# N 平台Launcher左右滑动AllAPP界面开发文档

## Launcher结构简要普及

Launcher是以MVC框架为宏观指导进行开发的，一切我们可以看到的界面都属于view层，大部分数据都是在LauncherModel（M）里加载的，加载完之后通过launcher（C）这个控制器绑定都各个view上。

## 需求开发大纲：

1. 在View层添加左右滑动的allapp界面（以下称Lallapp）。
2. 将Launcher的数据的增删改绑定到该view上。
3. 按照之前Launcher的框架处理view与view之间的切换。

## 具体开发：

按照需求大纲的步骤开始一步一步开发

1. 准备左右滑动的Lallapp界面的源码，这个在Android的L平台上有相应的源码可以以此为参考来开发Lallapp界面。

这个左右滑动的Lallapp界面是个自定义的view，下边准备上这个自定义view的源码，

下边这三个文件就是这个自定义view的核心

AppsCustomizePagedView.java

AppsCustomizeTabHost.java

AppsCustomizeCellLayout.java

还有他们的父类

PagedViewCellLayout.java

PagedViewCellLayoutChildren.java

PagedViewGridLayout.java

PagedViewWidget.java

PagedViewWidgetImageView.java

PagedViewWithDraggableItems.java

将这6个文件添加到src/com/android/launcher3/目录下

添加它们引用的xml

在res目录下

res/layout/apps\_customize\_application.xml

res/layout/apps\_customize\_progressbar.xml

res/layout/apps\_customize\_widget.xml

res/layout/apps\_customize\_pane.xml

对于和原来launcher中的继承关系和接口的实现进行一些兼容，把该实现的方法都再实现上，这样就完成了自定义view的源码添加工作。

1. 将这个view绑定到launcher中

在\res\layout-port\launcher.xml中的

<com.android.launcher3.DragLayer>

标签里边添加

<include layout="@layout/apps\_customize\_pane"  
 android:id="@+id/apps\_customize\_pane"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:visibility="invisible"/>

这就将这个view添加到launcher.java这个activity中了

在launcher.java中药找到这个view

添加view变量

private AppsCustomizeTabHost mAppsCustomizeTabHost;  
private AppsCustomizePagedView mAppsCustomizeContent;

在setupOverviewPanel()方法的最后添加

// Setup AppsCustomize  
mAppsCustomizeTabHost = (AppsCustomizeTabHost) findViewById(R.id.*apps\_customize\_pane*);  
mAppsCustomizeContent =  
 (AppsCustomizePagedView) mAppsCustomizeTabHost.findViewById(R.id.*apps\_customize\_pane\_content*);

mAppsCustomizeContent.setup(this, mDragController);//注册拖拽事件 下文讲

## 将Launcher的数据的增删改绑定到该view上。

在launcher.java实现的LauncherModel的方法bindAllApplications(final ArrayList<AppInfo> apps) 里添加如下代码

if (mAppsCustomizeContent != null) {  
 mAppsCustomizeContent.setApps(apps);  
}

同理在bindAppsUpdated(final ArrayList<AppInfo> apps)

方法里添加

if (mAppsCustomizeContent != null) {  
 mAppsCustomizeContent.updateApps(apps);  
}

在bindAppInfosRemoved(final ArrayList<AppInfo> appInfos)方法里添加

// Update LAllApps  
if(mAppsCustomizeContent != null) {  
 mAppsCustomizeContent.removeApps(appInfos);  
}

这样就将数据绑定到Lallapps这个view上了还有增删的相关操作也绑定上了

其实仔细看一下这些地方都是launcher.java操作上下滑动的allapp的地方。

接下来的Launcher.java这个控制器进行view与view之间的切换的时候也是参考launcher这个应用中原来控制上下滑到的地方做文章就可以了

## 按照之前Launcher的框架处理view与view之间的切换。

View与view之间的切换无非包括从workspace切换到Lallapp界面。从Lallapp界面切换到workspace界面，从Lallapp界面拖动icon到workspace界面上，主要就这三个切换。

Android6.0之后这些状态之间的切换单独封装到一个类里

LauncherStateTransitionAnimation.java里

说下大致的流程

从workspace进入到Lallapp界面是隐藏workspace显示Lallapp

从Lallapp返回到workspace时隐藏Lallapp显示workspace

所以可以大体上分为四个方法

Workspace模块两个方法：显示workspace和隐藏workspace

Lallapp模块两个方法： 显示Lallapp和隐藏Lallapp

Workspace的显示和隐藏（这个是带有动画的和launcher里的hideWorkspaceSearchAndHotseat 和 showWorkspace不一样）也被封装到一个单独的类WorkspaceStateTransitionAnimation.java里，核心方法是animateWorkspace，这里我们就不多讲了。

Lallapp的显示和隐藏的方法原来的位置在L平台的launcher.java里也得移植过来，按照N平台launcher的架构就移植到LauncherStateTransitionAnimation.java里，显示Lallapp方法是showAppsCustomizeHelper隐藏Lallapp的方法是hideAppsCustomizeHelper，添加完之后肯定得做一些修改做兼容，怎么修改的这里就不说了，有提交记录的。

在LauncherStateTransitionAnimation.java里从workspace到Lallapp界面的方法是startAnimationToWorkspaceFromAllApps看方法名称就知道了，这个方法的最后就是原来调用从workspace到allapp界面的地方，所以我们在最后将原来的方法改为

if(LauncherAppState.*isLRAllApp*()){  
 mCurrentAnimation = hideAppsCustomizeHelper(toWorkspaceState,toWorkspacePage,animated ,onCompleteRunnable);  
} else {  
 // Only animate the search bar if animating to spring loaded mode from all apps  
 mCurrentAnimation = startAnimationToWorkspaceFromOverlay(fromWorkspaceState, toWorkspaceState,  
 toWorkspacePage, mLauncher.getAllAppsButton(), appsView,  
 animated, onCompleteRunnable, cb);  
}

LauncherAppState.isLRAllApp()这个是我添加的控制点击allapp按钮的时候是进上下滑动的allapp界面还是进左后滑动的Lallapp界面

同理在startAnimationToAllApps方法的最后将原来的代码修改为

if(LauncherAppState.*isLRAllApp*()){  
 mCurrentAnimation = showAppsCustomizeHelper(animated, true);  
}else {  
 // Only animate the search bar if animating from spring loaded mode back to all apps  
 mCurrentAnimation = startAnimationToOverlay(fromWorkspaceState,  
 Workspace.State.*NORMAL\_HIDDEN*, buttonView, toView, animated, cb);  
}

从Lallapp界面拖动也是走的隐藏Lallapp显示workspace方法所以不用考虑要单独处理这个显示了，从这里来看launcher这个这个应用的代码复用性真的很高。只不过需要将Lallapp的拖动注册到DragController上，这个在前文说道了在setupOverviewPanel()方法中添加的的mAppsCustomizeContent.setup(this, mDragController);

大体上移植步骤已经讲完了。后边有什么问题在补充吧。