在 Android 应用中，Fragment 是一个独立的模块化组件，可以嵌入到 Activity 中，并且可以在多个 Activity 中重复使用。每个 Fragment 都必须附加到某个 Activity 上才能显示和运行。

当你调用 getActivity() 时，你获得了当前 Fragment 所属的 Activity 的实例。这个方法在很多场景下非常有用，例如：

1. **获取上下文（Context）**：你可以使用 getActivity() 来获取 Context 对象，这对于需要上下文的操作非常重要，比如创建适配器、启动其他活动、显示对话框等。
2. **与 Activity 交互**：你可以通过 getActivity() 来调用 Activity 的方法，或者传递数据到 Activity 中。

GroupListFragment

public class GroupListFragment extends Fragment {

private static final String DIALOG\_TITLE = "添加分组";

private static final String DIALOG\_CONTENT = "请输入新的分组名称";

private static final String DIALOG\_INPUT\_HINT = "分组名称";

private static final String TOAST\_ADD\_SUCCESS = "添加成功";

private static final String TOAST\_GROUP\_EXISTS = "分组已存在";

public ExpandableListView expandableListView;//Android SDK标准组件，用于显示一个可扩展的列表，每个列表项可以展开以显示子项。

public GroupAdapter groupAdapter;

List<PeopleGroup> peopleGroupList = new ArrayList<>();

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.list\_fragment\_2, container, false);//使用 LayoutInflater 将 XML 布局文件 list\_fragment\_2 转换为一个 View 对象

groupAdapter = new GroupAdapter(getActivity(), peopleGroupList);//这里的 getActivity() 用于获取当前 Fragment 所附加的 Activity 对象，然后将其传递给 GroupAdapter 的构造函数，确保适配器拥有正确的上下文（Context）。

expandableListView = view.findViewById(R.id.expand\_list);

expandableListView.setAdapter(groupAdapter);

FloatingActionButton fab = view.findViewById(R.id.floating\_action\_button);//设置浮动操作按钮

fab.setColorFilter(Color.WHITE);

fab.setOnClickListener(new View.OnClickListener() {//设置按钮点击事件

@Override

public void onClick(View v) {

new XPopup.Builder(getContext())

.asInputConfirm(DIALOG\_TITLE, DIALOG\_CONTENT, DIALOG\_INPUT\_HINT, new OnInputConfirmListener() {//设置输入确认对话框的标题、内容和提示，并定义确认按钮的监听器。

@Override

public void onConfirm(String text) {

List<PeopleGroup> list = LitePal.where("name = ?", text).find(PeopleGroup.class);//LitePal.where("name = ?", text).find(PeopleGroup.class)：查询数据库中名称为 text 的分组。

if (list == null || list.isEmpty()) {

PeopleGroup group = new PeopleGroup();

group.setId(System.currentTimeMillis());//设置分组的 ID 为当前时间戳。

group.setName(text);

group.setPeopleList(new ArrayList<>());

group.save();//将新分组保存到数据库。

peopleGroupList.add(group);//将新分组添加到分组列表。

groupAdapter.notifyDataSetChanged();//通知适配器数据已更改。

Toast.makeText(getContext(), TOAST\_ADD\_SUCCESS, Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(getContext(), TOAST\_GROUP\_EXISTS, Toast.LENGTH\_SHORT).show();

}

}

})

.show();

}

});

return view;

}

@Override

public void onHiddenChanged(boolean hidden) {//boolean hidden：表示 Fragment 当前是否隐藏。true 表示隐藏，false 表示显示。

super.onHiddenChanged(hidden);//super.onHiddenChanged(hidden)：调用父类 Fragment 的 onHiddenChanged 方法，以确保父类的隐藏状态变化逻辑也能执行。

if (!hidden) {

onResume();

}

}

@Override

public void onResume() {

super.onResume();

List<PeopleGroup> peopleGroupList = LitePal.findAll(PeopleGroup.class);//使用 LitePal ORM 框架查询数据库中的所有 PeopleGroup 对象，并将结果存储在 peopleGroupList 中。

List<People> peopleList = LitePal.findAll(People.class);//使用 LitePal ORM 框架查询数据库中的所有 People 对象，并将结果存储在 peopleList 中。

for (PeopleGroup peopleGroup : peopleGroupList) {

List<People> list = new ArrayList<>();

String name = peopleGroup.getName();

for (People people : peopleList) {

if (!TextUtils.isEmpty(people.getG())) {

if (people.getG().equals(name)) {

list.add(people);

}

}

}

peopleGroup.setPeopleList(list);//将经过筛选后的 list 设置为当前 PeopleGroup 对象的联系人列表。

}

this.peopleGroupList.clear();

this.peopleGroupList.addAll(peopleGroupList);

groupAdapter.notifyDataSetChanged();//知 ExpandableListView 使用的适配器 groupAdapter 数据已经改变，需要刷新界面显示。

}

}