QINIU
七牛云存储

七牛云存储必备知识

@七牛云存储

QINIU
七牛云存储

主要内容

- 传统文件名和七牛云储存的文件名（key）的区别
- URL安全的Base64编码
- EncodedEntryURI格式
- 七牛云存储文件唯一性hash算法Etag
- AccessKey和SecretKey的用途
- 七牛用于签名请求内容的hmac_sha1散列算法
- 上传策略PutPolicy的用途
- 上传凭证，下载凭证，管理凭证的用途
- HTTP协议及浏览器相关知识

传统文件名和七牛文件名的区别

1. 传统文件是以目录结构来组织的，存在文件目录和文件名的概念

2. 七牛云存储的文件是以空间来组织的，不支持多级目录，文件名就是文件的key，文件的key就是文件名。比如空间qiniulab-public空间下面的image/2014/11/18/jemygraw.png文件，这里这个长长的image/2014/11/18/jemygraw.png就是这个文件的名字或者叫做文件key。

3. 为了支持传统目录结构的数据进行迁移，您可以为原始的文件名添加前缀，构成一个新的文件名。

4. 七牛的每一个空间下，每个文件的名字都是唯一的，不存在相同文件名的文件，所以一般也将文件名叫做文件的key，那文件本身当然就是这个key对应的value了。

URL安全的Base64编码

七牛的整个API服务建立在HTTP协议之上，开发者通过HTTP协议来传递和七牛云服务器之间进行数据交换的参数。这些参数为了能够尽可能多地支持不同的字符，必须寻找一种合适的编码方式，而不是传递原有的参数。

Base64编码是一种非常简单的数据编码方式，并且得到了各种编程语言的默认库或包支持。但是传统的Base64编码不适合用来在HTTP协议中传输数据，因为它会把加号(+)和斜杠(/)进行URL转义。为了避免这个问题，我们使用URL安全的Base64编码。

URL安全的Base64编码适用于以URL方式传递Base64编码结果的场景。该编码方式的基本过程是先将内容以标准Base64格式编码为字符串，然后检查该结果字符串，将字符串中的加号(+)换成中划线(-)，并且将斜杠(/)换成下划线(_)，同时尾部保持填充等号=。

http://developer.qiniu.com/docs/v6/api/overview/appendix.html#urlsafe-base64

URL安全的Base64编码

Python2.x语言支持

```
import base64
encodedStr=base64.urlsafe_b64encode('hello qiniu')
print(encodedStr)
aGVsbG8gcWluaXU=
decodedStr=base64.urlsafe_b64decode(encodedStr)
print(decodedStr)
hello qiniu
```

Python3.x语言支持

```
import base64
encodedStr=base64.urlsafe_b64encode('hello
qiniu'.encode('utf-8')).decode('utf-8')
print(encodedStr)
aGVsbG8gcWluaXU=
decodedStr=base64.urlsafe_b64decode(encodedStr.encode('
utf-8')).decode('utf-8')
print(decodedStr)
hello qiniu
```

URL安全的Base64编码

Golang语言支持

```
package main

import (
	"encoding/base64"
	"fmt"
)

func main() {
	var text = "hello qiniu"
	encodedStr := base64.URLEncoding.EncodeToString([]byte(text))
	fmt.Println(encodedStr)
	decodedBytes, _ := base64.URLEncoding.DecodeString(encodedStr)
	fmt.Println(string(decodedBytes))
}
```

URL安全的Base64编码

PHP语言支持

```
function Qiniu_Encode($str) // URLSafeBase64Encode
{
	$find = array('+', '/');
	$replace = array('-', '_');
	return str_replace($find, $replace, base64_encode($str));
}

function Qiniu_Decode($str)
{
	$find = array('-', '_');
	$replace = array('+', '/');
	return base64_decode(str_replace($find, $replace, $str));
}
```

EncodedEntryURI格式

这个EncodedEntryURI是七牛API中用来提供空间文件保存或管理的参数时的数据格式。Encoded和EntryURI可以分开理解。换句话说EncodedEntryURI就是用URL安全的Base64编码来对EntryURI进行编码之后得到的结果。

那EntryURI是什么，其实就是空间和文件key的组合，组合方式为“<空间名>:<文件key>”，就是用冒号（:）将空间和文件key组合起来。

写成伪代码就是：
entry = '<Bucket>:<Key>'
encodedEntryURI = urlsafe_base64_encode(entry)

注意上面entry里面的<>是表示这个内部的内容是参数，不属于实际值的一部分，实际代码中不应该添加。

http://developer.qiniu.com/docs/v6/api/reference/data-formats.html#data-format-encoded-entry-uri

七牛云存储文件唯一性hash算法

七牛对每一个上传到服务器的文件，都会根据文件的内容来计算一个表示文件唯一性的hash值，这个算法的原理和实现都是公开的。

这个hash值在文件上传时没有指定文件保存key的情况下，会被默认当作文件的key使用。

可以下载七牛提供的本地工具来计算文件的etag，详细的各种版本的语言实现，可以查看下面的Github链接。

```
jemy-MacBook:Temp2 jemy$ qetag app.mp4
luIcB7RTM6_2AHskBG7kKxoPgKs2
```

各语言版本的etag实现： https://github.com/qiniu/qetag

七牛云存储文件唯一性hash算法

各个操作系统之下qetag计算工具

Mac OS:
http://devtools.qiniu.io/qiniu-devtools-darwin_amd64-current.tar.gz
Linux 64bits:
http://devtools.qiniu.io/qiniu-devtools-linux_amd64-current.tar.gz
Linux 32bits:
http://devtools.qiniu.io/qiniu-devtools-linux_386-current.tar.gz
Linux ARMv6:
http://devtools.qiniu.io/qiniu-devtools-linux_arm-current.tar.gz
Windows 32bits:
http://devtools.qiniu.io/qiniu-devtools-windows_386-current.zip
Windows 64bits:
http://devtools.qiniu.io/qiniu-devtools-windows_amd64-current.zip

AccessKey和SecretKey的作用

AccessKey和SecretKey的作用

七牛的文件上传，下载，管理以及数据处理等服务都需要访问权限控制。而AccessKey和SecretKey就是这个访问权限控制体系中重要的一环。

AccessKey和SecretKey主要作用是配合hmac_sha1散列算法来对发送给七牛服务器的HTTP请求数据做签名，其中AccessKey和请求数据都会以明文在网络上传输给七牛服务器，七牛服务器再根据获得的AccessKey找到系统中对应的SecretKey，重新对HTTP请求数据做一次签名，然后用来和客户端传输过来的数据签名做比对，如果匹配，那么七牛服务器认为该HTTP请求是合法的，否则就是不合法的。

AccessKey和SecretKey是成对出现的，不可以交叉配对使用，建议客户通过七牛的后台系统定期更换AccessKey和SecretKey。

警告⚠：出于安全性的考虑，AccessKey和SecretKey只能保存在业务服务器端，不可以以任何文本或者二进制的方式出现在分发给客户使用的App或者网页端应用中。任何客户端所需要的凭证都必须由业务服务器颁发，否则因密钥泄漏导致的数据损毁后果自负。

七牛用于签名请求内容的hmac_sha1散列算法

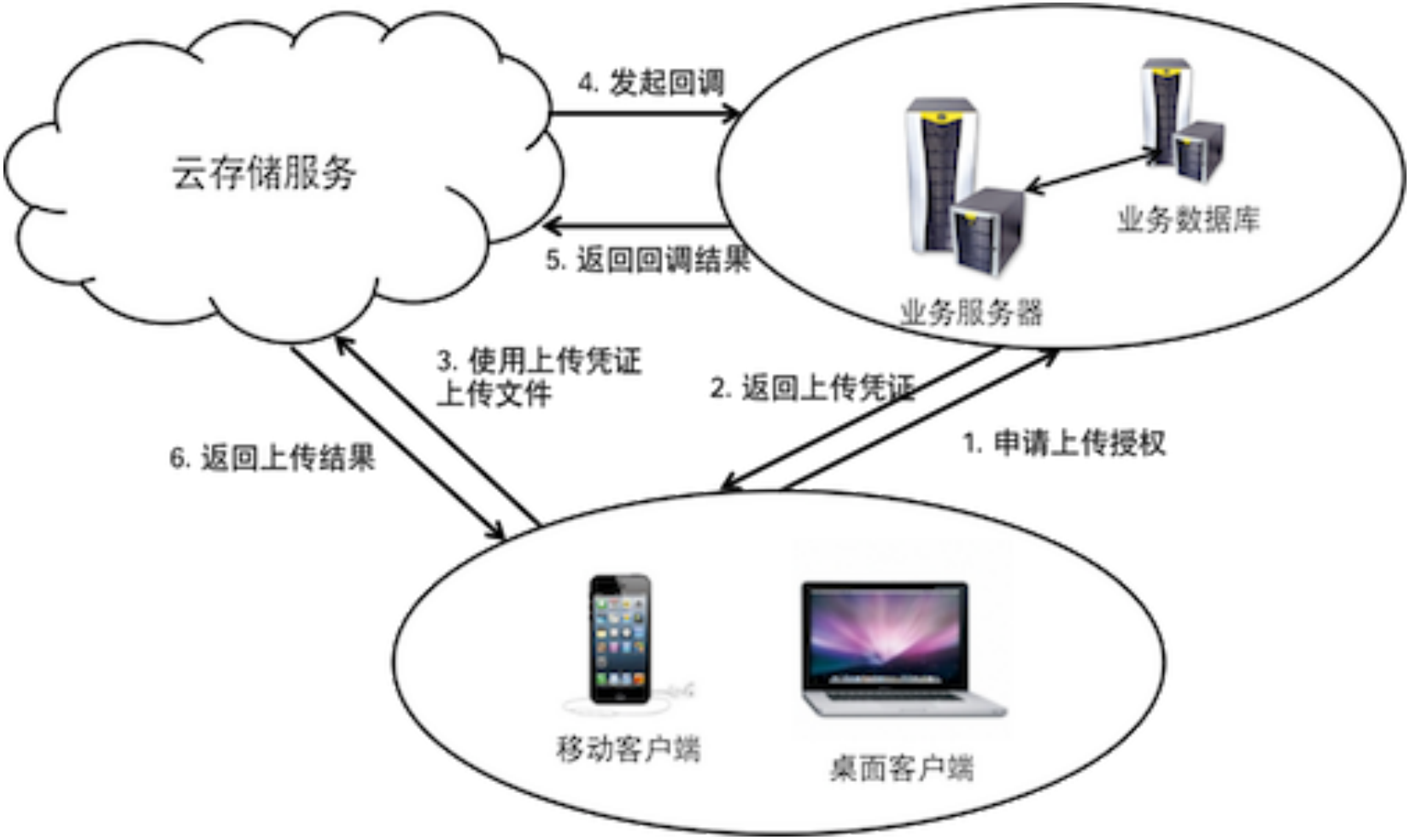
sha1是一种散列算法，一般用来对数据生成信息摘要。广泛应用于数据安全和加密领域。hmac_sha1是sha1的一种带密钥的版本，通过引入一个额外的SecretKey来结合消息内容做散列，增强散列结果被破解的难度。

七牛使用hmac_sha1散列算法来对请求内容做签名，然后把签名的结果和请求内容原文以及AccessKey一起传递给七牛服务器。七牛服务器根据客户端请求传递过来的AccessKey找到对应的SecretKey，然后使用和业务服务器同样的签名方式对请求内容原文再做一次签名，然后比对业务服务器的签名和七牛服务器的签名，如果两者一致，那么表示该请求是合法的。

具体来说，应用在上传凭证，私有资源下载凭证和资源管理凭证的生成以及回调请求合法性验证的过程中。

上传策略PutPolicy的用途

上传策略PutPolicy的用途



上传策略PutPolicy的用途

1. 七牛云存储服务器如何得知客户端上传来的文件存储在哪个空间？
2. 七牛云存储服务器该以什么名字保存客户端上传来的文件？
3. 客户端上传文件到七牛云存储服务器后，如何将上传的信息和业务服务器交互？
4. 客户端如何才能自定义设置七牛云存储服务器返回的回复内容？
5. 七牛云存储服务器如何得知客户端上传来的文件是经过授权的呢？
6. 七牛云存储服务器如何在文件上传完立即触发图片处理和视频处理等操作？
7. 七牛云存储服务器是否支持同名文件覆盖上传？

上传策略PutPolicy的用途

上传策略是一个参数集合，该集合包括：

1. 所要上传的文件的基本参数信息，
2. 和开发者业务服务器之间进行数据交换的方式（可选），
3. 文件上传之后的数据处理及持久化操作（可选），
4. 文件上传时进行类型检测，类型限制和大小限制的参数（可选）

上传策略是包含在上传Token里面的。七牛要求每一个进行上传的HTTP POST请求必须提供一个参数名为token的字符串。这个字符串由三个部分组成，分别是：

1. AccessKey（蓝色部分），
2. 利用hmac_sha1算法和SecretKey对PutPolicy进行散列计算然后把散列结果再进行URL安全Base64编码后的值（绿色部分），
3. 以URL安全的Base64进行编码的PutPolicy（红色部分）。彼此之间用冒号(:)连接。

所以，您看到的上传Token，是类似如下的格式：

pObK-5uirmOAtYGM705oxIco1m9xlqwONnYyLOoI:emuShanUKYxf1YdxiK8Qp4ohnxY=:eyJzY29wZSI6ImplbXlkZW1vYiIsImRlYWRsaW5lIjoxNDE2MzA3MDM4fQ==

上传策略PutPolicy的用途

上传策略PutPolicy的文档：
http://developer.qiniu.com/docs/v6/api/reference/security/put-policy.html

所要上传的文件的基本信息

scope	空间名称，或者是空间:文件key（这种方式用于覆盖上传）
deadline	上传授权的截至时间，Unix时间戳（以秒为单位）
insertOnly	当设置为非0值时，限定上传的文件为新增文件，这个值在使用覆盖上传的时候必须为0，否则不能覆盖已有文件。默认为非0值。
saveKey	在HTTP请求中没有提供POST参数key的情况下，会被当作文件key使用。
endUser	唯一属主标志符，可以用来表示用户id。比如@jemygraw这样的值。

上传策略PutPolicy的用途

和开发者业务服务器之间进行数据交换的方式一 - 重定向：

returnUrl	文件上传完之后七牛服务器重定向的地址
returnBody	文件上传完之后七牛服务器重定向到returnUrl时，携带的url参数，即upload_ret的值。

和开发者业务服务器之间进行数据交换的方式二 - 回调业务服务器：

callbackUrl	七牛服务器回调业务服务器的地址
callbackHost	七牛服务器回调业务服务器的主机地址，一般建议设置为ip地址，以提高回调速度
callbackBody	七牛服务器回调业务服务器时的POST请求所携带的请求Body。

上传策略PutPolicy的用途

文件上传之后的持久化操作

persistentOps	资源上传成功后触发的持久化处理指令，可以支持多个指令集合。彼此用分号(;)隔开。
persistentNotifyUrl	资源上传完后持久化处理之后通知业务服务器的地址。
persistentPipeline	进行资源预处理的队列名称，指定为空的时候，表示使用公用队列，公用队列处理速度不如专用队列快。

在PutPolicy里面指定文件上传之后的持久化处理操作是七牛支持的文件持久化操作中的一种自动触发的方式。

上传策略PutPolicy的用途

上传策略是一个定义文件上传客户端，业务服务器和七牛服务器彼此进行信息交互的参数集合。

上传策略里面可以设置上传策略的有效期，杜绝任何访问权限控制问题。

上传策略的参数是在开发者业务服务器组装，并以签名的Token方式提供给上传客户端的。

上传策略和AccessKey是明文传输给七牛服务器的。

七牛服务器对上传策略的校验方式就是按照和业务服务器相同的方式重新计算上传策略得到Token和上传客户端传递过来的Token进行比对。

上传策略仅仅用在上传文件客户端，业务服务器，七牛服务器彼此之间对上传文件进行信息交互的这个业务场景中。

上传凭证，下载凭证，管理凭证

上传凭证：

适用于文件上传服务的授权。
http://developer.qiniu.com/docs/v6/api/reference/security/upload-token.html

下载凭证

适用于私有空间文件访问的授权。
http://developer.qiniu.com/docs/v6/api/reference/security/download-token.html

管理凭证

适用于空间文件管理操作的授权。
http://developer.qiniu.com/docs/v6/api/reference/security/access-token.html

上传凭证，下载凭证，管理凭证

QINIU 七牛云存储

上传凭证

上传凭证是业务服务器根据客户端文件上传时所采用的上传策略，使用七牛API中规定的算法生成并颁发给上传客户端用来进行文件上传的授权字符串。该授权字符串会被文件上传端在上传文件时，以HTTP POST请求参数的方式传递给七牛云存储服务器。

具体操作是：

1. 业务服务器根据文件上传端的实际需求组装PutPolicy的参数
2. 将组装好的PutPolicy转换为对应的JSON字符串，称为putPolicyData
3. 使用hmac_sha1算法，结合SecretKey对putPolicyData做散列计算，得到结果signedPutPolicyData
4. 使用URL安全的Base64编码将signedPutPolicyData进行编码，得到结果encodedSignedPutPolicyData
5. 使用URL安全的Base64编码对已是PutPolicy的JSON字符串的putPolicyData进行编码得到encodedPutPolicyData
6. 将AccessKey，encodedSignedPutPolicyData，encodedPutPolicyData用冒号(:)连接起来得到upload Token。

上传凭证，下载凭证，管理凭证

上传凭证伪代码

putPolicyData=putPolicy.toJSON()
signedPutPolicyData=hmacSha1(putPolicyData, *secretKey*)
encodedSignedPutPolicyData=urlsafeBase64Encode(signedPutPolicyData)
encodedPutPolicyData=urlsafeBase64Encode(putPolicyData)
uploadToken=*accessKey*+":"+encodedSignedPutPolicyData
+":"+encodedPutPolicyData

1. 上面伪代码里面的accessKey和secretKey可以从你的后台获得，必须放在业务服务器端。

2. 上传凭证的生成必须放在业务服务器，建议每上传一个文件，就新创建一个上传凭证。

3. PutPolicy在进行hmac_sha1散列计算之前，必须转为JSON字符串，另外PutPolicy里面的参数如果采用默认值或者空值的话，就不要设置。

上传凭证，下载凭证，管理凭证

下载凭证：

下载凭证是业务服务器颁发给终端客户用来对七牛云存储服务器上私有空间中的文件进行访问的授权字符串。该授权字符串会附加在终端客户访问七牛云存储服务器上私有空间的文件链接的后面。对七牛云存储服务器上的公开空间中文件访问不需要下载凭证。下载凭证涉及到授权有效期的问题。

具体的操作是:

1. 对私有空间中的文件，先按照访问公开空间中的文件的方式构建一个Base Url，
2. 然后为这个Base Url设置一个授权过期时间戳，这个过期时间戳即是一个Unix时间戳，单位是秒，而且必须是一个未来的时间戳，否则将无法访问资源。这个过期时间戳的参数名为e，以URL参数的方式附加，形成一个新的URL，称为ExpireUrl。
3. 然后使用hmac_sha1算法，结合SecretKey对这个ExpireUrl进行散列计算得到结果signedExpireUrl，然后把signedExpireUrl用URL安全的Base64编码进行编码得到encodedSignedExpireUrl。
4. 将AccessKey和encodedSignedExpireUrl用冒号(:)连接起来得到downloadToken
5. 将downloadToken以参数名为token的URL参数方式接在ExpireUrl后面，得到最终的私有空间资源下载的新链接newUrl。

上传凭证，下载凭证，管理凭证

下载凭证伪代码

```
expireInSeconds=3600
now = datetime.now().toTimestampInSeconds()
expireTimestamp= now + expireInSeconds
if (baseUrl.hasQueryParams())
{
    expireUrl=baseUrl+"&e="+expireTimestamp
}
else
{
    expireUrl=baseUrl+"?e="+expireTimestamp
}
signedExpireUrl=hmacSha1(expireUrl, secretKey)
encodedSignedExpireUrl=urlsafeBase64Encode(signedExpireUrl)
downloadToken=accessKey+":"+encodedSignedExpireUrl
newUrl=expireUrl+"&token="+downloadToken
```

上传凭证，下载凭证，管理凭证

下载凭证

1. 上面伪代码里面的accessKey和secretKey可以从你的后台获得，必须放在业务服务器端。

2. 下载凭证为了保证授权时间的有效性，需要业务服务器和七牛云存储服务器的时间一致。七牛云存储服务器的时间设置为CST，即UTC+8时间。和终端用户时间无关。

3. 下载凭证必须由业务服务器生成并颁发给终端用户。

4. 私有空间文件的下载凭证过期之后必须重新从业务服务器获取新的下载凭证。

5. 不建议客户将私有空间的授权时间设得太久，这样就等同于公开空间了。

上传凭证，下载凭证，管理凭证

管理凭证：

管理凭证是业务服务器要求对存储在七牛云存储服务器上面的数据进行管理操作时，必须提供的权限验证字符串。该权限验证字符串以HTTP的请求Header的方式传递给七牛云存储服务器。Header的名字叫做Authorization。

具体操作是：

1. 根据七牛云存储数据管理API的规范，构建HTTP管理请求Request。
2. 获取Request的URL里面的Path信息和Query查询串，构成一个新的待签名字符串。如果Query查询串为空，那么就只使用Path。这样这一步的操作就是得到Path或者Path?Query字符串。
3. 获取Request的请求Body，如果Body为空就设置为空字符串。
4. 将第二步中的字符串和第三步中的字符串用“\n”连接起来，获得待签名字符串Path\nBody或者Path?Query\nBody。
5. 使用hmac_sha1算法，结合SecretKey对第4步中的待签名字符串进行散列计算得到散列结果
6. 对第5步里面的散列结果进行URL安全的Base64编码获得一个encodedSignData。
7. 将AccessKey和第6步里面的encodedSignData用冒号(:)连接起来得到管理凭证Access Token。

上传凭证，下载凭证，管理凭证

管理凭证伪代码

```
requestURL=httpRequest.RequestURL
requestPath=requestURL.Path
requestQuery=requestURL.QueryString
if(requestQuery!="")
{
    requestPath+="?"+requestQuery
}
requestBody=httpRequest.RequestBody
signString=requestPath+"\n"+requestBody
signedData=hmacSha1(signString, secretKey)
encodedSignedData=urlsafeBase64Encode(signedData)
accessToken=accessKey+":"+encodedSignedData
```

上传凭证，下载凭证，管理凭证

管理凭证

1. 上面伪代码里面的accessKey和secretKey可以从你的后台获得，必须放在业务服务器端。

2. 空间资源管理操作建议只通过业务服务器去管理，也就是说这个管理凭证只建议在业务服务器和七牛服务器之间传递。

3. 管理凭证只能由业务服务器生成，供业务服务器管理存储在七牛云存储服务器上面的数据所用，不能颁发给客户端，否则造成的数据损毁，责任自负。

4. 管理凭证可以管理公开空间和私有空间的文件，包括获取文件的基本信息，拷贝文件，移动文件，删除文件，列举文件等等。另外管理凭证还可以用来授权七牛服务器数据处理持久化。

上传凭证，下载凭证，管理凭证

管中窥豹

从我们这里讲到的三种凭证我们可以大致了解七牛云存储提供的基本服务。

上传凭证：
对应于文件上传。

下载凭证：
对应于文件下载。

管理凭证：
对应于资源管理和手动数据处理持久化。

HTTP协议及相关知识

基本支持

七牛云存储目前使用的是标准的HTTP的协议来进行文件的上传，下载，和管理。支持的请求方式主要为GET和POST。其中GET请求一般用来获取不可变的信息，比如下载文件，获取文件基本信息等。POST方式一般用来支持对七牛云存储服务器上数据做改动的请求，比如文件上传，文件重命名，文件移动，文件删除等等。

HTTP请求包括请求的头部（Header）和请求的内容（Body）。这两者用"\r\n"分隔开。

七牛云存储的文件上传请求组织格式遵循RFC2388标准，就是和一般用浏览器上传文件时，浏览器组织请求内容的方式相同。所以你也可以直接通过浏览器上传文件到七牛云存储服务器。

HTTP协议及相关知识

请求的头部参数Content-Type

multipart/form-data; boundary=----WebKitFormBoundaryfwK8W37DaWs4pfzV
这是上传文件是HTTP请求所需要设置的头部信息，如果是浏览器表单提交上传请求，一般由浏览器自行设置。

application/x-www-form-urlencoded; charset=UTF-8
这个是资源管理和手动调用数据处理持久化操作时设置的头部信息，表示请求的数据内容是以url编码的方式提供的。

回复的头部参数Content-Type

application/json
七牛返回给终端用户或者业务服务器的数据类型，表示该数据的内容是JSON格式。

HTTP协议及浏览器相关知识

浏览器实现了HTTP的协议，支持GET和POST请求。

浏览器使用HTTP协议和远程服务器进行通讯，遵守标准的HTTP协议格式。

HTTP请求包含请求头部（Header）和请求实体（Body）。

浏览器不是唯一能够发送HTTP请求的东西。

主流编程语言都提供了支持HTTP协议的库或者包。所以您可以使用这些库或包提供的功能来和七牛云存储服务器进行通讯。

浏览器可以使用ajax来和远程服务器进行通讯，同样遵循标准HTTP协议，但是可以不用刷新页面。

页面是浏览器呈现数据的方式，后台数据通讯是标准HTTP的协议。

HTTP协议及浏览器相关知识

1. 演示GET请求

了解GET请求的请求内容和回复内容格式。

2. 演示非文件上传POST请求

了解非文件上传的POST请求内容和回复内容格式

3. 演示文件上传POST请求

了解文件上传的POST请求内容和回复内容格式

QINIU
七牛云存储

成为最专业的云存储服务提供商！