**BFC下载管理器SDK**

BFC小组 制定

2014-07-31

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **发布日期** | **主要内容** | **负责人** |
| V1.0 | 2014-8-18 | 首版下载内容 | 李超 |
| - | 内部版本 | 维护文档，增加了对BFCUtil包依赖的描述 | 李超 |
|  | 整合 | 核对SDK文档与实际接口 | 谢钰铭 |
|  | 修定 | 发布Component下载组件 | 谢钰铭 |

目录

目录 3

1、 版本信息 4

2、 适用机型 4

3、 依赖JAR包 4

4、 原理简介 4

5、 具体方法 5

*(1)* 基本操作 5

*(2)* 状态监测(Observer) 8

*(3)* 状态监测(BroadcastReceiver) 10

*(4)* 常量 12

附录一 兼容接口 12

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 \_DownloadManager\_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；
* 用到MD5校验的工程，需要将MD5.jar放在工程的libs目录中；

1. 原理简介

* DowloadManager组件单独维护一张表，用于操作下载相关的记录， 提供增删改查接口，操作下载数据库，完成对下载任务的操作；
* 应用通过自定义ContentObserver监听下载数据库的变化，来实时更新view层下载进度、状态的显示；

1. 具体方法
2. 基本操作

添加、删除、暂停等操作通过DownloadController功能类，或DownloadCommand功能类（兼容旧版本，下载教育类资源）来实现。具体操作如下：

1. **添加任务（批量）**

//第一步：创建一个下载任务对象 下载地址、文件名、存储路径是不可少的参数

final String url = “<http://dv.eebbk.net/xzzb/h8/xktb/[人教版]七年级数学下册.zip>”;

final String name = “[人教版]七年级数学下册.tia”;

final String path = “/mnt/sdcard”;

DownloadTask task = new DownloadTask(url, name, path);

//第二步：添加一些其他属性，可跳过

//md5校验，设置后能自动对下载数据进行MD5校验，不设置不校验

task.SetMD5(“1111111111111111”);

//文件后缀名(如果设置后缀名，文件名不能包含后缀名)

task.setFileExtension(“.tia”);

//设置通知栏是否可见

task.setNotificationVisibility(Request.VISIBILITY\_HIDDEN);

//是否要排队(新增接口)

task.setNeedQueue(false);

//是否要自动切换路径(新增接口)

task.setAllowAdjustSavePath(true);

//设置下载线程的优先级(新增接口)

task.setPriority( DownloadConstants.PRIORITY\_NORMAL);

// 设置下载时所允许的网络类型(新增接口)

task.setAllowedNetworkTypes(Request.NETWORK\_WIFI);

//扩展字段记录一些信息，用于附加信息存放于数据库中，使用方便，反查下载记 //录时也能用上（非常推荐使用）

task.putExtra(name, value);

……

//第三步：添加任务到下载控制器中

mDownloadController.addTask(task);

1. **删除任务（批量）**

//通过添加任务后的返回ID或者下载任务进行删除,文件也一起删除

mDownloadController.deleteTask(task);

//只删除任务不删除文件（新增接口）

mDownloadController.deleteTaskWithoutFile(task);

1. **暂停任务（批量）**

//通过添加任务后的返回ID或者下载任务进行暂停

mDownloadController.pauseTask(task);

1. **恢复任务（批量）**

//通过添加任务后的返回ID或者下载任务进行恢复

mDownloadController.resumeTask(task);

1. **重新下载（批量）**

//通过添加任务后的返回ID或者下载任务进行重新下载

mDownloadController.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

mDownloadController.refreshData(tasks);

1. **查询下载任务**
2. 通过ID条件查询

//直接通过ID来查询

ITask task = mDownloadController.getTaskById(id);

1. 通过Status条件查询

//直接通过Status来查询

ArrayList<ITask>taskList=mDownloadController.getTaskByStatus

(DownloadManager.STATUS\_FAILED);

1. 通过扩展字段条件查询（推荐使用）

//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下载任务

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

1. 通过多种条件查询

//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(DownloadManager.STATUS\_FAILED);

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

//第三步：查询

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

1. 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 DownloadChangeObserver extends ContentObserver {

public DownloadChangeObserver ( ) {

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=mDownloadController.

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

for(ITask task : taskList){

switch(task.getState()){

case DownloadManager.STATUS\_FAILED:

break;

case DownloadManager.STATUS\_SUCCESSFUL:

break;

case DownloadManager.STATUS\_PAUSED:

break;

case DownloadManager.STATUS\_PENDING:

case DownloadManager.STATUS\_RUNNING:

break;

default:break;

}

}

}

}

第二步：生成一个监听对象

DownloadController mDownloadObserver = new DownloadChangeObserver();

第三步：设置监听，在Activity 的 onResume中

getContentResolver().registerContentObserver(Impl.ALL\_DOWNLOADS\_CONTENT\_URI, true, mDownloadObserver);

第四步：取消监听，在Activity 的onPause 中

getContentResolver().unregisterContentObserver(mDownloadObserver);

1. 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.DOWNLOAD\_COMPLETE" />

</intent-filter>

</receiver>

**方式2：动态注册**

IntentFilter filter= new IntentFilter();

filter.addAction(DownloadConstants.ACTION\_DOWNLOAD\_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) {

//获取任务ID

long id = intent.getLongExtra(DownloadConstants.

EXTRA\_DOWNLOAD\_ID,-1);

//获取扩展任务信息

HashMap<String, String> extra = (HashMap<String, String>) intent

.getSerializableExtra(DownloadConstants.COLUMN\_EXTRAS);

//获取文件下载路径

String filePath = intent.getStringExtra(DownloadConstants.

EXTRA\_DOWNLOADED\_PATH);

//其它处理

….....

}

1. 常量

下载相关的各种状态常量放在downloadManager.jar 的 DownloadConstants类中。存放一些下载状态值、错误码、数据库的字段名等。

|  |  |
| --- | --- |
| **状态码** | **意义** |
| STATUS\_PENDING | 等待 |
| STATUS\_RUNNING | 下载中 |
| STATUS\_PAUSED | 暂停 |
| STATUS\_SUCCESSFUL | 成功 |
| STATUS\_FAILED | 失败 |

1. 附录一 兼容接口

DownloadCommand有特殊的功能接口，是为了兼容h10保留的，具体功能请查询详细接口文档；

/\*\*

\* 下载基础接口

\*/

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);

/\*\*

\* 删除任务（包含文件）

\*

\* @param ids

\* 通过 downloadID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（包含文件）

\*

\* @param pTasks

\* 单任务或任务数组皆可

\* @return 成功删除的数量

\*/

int deleteTask(ITask... pTasks);

/\*\*

\* 删除任务

\*

\* @param ids

\* 通过 downloadID删除任务，单任务或任务数组皆可

\* @return 成功删除的数量

\*/

int deleteTaskWithoutFile(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 deleteTaskWithoutFile(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downloadID查询

\* @return 下载任务

\*/

ITask getTaskById(long id);

/\*\*

\* 查询任务

\* @param status 根据下载状态查询

\* @return 相应状态的下载列表

\*/

ArrayList<ITask> getTaskByStatus(int status);

/\*\*

\* 获取所有的下载任务

\* @return 所有任务列表

\*/

ArrayList<ITask> getTask();

/\*\*

\* 查询任务

\* @param query 根据自定义query查询符合条件的下载任务

\* @return 符合query的任务列表

\*/

ArrayList<ITask> getTask(Query query);

/\*\*

\* 查询任务 根据扩展字段

\* @param keys 键数组

\* @param values 值数组

\* @return 相应的任务列表

\*/

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

/\*\*

\* 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);

/\*\*

\* 重新下载 根据downloadID

\* @param ids downloadID

\* @return 重新下载成功的数量

\*/

int reloadTask(long... ids);

/\*\*

\* 重新下载 根据下载任务

\* @param pTasks 重新下载的任务数组

\* @return 重新下载成功的数量

\*/

int reloadTask(ITask... pTasks);

/\*\*

\* 暂停下载 根据downloadID

\* @param ids downloadID

\* @return 暂停下载成功的数量

\*/

int pauseTask(long... ids);

/\*\*

\* 暂停下载 根据下载任务

\* @param pTasks 暂停的任务数组

\* @return 暂停成功的数量

\*/

int pauseTask(ITask... pTasks);

/\*\*

\* 恢复下载 根据downloadID

\* @param ids downloadID

\* @return 恢复下载成功的数量

\*/

int resumeTask(long... ids);

/\*\*

\* 恢复下载 根据下载任务

\* @param pTasks 恢复下载的任务数组

\* @return 恢复下载成功的数量

\*/

int resumeTask(ITask... pTasks);

}