**BFC上传管理器SDK**

BFC小组 制定

2015-02-10

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **发布日期** | **主要内容** | **负责人** |
| V1.0 | 2014-12-11 | 首版上传器接口 | 邓俊辉 |
| V1.00 | 2015-02-10 | 首版Component组件上传器发布 | 谢钰铭 |

目录

[目录 3](#_Toc3680)

[一、 版本信息 4](#_Toc7814)

[二、 适用机型 4](#_Toc8869)

[三、 依赖环境 4](#_Toc29250)

[四、 原理简介 4](#_Toc17010)

[五、 具体方法 5](#_Toc23545)

[(一) Manifest中需要一下定义 5](#_Toc23438)

[(二) 基本操作 6](#_Toc30980)

[1、 添加HTTP多文件上传任务（批量） 6](#_Toc2543)

[2、 添加FTP任务（批量） 7](#_Toc8569)

[3、 添加CLOUD多文件任务（批量） 8](#_Toc10598)

[4、 删除任务（批量） 9](#_Toc23986)

[5、 重新上传（批量） 9](#_Toc30089)

[6、 刷新任务状态（批量） 9](#_Toc1280)

[7、 查询上传任务 9](#_Toc12820)

[(a) 通过ID条件查询 9](#_Toc14113)

[(b) 通过扩展字段条件查询 9](#_Toc26384)

[(c) 通过多种条件组合查询 9](#_Toc11821)

[(三) 状态监测(Observer) 10](#_Toc22408)

[(四) 状态监测(BroadcastReceiver) 11](#_Toc5393)

[附录一 接口 13](#_Toc20117)

1. 版本信息

版本号：V1.00

1. 适用机型

Android通用机型，包括第三方android手机平台

1. 依赖JAR包

* 需要将BFC\_ Component \_UploadManager\_VX.XX.jar 放在工程libs目录下；
* 需要将gson-X.X.X.jar 放在工程的libs目录中；
* 需要将BFC\_Utils.jar 放在工程libs目录中；
* 需要将uploadserverentity.jar放在工程libs目录下；
* 需要将httpmime-4.1.3.jar 放在工程的libs目录中；
* 用到FTP上传相关的，需要将commons-net-3.3.jar 放在工程libs目录中；

1. 原理简介

* UploadManager为应用维护一个单独的上传服务与记录上传任务的SQL数据库，通过UploadProvider开放出来，并根据任务状态完成各种上传操作；
* 应用通过BFC\_ Component \_UploadManager\_VX.XX.jar 提供的增删改查接口，操作上传数据库，完成对上传任务的操作；
* 应用通过自定义ContentObserver监听上传数据库的变化，来实时更新view层上传进度、状态的显示；
* 应用也可以通过上传完成广播来监听上传结束；

1. 具体方法

* 基本操作

添加、删除、查询等操作通过UploadController功能类来实现。具体操作如下：

1. **添加HTTP多文件上传任务（批量）**

//第一步：生成一个Http上传任务

ITasktask = new HttpTask(Url);

//第二步：设置上传任务的相关属性

//设置上传的通知栏标题名

task.setFileName(fileName);

//添加需要上传的文件

task.addFileBody(key1 , filePath1);

task.addFileBody(key2 , filePath2);

......

//添加需要上传的键值对

task.addStringBody(key1, value1);

task.addStringBody(key2, value2);

......

//设置任务的扩展字段信息

task.putExtra(key1 , value1);

task.putExtra(key2, value2);

......

//设置通知栏显示效果

task.setNotificationVisibility(Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLET ED);

//第三步：添加上传任务

mUploadController.addTask(task);

1. **添加FTP任务（批量）**

//第一步：生成一个FTP上传任务

ITasktask = new FtpTask("172.28.15.232", 638, "BBKFTPClient", "lgG57R9N");

//第二步：设置上传任务的属性

//设置远程服务器的路径

task.setRemoteFilePath("TEST1//TEST2//");

//设置远程服务器的文件名

task.setRemoteFileName("TEST.apk");

//设置要上传的本地文件名

task.setFileName(fileName);

//设置要上传的文件路径

task.setFilePath(filePath);

//设置通知栏显示效果

task.setNotificationVisibility(Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLET ED);

//第三步：添加上传任务

mUploadController.addTask(task);

1. **添加CLOUD多文件任务（批量）**

//第一步：生成一个MultiCloudTask上传任务

ITasktask = new MultiCloudTask();

//第二步：设置上传任务的属性

//设置上传通知栏标题名

task.setFileName(fileName);

//添加上传的文件

task.addFile(fileName1 , filePath1);

task.addFile(fileName2 , filePath2);

......

//设置是否进行服务器文件覆盖

task.setOverWrite(false);

//设置扩展字段的扩展信息

task.putExtra(key1 , value1);

task.putExtra(key2 , value2);

......

//设置通知栏显示效果

task.setNotificationVisibility(Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLET ED);

//第三步：添加上传任务

mUploadController.addTask(task);

1. **删除任务（批量）**

//通过添加任务后的返回ID或者上传任务进行删除

mUploadController.deleteTask(task);

1. **重新上传（批量）**

//通过添加任务后的返回ID或者上传任务进行重新上传

mUploadController.reloadTask(task);

1. **刷新任务状态（批量）**

//通过传入参数ITask的列表，从数据库中读取最新状态，更新ITask列表

//一般用于Adapter中刷新数据

mUploadController.refreshData(tasks);

1. **查询上传任务**
2. 通过ID条件查询

//直接通过ID来查询

ITask task = mUploadController.getTaskById(id);

1. 通过扩展字段条件查询

//第一步：添加任务时，在扩展字段处加入扩展字段如：KEY : TEST , VALUE : 123

task.addExtras(“TEST”, “123”);

//第二步：通过这个扩展字段来查询这个上传任务

ArrayList<ITask>taskList =mUploadController.getTaskByExtras(query , new String[]{“TEST”} , new String[]{“123”});

1. 通过多种条件组合查询

//第一步：先创建一个查询用的Query

Query query = new query();

//第二步：设置查询的条件

query.setFilterById(id);

query.setFilterByStatus(UploadManager.STATUS\_FAILED);

query.addFilterByExtras(“TEST”, “123”);

//第三步：查询

ArrayList<ITask>taskList = mUploadController.getTask(query);

* 状态监测(Observer)

通过监听数据库发生改变的方法来实现状态监测

第一步：先定义一个ContentObserver如：

class UploadChangeObserver extends ContentObserver {

public UploadChangeObserver ( ) {

super(new Handler());

}

@Override

public void onChange ( boolean selfChange ) {

//通过指定的ID查询任务，或者通过固定的KEY:VALUE查询任务列表

ArrayList<ITask> taskList=mUpnloadController.

getTaskByExtras(new String[]{“module”} , new String[]{“智能答疑”});

for(ITask task : taskList){

switch(task.getState()){

case UploadManager.STATUS\_FAILED:

break;

case UploadManager.STATUS\_SUCCESSFUL:

break;

case UploadManager.STATUS\_PAUSED:

break;

case UploadManager.STATUS\_PENDING:

case UploadManager.STATUS\_RUNNING:

break;

default:break;

}

}

}

第二步：生成一个监听对象

UploadChangeObserver mUploadObserver = new UploadChangeObserver ();

第三步：设置监听，在Activity 的 onResume中

getContentResolver().registerContentObserver(Impl.ALL\_UPLOADS\_CONTENT\_URI, true, mUploadObserver );

第四步：取消监听，在Activity 的onPause 中

getContentResolver().unregisterContentObserver(mUploadObserver );

* 状态监测(BroadcastReceiver)

**方式1：静态注册**

AndroidManifest.xml中用标签生命注册，并在标签内用标签设置过滤器。

<receiver android:name="myRecevice">

<intent-filter>

<action android:name="android.intent.action.UPLOAD\_COMPLETE" />

</intent-filter>

</receiver>

**方式2：动态注册**

IntentFilter filter= new IntentFilter();

filter.addAction(DownloadConstants.ACTION\_UPLOAD\_COMPLETE);

//在onStart中注册

registerReceiver(myRecevice,filter);

//onStop中取消

unregisterReceiver(myRecevice);

**广播信息获取**

public void onReceive(Context context, Intent intent) {

//获取上传任务ID

long id = intent.getLongExtra(UploadConstants.EXTRA\_UPLOAD\_ID, -1);

//获取上传任务类型

int taskType = intent.getIntExtra(Impl.COLUMN\_TASK\_TYPE, -1);

//获取上传任务结束后的输出

String output = intent.getStringExtra(Impl.COLUMN\_OUTPUT);

......

}

附录一 接口

详细接口文档；

/\*\*

\* 上传基础接口

\*/

public interface IController {

/\*\*

\* 添加任务

\*

\* @param pTasks

\* 单任务或任务数组皆可

\* @return 成功添加的数量

\*/

int addTask(ITask... pTasks);

/\*\*

\* 添加任务

\*

\* @param pTasks

\* 单任务或任务数组皆可

\* @return 成功添加的数量

\*/

int addTask(ArrayList<ITask> pTasks) ;

/\*\*

\* 删除任务

\*

\* @param ids

\* 通过 id删除任务，单任务或任务数组皆可

\* @return 成功删除的数量

\*/

int deleteTask(long... ids);

/\*\*

\* 删除任务

\*

\* @param pTasks

\* 单任务或任务数组皆可

\* @return 成功删除的数量

\*/

int deleteTask(ITask... pTasks);

/\*\*

\* 刷新任务状态

\*

\* @param pTasks

\* 单任务或任务数组皆可

\* @return 成功刷新的数量

\*/

int refreshData(ITask... pTasks);

/\*\*

\* 查询任务

\*

\* @param id

\* 根据id查询

\* @return 上传任务

\*/

ITask getTaskById(long id);

/\*\*

\* 查询任务

\* @param query 根据自定义query查询符合条件的上传任务

\* @return 符合query的任务列表

\*/

ArrayList<ITask> getTask(Query query);

/\*\*

\* 查询任务 根据扩展字段与query

\* @param query

\* @param keys 键数组

\* @param values 值数组

\* @return 相应的任务列表

\*/

ArrayList<ITask> getTaskByExtras(Query query, String[] keys, String[] values);

/\*\*

\* 重新上传 根据ids

\* @param ids id

\* @return 重新上传成功的数量

\*/

int reloadTask(long... ids);

/\*\*

\* 重新上传 根据上传任务

\* @param pTasks 重新上传的任务数组

\* @return 重新上传成功的数量

\*/

int reloadTask(ITask... pTasks);

}