObject* | | | |
| 调用 | OssClient.putObject(key, secret, publicUrl, "bucketone1","objectone8","private",inputStream) | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| ossAuth | String | | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| content | String | | | object的内容（文件流） |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |

事例片段如下：

/\*\*

\* 2上传本地文件

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone8：object的名字

\* （第6个参数）private： object权限（private, public-read, public-read-write）

\*（ 第7个参数）inputStream： object的内容（文件流）

\*/

**private** **static** **void** doPutObjectInputStream() {

// Bucket权限: private, public-read, public-read-write

**try** {

File file=**new** File("C:\\Users\\Administrator\\Desktop\\1.png");

InputStream inputStream = **new** FileInputStream(file);

OssClient.*putObject*(*key*, *secret*, *publicUrl*, "bucketone1","objectone8","private",inputStream);

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

}

### 3.3 **getObject（**获取object对象信息**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 获取object对象信息 | | | |
| Method | getObject | | | |
| 调用 | OssClient.getObject(key, secret, publicUrl,"bucketone1","objectone8") | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,result:"Data=hello object lijufa2",code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |
| result | | String | object对象信息 | |

事例片段如下：

/\*\*

\* 3获取object

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone8：object的名字

\*/

**private** **static** **void** doGetObject() {

OssClient.*getObject*(*key*, *secret*, *publicUrl*,"bucketone1","objectone8");

}

### 3.4 **getObjectInfo（**获取object对象原数据信息**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 获取object对象原数据信息 | | | |
| Method | getObjectInfo | | | |
| 调用 | OssClient.getObjectInfo(key, secret, publicUrl,"bucketone1","objectone2") | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,result:"{Accept-Ranges=[bytes], null=[HTTP/1.0 200 OK], Server=[Werkzeug/0.15.5 Python/3.6.8], Cache-Control=[no-cache], ETag=[\"b04ef08351f91456b942b62d30bf963c\"], Connection=[Keep-Alive], x-amz-request-id=[tx0000000000000000f956b-005e78694d-873895-default], Last-Modified=[Mon, 23 Mar 2020 06:01:47 GMT], Content-Length=[0], Date=[Mon, 23 Mar 2020 07:46:21 GMT], Content-Type=[binary/octet-stream]}",code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |
| result | | String | object对象原数据信息 | |

事例片段如下：

/\*\*

\*4 得到文件原数据信息

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone2：object的名字

\*

\*/

**private** **static** **void** doGetObjectInfo() {

// Bucket权限: private, public-read, public-read-write

OssClient.*getObjectInfo*(*key*, *secret*, *publicUrl*,"bucketone1","objectone2");

}

### 3.5 **deleteObject（**删除Object对象**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 删除Object对象 | | | |
| Method | deleteObject | | | |
| 调用 | OssClient.deleteObject(key, secret, publicUrl,"bucketone1","objectone2") | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |

事例片段如下：

/\*\*

\*5 删除Object

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone2：object的名字

\*

\*/

**private** **static** **void** doDeleteObjectOperation() {

OssClient.*deleteObject*(*key*, *secret*, *publicUrl*,"bucketone1","objectone2");

}

### 3.6 **getObjectAcl（**获取Object的访问权限**）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | 获取Object的访问权限 | | |
| Method | getObjectAcl | | |
| 调用 | OssClient.getObjectAcl(key, secret, publicUrl, "bucketone1","objectone2") | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| objectName | string | | object 名字 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| 返回内容：  “private” | | | |
| Response Elements | | | |
| Name | Type | Description | |
| 返回内容为字符串（注：Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写））） | | | |

事例片段如下：

/\*\*

\* 6 获取存Object的访问权限。

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone2：object的名字

\*/

**private** **static** **void** doGetObjectAcl() {

OssClient.*getObjectAcl*(*key*, *secret*, *publicUrl*, "bucketone1","objectone2");

}

### 3.7 **setObjectAcl（**设置Object的访问权限**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 设置Object的访问权限 | | | |
| Method | setObjectAcl | | | |
| 调用 | OssClient.setObjectAcl(key, secret, publicUrl, "bucketone1","objectone2","public-read-write") | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| ossAuth | String | | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |

事例片段如下：

/\*\*

\* 7 设置Object的访问权限。

\*

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone2：object的名字

\* 第6个参数）private： object权限（private, public-read, public-read-write）

\*/

**private** **static** **void** doSetObjectAcl() {

OssClient.*setObjectAcl*(*key*, *secret*, *publicUrl*, "bucketone1","objectone2","public-read-write");

}

### 3.8 **doesObjectExist（**判断Object是否存在**）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能说明 | 判断Object是否存在 | | |
| Method | doesObjectExist | | |
| 调用 | OssClient.doesObjectExist(key, secret, publicUrl,"bucketone1","objectone2") | | |
| 必选参数 | | | |
| Name | Type | | Description |
| *key* | string | | *用户key；有厂商提供* |
| *secret* | string | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | *项目url地址；有厂商提供* |
| bucketName | string | | Bucket 名字 |
| objectName | string | | object 名字 |
| 可选参数 | | | |
| Name | Type | | Description |
| Response JSON example | | | |
| 返回内容：  True 或false | | | |
| Response Elements | | | |
| Name | Type | Description | |
| 返回内容为True 或false（boolean 类型） | | | |

事例片段如下：

/\*

\* 8 判断Object是否存在 True or False

\* （第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone2：object的名字

\*/

**private** **static** **void** doesObjectExistOperation() {

OssClient.*doesObjectExist*(*key*, *secret*, *publicUrl*,"bucketone1","objectone2");

}

### 3.9 **generatePresignedUrl （**生成以GET方法访问的签名URL，访客可以直接通过浏览器访问相关内容**）**

|  |  |  |
| --- | --- | --- |
| 功能说明 | 生成以GET方法访问的签名URL，访客可以直接通过浏览器访问相关内容 | |
| Method | generatePresignedUrl | |
| 调用 | OssClient.generatePresignedUrl(key, secret, publicUrl, "bucketone1","objectone8",600) | |
| 必选参数 | | |
| Name | Type | Description |
| *key* | string | *用户key；有厂商提供* |
| *secret* | string | *用户秘钥；有厂商提供* |
| *publicUrl* | string | *项目url地址；有厂商提供* |
| bucketName | string | Bucket 名字 |
| objectName | string | object 名字 |
| expires | int | 访问有效期单位（秒） |
| 可选参数 | | |
| Name | Type | Description |
| Response JSON example | | |
| http://172.16.100.54:8060/bucketone1/objectone8?AWSAccessKeyId=CHKDE5ZSRD5CW9J6UBHC&Expires=1584952296&Signature=ma5NXe32XJljf62PgH1sobYCZ0k%3D | | |
| Response Elements | | |
| Name | Type | Description |
| 返回内容为字符串url | | |

事例片段如下：

/\*\*

\* 9

\* // 生成以GET方法访问的签名URL，访客可以直接通过浏览器访问相关内容。

\*（第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone8：object的名字

\* （第6个参数）600：访问有效期单位（秒）

\*/

**private** **static** **void** generatePresignedUrlOperation() {

OssClient.*generatePresignedUrl*(*key*, *secret*, *publicUrl*, "bucketone1","objectone8",600);

}

### 3.10 **getObjectToLocal（** 获取object对象内容 , 并保存到本地**）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能说明 | 获取object对象内容 , 并保存到本地 | | | |
| Method | getObjectToLocal | | | |
| 调用 | OssClient.getObjectToLocal(key, secret, publicUrl,"bucketone1","objectone8","C:\\Users\\Administrator\\Desktop\\") | | | |
| 必选参数 | | | | |
| Name | Type | | | Description |
| *key* | string | | | *用户key；有厂商提供* |
| *secret* | string | | | *用户秘钥；有厂商提供* |
| *publicUrl* | string | | | *项目url地址；有厂商提供* |
| bucketName | string | | | Bucket 名字 |
| objectName | string | | | object 名字 |
| ossAuth | String | | | Bucket权限（private（私有）, public-read（公共读）, public-read-write（公共读写）） |
| localUrl | String | | | Object要存的本地路径 |
| 可选参数 | | | | |
| Name | Type | | | Description |
| Response JSON example | | | | |
| {success:true,result:"C:\\Users\\Administrator\\Desktop\\objectone8",code:"success",message:"success"} | | | | |
| Response Elements | | | | |
| Name | Type | | Description | |
| success | | Boolean | true：成功 false：失败 | |
| code | | String | Success：成功 false：失败 | |
| message | | String | 描述信息 | |
| result | | String | Object保存的本地路径 | |

事例片段如下：

/\*\*

\*

\* 10 获取object对象内容 , 并保存到本地

\*

\*（第1个参数）key：

\* （第2个参数）secret：

\* （第3个参数）publicUrl：

\* （第4个参数）bucketone1：bucket的名字

\* （第5个参数）objectone8：object的名字

\* （第6个参数）C:\\Users\\Administrator\\Desktop\\：本地路径

\*/

**private** **static** **void** doGetObjectToLocal(){

OssClient.*getObjectToLocal*(*key*, *secret*, *publicUrl*,"bucketone1","objectone8","C:\\Users\\Administrator\\Desktop\\");

}