Menu

1.通过加载menu/ xxx.xml文件创建菜单

Activity 每次启动，依次回调 onCreateOptionsMenu、 onPrepareOptionsMenu
`onCreateOptionMenu` ： 只在activity启动时，展示menu调用一次
`onPrepareOptionsMenu` ：在每次展示menu时，都会调用。即第一次在activity启动时，显示menu调用，以后每次点击菜单按钮，都会再次调用

`onCreateOptionMenu`用于初始化菜单栏，一般是通过加载xml文件，创建菜单

```
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.demo1, menu);
return true;
```

`onPrepareOptionsMenu`: 在上方法调用之后调用，返回true，显示菜单，false，不显示，在`invalidateOptionsMenu`之后也会调用，主要用于动态更新菜单栏

官方文档：

- 在 Android 2.3.x 及更低版本中，每当用户打开选项菜单时（按“菜单”按钮），系统均会调用 `onPrepareOptionsMenu()`。
- 在 Android 3.0 及更高版本中，当菜单项显示在应用栏中时，选项菜单被视为始终处于打开状态。 发生事件时，如果您要执行菜单更新，则必须调用 `invalidateOptionsMenu()` 来请求系统调用 `onPrepareOptionsMenu()`。

但是**根据测试**，发现在Android7.0 版本

- 第一次启动，依次执行`onCreateOptionMenu`、 `invalidateOptionsMenu` 方法
- 每次按菜单按钮，系统只会调用 `onPrepareOptionsMenu()`
- 而且，如果手动调用`invalidateOptionsMenu`方法，系统就会依次调用 `onCreateOptionMenu`、 `invalidateOptionsMenu` 方法

2. 创建浮动上下文菜单

1. 通过调用 registerForContextMenu()，注册应与上下文菜单关联的 View 并将其传递给 View。

```
//1.对View组件进行注册 上下文菜单按钮
registerForContextMenu(imageView);
registerForContextMenu(button);
```

2. 在 Activity 或 Fragment 中实现 onCreateContextMenu() 方法。
   当注册后的视图收到长按事件时，系统将调用您的 onCreateContextMenu() 方法。在此方法中，您通常可通过扩充菜单资源来定义菜单项。
   注意：如果 Activity 有多个视图，**每个视图均提供不同的上下文菜单**，则可使用上述方法中的参数`View v`确定要扩充的上下文菜单。v 对应长按的视图控件。

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMe
nu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();

    if (v.getId() == R.id.context_img) {
        inflater.inflate(R.menu.context_menu, menu);
    } else {
        inflater.inflate(R.menu.context_menu2, menu);
    }
}
```

3.**实现 onContextItemSelected()。**
用户点击选择菜单项时，系统将调用此方法，以便您能够执行适当的操作。

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.get
MenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

成功处理菜单项后，系统将返回 true。**如果未处理菜单项，则应将菜单项传递给超类实现。**如果 Activity 包括片段，则 Activity 将先收到此回调。 通过在未处理的情况下调用超类，系统会将事件逐一传递给每个片段中相应的回调方法（按照每个片段的添加顺序），直到返回 true 或 false 为止。（Activity 和 android.app.Fragment 的默认实现返回 false，因此您始终应在未处理的情况下调用超类。）

3.创建 ActionMode（上下文操作模式）

1. 为单个视图启用上下文模式

1. 实现 ActionMode.Callback 接口。在其回调方法中，您既可以为上下文操作栏指定操作，又可以响应操作项目的点击事件，还可以处理操作模式的其他生命周期事件。
2. 当需要显示操作栏时（例如，用户长按视图），请调用 startActionMode()。启用上下文操作模式。

```
startActionMode() ：接收参数：ActionMode.Callback的实现子类，返回ActionMode对象
```

```
public class ContextOperate extends AppCompatActivity {
    private Context mContext;
    private ActionMode mActionMode;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_context_operate);
        mContext = this;
        Button button = (Button) findViewById(R.id.context_operat
e);
        button.setOnLongClickListener(new View.OnLongClickListene
r() {
            @Override
            public boolean onLongClick(View v) {
                if (mActionMode != null) {
                    return false;
                }
                mActionMode = startActionMode(mActionModeCallback);
                return true;
            }
        });
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mActionMode != null) {
                    mActionMode.invalidate();
                }
            }
        });
    }

    private ActionMode.Callback mActionModeCallback = new ActionMod
e.Callback() {
        private static final String TAG = "ContextOperate";
        private int num=0;
        //Called each time the action mode is created
        // 通过startActionMode()、或者 mActionMode.invalidate() 都会创
建新的 actionMode，调用该方法
        //但是点击 菜单按钮（三个点），不会触发该方法，只会触发 onPrepareAct
ionMode方法，和浮动上下文菜单类似
        // 加载菜单xml文件， 创建上下文ActionMode
        @Override
        public boolean onCreateActionMode(ActionMode mode, Menu men
u) {
            Log.e(TAG, "onCreateActionMode");
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.context_menu2, menu);
```

```
            return true;
        }

        // Called each time the action mode is shown. Always called
after onCreateActionMode, but
        // may be called multiple times if the mode is invalidated.
        @Override
        public boolean onPrepareActionMode(ActionMode mode, Menu me
nu) {
            Log.e(TAG, "onPrepareActionMode"+(++num));

            menu.add(0, 0, 0, "哈哈"+num);
            return true;   // Return false if nothing is done
        }

        // Called when the user selects a contextual menu item
        @Override
        public boolean onActionItemClicked(ActionMode mode, MenuIte
m item) {
            switch (item.getItemId()) {
                case R.id.delete:
                    Toast.makeText(mContext, "按钮_删除", Toast.LENGT
H_SHORT).show();
                    // 已经获取到item，现在结束 actionMode了
                    mode.finish();
                    return true;
                case R.id.edit:
                    Toast.makeText(mContext, "按钮_编辑", Toast.LENGT
H_SHORT).show();
                    mode.finish();
                    return true;
                default:
                    return false;
            }
        }

        @Override
        public void onDestroyActionMode(ActionMode mode) {
            mActionMode = null;
        }

    };


}
```

3. 为ListView 创建上下文模式

1. 实现 AbsListView.MultiChoiceModeListener 接口，并使用 setMultiChoiceModeListener() 为视图组设置该接口。在侦听器的回调方法中，您既可以为上下文操作栏指定操作，也可以响应操作项目的点击事件，还可以处理从 ActionMode.Callback 接口继承的其他回调。
2. 使用 CHOICE_MODE_MULTIPLE_MODAL 参数调用 setChoiceMode()。

```
public class ListActivityMenu extends AppCompatActivity {
    private static final String TAG = "ListActivityMenu";
    private Context mContext = ListActivityMenu.this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_view_menu);
        String[] str = {
                "张三1", "李四2", "王五3", "张三4", "李四5", "王五6",
"张三7", "李四8",
                "王五9", "张三10", "李四11", "王五12", "张三13", "李四1
4", "王五15"
        };
        ArrayAdapter<String> adapter = new ArrayAdapter<>(ListActiv
ityMenu.this, android.R.layout.simple_list_item_1, str);
        final ListView listView = (ListView) findViewById(R.id.lis
t_view_menu);
        listView.setAdapter(adapter);
        mContext = this;

        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODA
L);
        listView.setMultiChoiceModeListener(new AbsListView.MultiCh
oiceModeListener() {
            //当出现 ActionMode以后，点击列表中的每一项，该项就会被设置成选
中 checked =true,再次点击就会变成 false
            @Override
            public void onItemCheckedStateChanged(ActionMode mode,
int position, long id, boolean checked) {
                Toast.makeText(mContext, "position" + position + "c
hanged," + checked, Toast.LENGTH_SHORT).show();
                // 如何获取被点击的 ListView内部的子项Item
 /*                 if (checked) {
                    item.setBackgroundColor(getResources().getColo
r(R.color.colorItemBack));
                }else{
                    item.setBackgroundColor(getResources().getColo
r(android.R.color.background_light));
                }*/
            }

            @Override
            public boolean onCreateActionMode(ActionMode mode, Menu
menu) {
                MenuInflater inflater = getMenuInflater();
                inflater.inflate(R.menu.context_menu2, menu);
                return true;
```

```
            }

            @Override
            public boolean onPrepareActionMode(ActionMode mode, Men
u menu) {
                return true;
            }

            @Override
            public boolean onActionItemClicked(ActionMode mode, Men
uItem item) {
                switch (item.getItemId()) {
                    case R.id.delete:
                        Toast.makeText(mContext, "按钮_删除", Toast.L
ENGTH_SHORT).show();
                        // 已经获取到item，现在结束 actionMode了

                        deleteSelected();
                        mode.finish();
                        return true;
                    case R.id.edit:
                        Toast.makeText(mContext, "按钮_编辑", Toast.L
ENGTH_SHORT).show();
                        mode.finish();
                        return true;
                    default:
                        return false;
                }
            }

            // 如何删除ListView内部的 子项
            public void deleteSelected() {
                SparseBooleanArray array = listView.getCheckedItemP
ositions();
                Toast.makeText(mContext, array.toString(), Toast.LE
NGTH_SHORT).show();

                Log.e(TAG, "deleteSelected" + array.size());

                for (int i = 0; i < array.size(); i++) {
                    int key = array.keyAt(i);
                    boolean res = array.get(key);
                    Log.e(TAG, key + "==>" + res);
                }

            }

            @Override
            public void onDestroyActionMode(ActionMode mode) {
```

```
            }


        });
    }


}
```