Fragment与Activity、Fragment相互交互

原创 xyzso1z 2020-08-09 17:02:39 40 收藏 编辑 版权

分类专栏: Android 文章标签: Fragment Activity

一、分类

1 Activity 与 Fragment 之间通信

- Activity 传递数据到 Fragment
- Fragment 传递数据到 Activity

2 Fragment 与 Fragment 之间通信

二、 Activity 与 Fragment 之间通信

2.1、 Activity 如何传递数据到 Fragment

方法1

使用 Fragment.setArguments(bundle) ,把 bundle 设置到 fragment ,在 Fragment 中 onCreateView() 中取出。

- 在 Activity 中动态添加 Fragment 时使用 Fragment 的 setArguments(bundle) 方法把 bundle 对象传递给 fragment

```
        // 步骤1：获取FragmentManager
        FragmentManager fragmentManager = getSupportFragmentManager();

        // 步骤2：获取FragmentTransaction
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

        // 步骤3：创建需要添加的Fragment
        ActivityAndFragmentDataCommunicationFragment fragment = new ActivityAndFragmentDataCommuni

        // 步骤4:创建Bundle对象
        // 作用:存储数据，并传递到Fragment中
        Bundle bundle = new Bundle();

        // 步骤5:往bundle中添加数据
        bundle.putString("message", editText.getText().toString());

        // 步骤6:把数据设置到Fragment中
        fragment.setArguments(bundle);

        // 步骤7：动态添加fragment
        // 即将创建的fragment添加到Activity布局文件中定义的占位符中（FrameLayout）
        fragmentTransaction.add(R.id.fragment, fragment);
        fragmentTransaction.commit();
```

- 在 Fragment 中接收 Activity 传递的 Bundle 对象

```
    //1.获取Bundle
     Bundle bundle = getArguments();
     //2. 判断bundle对象是否包含key值为message的数据
```

```
    if (bundle != null && bundle.containsKey("message")) {
        //3. 从bundle中获取key值为message的数据
        String message = bundle.getString("message");
    }
```

方法2

可以在 `fragment` 中设置 `public` 方法，因 `Activity` 持有 `fragment` 对象，故可以直接调用 `fragment` 中的方法。

2.2、 Fragment 如何传递数据给 Activity

通过**接口回调**，在 `Activity` 中把接口引用传递给 `Fragement` ，

- 设置回调接口

```
public interface ICallBack {
  /**
   * 向Activity传递数据
   */
  void sendDataToActivity(String string);

  /**
   * 从Activity获取数据
   */
  String getDataFromActivity();
}
```

- 在 `Fragment` 中接收 `ICallBack` 引用

```
  private ICallBack callBack;

  //设置 ICallBack接口回调 方法
  public void sendCallBack(ICallBack callBack) {
      this.callBack = callBack;
  }

  @Override
  public void onClick(View v) {
      int id = v.getId();
      if (id == R.id.get) {
          getDataFromActivity();
      } else if (id == R.id.commit) {
          sendDataToActivity();
      }
  }

   /**
   * 向Activity传递数据
   * 调用接口sendDataToActivity()方法,使Activity接收回调
   */
  private void sendDataToActivity() {
      if (callBack != null) {
          callBack.sendDataToActivity(editText.getText().toString());
      }
  }

```

```
    /**
     * 从Activity获取数据
     * 调用接口getDataFromActivity()方法,使Activity接收回调
     */
    private void getDataFromActivity() {
        if (callBack != null) {
            String data = callBack.getDataFromActivity();
            textView.setText(data);
        }
    }
```

- 在 Activity 中将 callBack 引用设置给 fragment

```
        ActivityAndFragmentDataCommunicationFragment fragment = new ActivityAndFragmentDataCommuni

        //设置fragment数据回调
        fragment.sendCallBack(new ICallBack() {
            @Override
            public void sendDataToActivity(String string) {
                // 当fragment中调用callBack.sendDataToActivity()方法时会回调到该方法,
                // 即把fragment中数据传递到Activity
                textView.setText(string);
            }

            @Override
            public String getDataFromActivity() {
             // 当fragment中调用callBack.getDataFromActivity()方法时会回调到该方法,
             // 该方法return 的数据fragment会获取到, 即Fragment通过此方式获取到Activity中数据
                return editText.getText().toString();
            }
        });
```

根据上述说明可以看到 fragment 如何从 Activity 中获取数据以及 fragment 如何将数据传递给 Activity （此处不仅仅传递和获取了数据，而且直接调用Activity中的方法）。

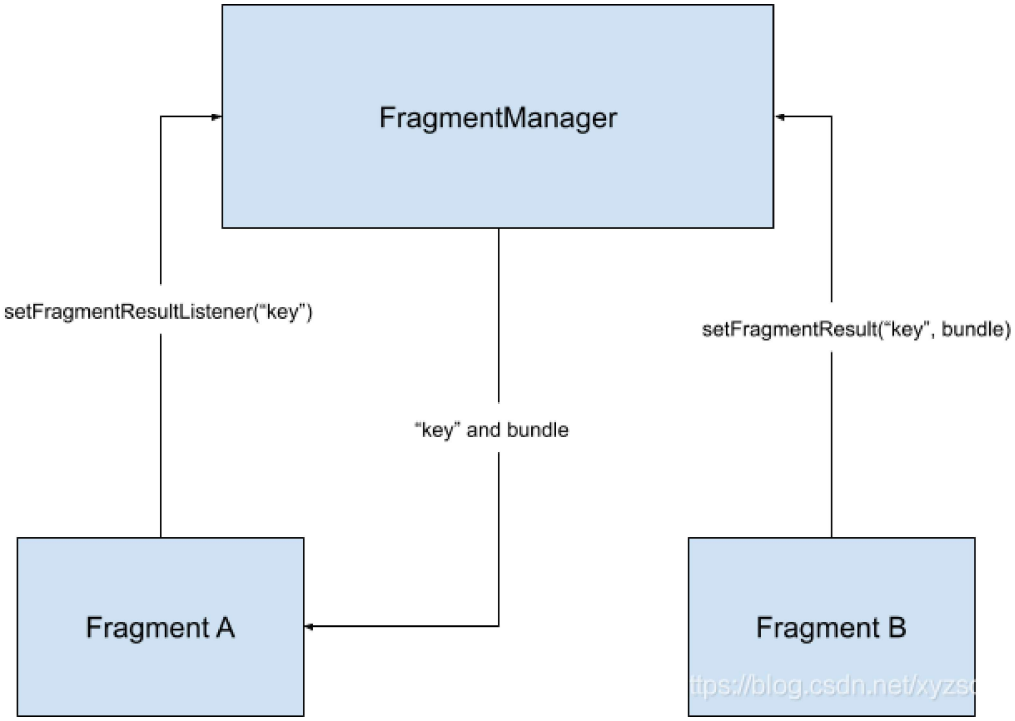
三、 Fragment 与 Fragment 之间通信

因为每个 FragmentManager 都会实现 FragmentResutOwner 。这意味着 FragmentManager 可以充当 Fragment result 的集中存储区。此更改通过设置 Fragment 结果并监听这些结果，而**不要 Fragment 直接引用彼此，让单独的 Fragment 相互通信。**
如需将数据从 Fragment B 传回到 Fragment A ,要先在接收结果的 Fragment A 上设置结果监听器。在 Fragment A 的 FragmentManager 上调用 setFragmetResultListener() API,如下所示：

```
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getParentFragmentManager().setFragmentResultListener("key", this, new FragmentResultListener() {
        @Override
        public void onFragmentResult(@NonNull String key, @NonNull Bundle bundle) {
            // We use a String here, but any type that can be put in a Bundle is supported
            String result = bundle.getString("bundleKey");
            // Do something with the result...
        }

```

```
    });
}
```



`Fragment B` 使用 `FragmentManager` 将数据发送到 `Fragment A`

在生成结果的 `Fragment B` 中，必须使用相同的 `requestKey` 在同一 `FragmentManager` 上设置结果。可以使用 `setFragmentReusult()` API来完成此操作：

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Bundle result = new Bundle();
        result.putString("bundleKey", "result");
        getParentFragmentManager().setFragmentResult("requestKey", result);
    }
});
```

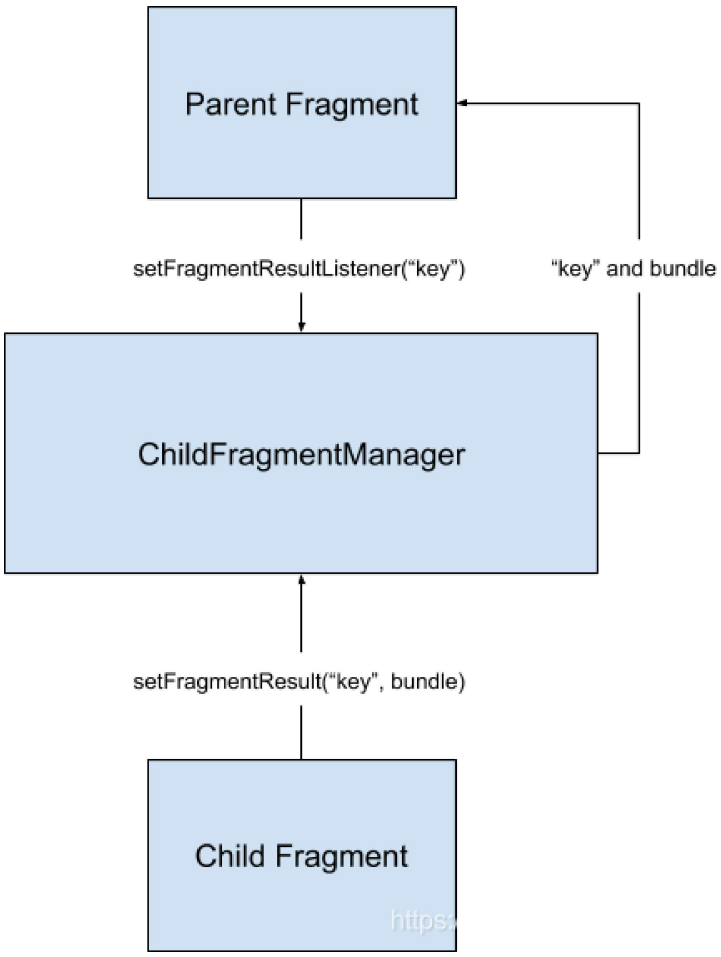
然后，在值为 `STARTED` 后， `Fragment A` 便会收到结果并执行监听器回调。
对于一个 `key` ，只能有一个监听器和一个结果。如果对同一 `key` 多次调用 `setResult()` ，则系统会将 `Fragment B` 从返回堆栈退出之前的最近 `result` 发送给 `Fragment A` 。如果设置的结果没有相应的监听器来接收，则结果会存储在 `FragmentManager` 中，知道设置一个具有相同 `key` 的监听器。监听器的 `Fragment` 必须为 `STARTED` ,然后该 `Fragment` 才能收到结果。监听器收到结果并触发 `onFragmentResult()` 回调后，结果会被清除。这种行为有两个主要影响：

- 返回堆栈上的 `Fragment` 只有在被弹出栈顶且未 `STARTED` 之后才会受到结果
- 如果在设置结果时监听结果的 `Fragment` 为 `STARTED` ,则系统会立即触发监听器的回调。
  注意：某个 `Fragment` 为 `DESTROYED` 之后，就无法再对该 `Fragment` 设置监听器。

3.1在父级 `Fragment` 和子级 `Fragment` 之间传递结果

如需将结果从子级 `Fragment` 传递到父级 `Fragment`，父级 `Fragment` 在调用 `setFragmentResultListener()` 时应该使用 `getChildFragmentManager()` 而不是 `getParentFragmentManager()`。

```
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // We set the listener on the child fragmentManager
    getChildFragmentManager().setFragmentResultListener("key", this, new FragmentResultListener() {
        @Override
        public void onFragmentResult(@NonNull String key, @NonNull Bundle bundle) {
            String result = bundle.getString("bundleKey");
            // Do something with the result..
        }
    });
}
```



子级 Fragment 可以使用 FragmentManager 将结果发送到其父级 Fragment。

子级Fragment在其FragmentManager上设置结果。然后，当父级Fragment为STARTED时便会收到结果：

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Bundle result = new Bundle();
        result.putString("bundleKey", "result");
```

```
            // The child fragment needs to still set the result on its parent fragment manager
            getParentFragmentManager().setFragmentResult("requestKey", result);
        }
    });
```