CursorAdapter继承了BaseAdapter，主要用于为实现将数据库中内容显示的ListView来设置适配器，更常用的SimpleCursorAdapter就继承了CursorAdapter类

接下来看看其部分常用部分的源码

getView（）方法

public View getView(int position, View convertView, ViewGroup parent) {  
 if (!mDataValid) {  
 throw new IllegalStateException("this should only be called when the cursor is valid");  
 }  
 if (!mCursor.moveToPosition(position)) {  
 throw new IllegalStateException("couldn't move cursor to position " + position);  
 }  
 View v;  
 if (convertView == null) {  
 v = newView(mContext, mCursor, parent);  
 } else {  
 v = convertView;  
 }  
 bindView(v, mContext, mCursor);  
 return v;  
}

黄色部分正是通过Cursor游标的移动实现了为ListView每一个item获取数据，而此方法调用了newView（）和bindView（）两个方法，接下来看看这两个方法的源代码

*/\*\*  
 \* Makes a new view to hold the data pointed to by cursor.  
 \** ***@param*** *context Interface to application's global information  
 \** ***@param*** *cursor The cursor from which to get the data. The cursor is already  
 \* moved to the correct position.  
 \** ***@param*** *parent The parent to which the new view is attached to  
 \** ***@return*** *the newly created view.  
 \*/*public abstract View newView(Context context, Cursor cursor, ViewGroup parent);

通过文档注释我们可以知道newView（）方法返回的正是ListView中的item中的空间

*/\*\*  
 \* Bind an existing view to the data pointed to by cursor  
 \** ***@param*** *view Existing view, returned earlier by newView  
 \** ***@param*** *context Interface to application's global information  
 \** ***@param*** *cursor The cursor from which to get the data. The cursor is already  
 \* moved to the correct position.  
 \*/*public abstract void bindView(View view, Context context, Cursor cursor);

bindView（）方法为每一个View绑定了数据库中特定的一个data

另外CursorAdapter还有一个非常有用的方法:changeCursor（Cursor cursor）；

*/\*\*  
 \* Change the underlying cursor to a new cursor. If there is an existing cursor it will be  
 \* closed.  
 \*   
 \** ***@param*** *cursor The new cursor to be used  
 \*/*public void changeCursor(Cursor cursor) {  
 Cursor old = swapCursor(cursor);  
 if (old != null) {  
 old.close();  
 }  
}

通过源代码我们看到此方法调用了swapCursor（Cursor cursor）方法

*/\*\*  
 \* Swap in a new Cursor, returning the old Cursor. Unlike  
 \* {****@link*** *#changeCursor(Cursor)}, the returned old Cursor is <em>not</em>  
 \* closed.  
 \*  
 \** ***@param*** *newCursor The new cursor to be used.  
 \** ***@return*** *Returns the previously set Cursor, or null if there wasa not one.  
 \* If the given new Cursor is the same instance is the previously set  
 \* Cursor, null is also returned.  
 \*/*public Cursor swapCursor(Cursor newCursor) {  
 if (newCursor == mCursor) {  
 return null;  
 }  
 Cursor oldCursor = mCursor;  
 if (oldCursor != null) {  
 if (mChangeObserver != null) oldCursor.unregisterContentObserver(mChangeObserver);  
 if (mDataSetObserver != null) oldCursor.unregisterDataSetObserver(mDataSetObserver);  
 }  
 mCursor = newCursor;  
 if (newCursor != null) {  
 if (mChangeObserver != null) newCursor.registerContentObserver(mChangeObserver);  
 if (mDataSetObserver != null) newCursor.registerDataSetObserver(mDataSetObserver);  
 mRowIDColumn = newCursor.getColumnIndexOrThrow("\_id");  
 mDataValid = true;  
 // notify the observers about the new cursor  
 notifyDataSetChanged();  
 } else {  
 mRowIDColumn = -1;  
 mDataValid = false;  
 // notify the observers about the lack of a data set  
 notifyDataSetInvalidated();  
 }  
 return oldCursor;  
}

可以看到此changeCursor（）方法调用后效果相当于附带了调用了notifyDataSetChanged();我们就不用再显式调用notifyDataSetChanged();le