(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 107203369 A
(43)申请公布日 2017.09.26

(21)申请号 201610150785.X

(22)申请日 2016.03.16

(71)申请人 阿里巴巴集团控股有限公司
地址 英属开曼群岛大开曼资本大厦一座四层847号邮箱

(72)发明人 赵翔宇 丁靓子

(74)专利代理机构 北京博思佳知识产权代理有限公司 11415
代理人 林祥

(51)Int.Cl.
G06F 9/44(2006.01)

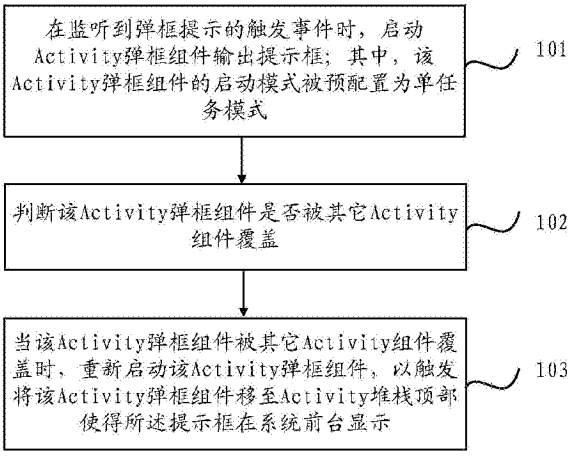
权利要求书2页 说明书10页 附图3页

(54)发明名称

基于Android的弹框提示方法及装置

(57)摘要

本申请提供一种基于Android的弹框提示方法及装置，其中的方法包括：在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；判断该Activity弹框组件是否被其它Activity组件覆盖；当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。本申请可以在启动Activity弹框组件向用户输出提示框后，确保该提示框不被覆盖，始终在系统前台显示。



CN 107203369 A

1.一种基于Android的弹框提示方法，其特征在于，该方法包括：

在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

判断该Activity弹框组件是否被其它Activity组件覆盖；

当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

2.根据权利要求1所述的方法，其特征在于，所述判断该Activity弹框组件是否被其它Activity组件覆盖包括：

判断所述Activity弹框组件的onPause或者onStop方法是否被调用；

当所述Activity弹框组件的onPause或者onStop方法被调用时，确定该Activity弹框组件被其它Activity组件覆盖。

3.根据权利要求2所述的方法，其特征在于，所述当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件包括：

当所述Activity弹框组件的onPause方法被调用，基于所述onPause方法重新启动该Activity弹框组件；

当所述Activity弹框组件的onStop方法被调用时，判断所述Activity弹框组件的onPause方法是否已被调用，如果所述onPause方法未被调用，则基于所述onStop方法重新启动该Activity弹框组件。

4.根据权利要求3所述的方法，其特征在于，所述基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件包括：

判断所述onPause方法或者所述onStop方法的调用是否是由针对所述Activity弹框组件的关闭事件所触发；

如果所述onPause方法或者所述onStop方法的调用不是由针对所述Activity弹框组件的关闭事件所触发，则基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件。

5.根据权利要求4所述的方法，所述方法还包括：

如果所述onPause方法或者所述onStop方法的调用是由针对所述Activity弹框组件的关闭事件所触发，则针对所述Activity弹框组件执行正常关闭。

6.一种基于Android的弹框提示装置，其特征在于，该装置包括：

启动模块，用于在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

判断模块，用于判断该Activity弹框组件是否被其它Activity组件覆盖；

重启模块，用于在该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

7.根据权利要求6所述的装置，其特征在于，所述判断模块具体用于：

判断所述Activity弹框组件的onPause或者onStop方法是否被调用；

当所述Activity弹框组件的onPause或者onStop方法被调用时，确定该Activity弹框组件被其它Activity组件覆盖。

8.根据权利要求7所述的装置，其特征在于，所述重启模块具体用于：

当所述Activity弹框组件的onPause方法被调用，基于所述onPause方法重新启动该Activity弹框组件；

当所述Activity弹框组件的onStop方法被调用时，判断所述Activity弹框组件的onPause方法是否已被调用，如果所述onPause方法未被调用，则基于所述onStop方法重新启动该Activity弹框组件。

9.根据权利要求8所述的装置，其特征在于，所述重启模块进一步用于：

判断所述onPause方法或者所述onStop方法的调用是否是由针对所述Activity弹框组件的关闭事件所触发；

如果所述onPause方法或者所述onStop方法的调用不是由针对所述Activity弹框组件的关闭事件所触发，则基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件。

10.根据权利要求9所述的装置，其特征在于，所述装置还包括：

关闭模块，用于在所述onPause方法或者所述onStop方法的调用是由针对所述Activity弹框组件的关闭事件所触发时，针对所述Activity弹框组件执行正常关闭。

基于Android的弹框提示方法及装置

技术领域

[0001] 本申请涉及通信领域，尤其涉及一种基于Android的弹框提示方法及装置。

背景技术

[0002] 在Android系统中，对于一些需要向用户输出弹框的事件，通过可以由系统向用户输出填充了提示文案的弹框来提示用户。例如，该事件可以是单点登录事件。所谓单点登陆，通常是指只允许用户在一台设备上登陆其账号，一旦用户在其它设备上使用相同的登录账号登陆，需要把用户从当前的设备上踢出登陆态。当系统检测到了单点登录事件时，可以向用户输出弹框提示用户当前登录账号已在其它设备上登录，此时该弹框中的提示文案可以包括用户被踢出登录的原因。然而，在相关技术中，Android系统在向用户输出弹框时，输出的弹框可能会被其它弹框或者其它页面所覆盖，从而影响用户对系统终端的APP或者基本功能的使用。

发明内容

[0003] 本申请提出一种基于Android的弹框提示方法，该方法包括：

[0004] 在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

[0005] 判断该Activity弹框组件是否被其它Activity组件覆盖；

[0006] 当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

[0007] 可选的，所述判断该Activity弹框组件是否被其它Activity组件覆盖包括：

[0008] 判断所述Activity弹框组件的onPause或者onStop方法是否被调用；

[0009] 当所述Activity弹框组件的onPause或者onStop方法被调用时，确定该Activity弹框组件被其它Activity组件覆盖。

[0010] 可选的，所述当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件包括：

[0011] 当所述Activity弹框组件的onPause方法被调用，基于所述onPause方法重新启动该Activity弹框组件；

[0012] 当所述Activity弹框组件的onStop方法被调用时，判断所述Activity弹框组件的onPause方法是否已被调用，如果所述onPause方法未被调用，则基于所述onStop方法重新启动该Activity弹框组件。

[0013] 可选的，所述基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件包括：

[0014] 判断所述onPause方法或者所述onStop方法的调用是否是由针对所述Activity弹框组件的关闭事件所触发；

[0015] 如果所述onPause方法或者所述onStop方法的调用不是由针对所述Activity弹框组件的关闭事件所触发，则基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件。

[0016] 可选的，所述方法还包括：

[0017] 如果所述onPause方法或者所述onStop方法的调用是由针对所述Activity弹框组件的关闭事件所触发，则针对所述Activity弹框组件执行正常关闭。

[0018] 本申请还提出一种基于Android的弹框提示装置，该装置包括：

[0019] 启动模块，用于在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

[0020] 判断模块，用于判断该Activity弹框组件是否被其它Activity组件覆盖；

[0021] 重启模块，用于在该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

[0022] 可选的，所述判断模块具体用于：

[0023] 判断所述Activity弹框组件的onPause或者onStop方法是否被调用；

[0024] 当所述Activity弹框组件的onPause或者onStop方法被调用时，确定该Activity弹框组件被其它Activity组件覆盖。

[0025] 可选的，所述重启模块具体用于：

[0026] 当所述Activity弹框组件的onPause方法被调用，基于所述onPause方法重新启动该Activity弹框组件；

[0027] 当所述Activity弹框组件的onStop方法被调用时，判断所述Activity弹框组件的onPause方法是否已被调用，如果所述onPause方法未被调用，则基于所述onStop方法重新启动该Activity弹框组件。

[0028] 可选的，所述重启模块进一步用于：

[0029] 判断所述onPause方法或者所述onStop方法的调用是否是由针对所述Activity弹框组件的关闭事件所触发；

[0030] 如果所述onPause方法或者所述onStop方法的调用不是由针对所述Activity弹框组件的关闭事件所触发，则基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件。

[0031] 可选的，所述装置还包括：

[0032] 关闭模块，用于在所述onPause方法或者所述onStop方法的调用是由针对所述Activity弹框组件的关闭事件所触发时，针对所述Activity弹框组件执行正常关闭。

[0033] 本申请中，通过在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；判断该Activity弹框组件是否被其它Activity组件覆盖；当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示，实现了当启动Activity弹框组件向用户输出了提示框后，可以确保该提示框不被覆盖，始终在系统前台显示。

附图说明

[0034] 图1是本申请一实施例提供的一种基于Android的弹框提示方法的流程图；

[0035] 图2是本申请一实施例提供的一种基于Android的弹框提示装置的逻辑框图；

[0036] 图3是本申请一实施例提供的承载所述一种基于Android的弹框提示装置的终端的硬件结构图。

具体实施方式

[0037] 在相关技术中，Android系统在向用户输出弹框时，通常可以通过如下方式实现：

[0038] 在示出的一种实现方式中，Android系统可以利用Android系统中的Window组件来进行弹框。

[0039] 通过Window组件来弹框，可以实现在任意页面弹框，且不被其他页面覆盖，然而通过Window组件来弹框，通常支持用户对弹框进行设置，因此当用户通过手动设置在系统中将弹框禁掉，则会导致系统无法进行弹框。

[0040] 在示出的另一种实现方式中，Android系统可以利用Activity组件来进行弹框。

[0041] 在Android系统中，Activity组件是一个负责与用户进行交互的组件，该组件可以提供一个交互界面，并在交互界面上显示一些交互控件(比如按钮、弹窗等)，用户可以通过操作这些交互控件与终端进行交互，终端在后台可以监听用户针对这些交互控件的事件，来完成与本次交互对应的任务。终端通过Activity组件来弹框，可以实现将系统中任意的Activity组件作为弹框的主体，在任意页面中来进行弹框。然而，由于Activity组件通常是以Activity堆栈的形态进行管理的，一旦该Activity组件在Activity堆栈中被其它Activity组件所覆盖，即该其它Activity组件占据Activity堆栈顶部，而该Activity组件不再处于Activity堆栈的顶部时，该Activity组件输出的弹框将会被其它弹框或者界面覆盖，因而无法在系统前台进行显示，对系统的APP或者基本功能的使用造成影响。

[0042] 在示出的另一种实现方式中，Android系统可以在一个固定的页面进行弹框。

[0043] 然而在一个固定的页面进行弹框，无法实现实现在任意界面进行弹框的目的，因此不够灵活，而且交互体验不好。例如，以在登录页面进行弹框为例，输出的弹框智能在登录页面进行显示，系统向用户输出弹框后，如果系统前台显示的为其它页面，那么只有用户在主动进行页面切换，回到登录页面才能够查看到系统输出的弹框，从而会导致系统输出的弹框无法及时的通知到用户。

[0044] 有鉴于此，本申请提出一种基于Android的弹框提示方法，通过在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；判断该Activity弹框组件是否被其它Activity组件覆盖；当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示；由于在本申请中，Activity弹框组件为单任务模式，因此当该Activity弹框组件被重新启动后，可以立即被移至Activity堆栈顶部，从而在启动Activity弹框组件向用户输出了提示框后，可以确保该提示框不被覆盖，始终在系统前台显示。

[0045] 同时，由于本申请不再基于Android系统中的Window组件进行弹框，不再支持用户

对弹框进行设置，因此可以避免由于用户通过手动设置在系统中将弹框禁掉导致无法进行弹框的问题。

[0046] 另外，由于本申请通过Activity弹框组件进行弹框，因此可以实现将系统中任意的Activity组件作为主体，在任意页面中来进行弹框，从而具有很高的灵活性。

[0047] 下面通过具体实施例并结合具体的应用场景对本申请进行描述。

[0048] 请参考图1，图1是本申请一实施例提供的一种基于Android的弹框提示方法，应用于终端，所述方法执行以下步骤：

[0049] 步骤101，在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

[0050] 在本例中，上述终端可以包括搭载Android系统的智能终端；例如，智能手机、平板电脑等移动终端等。

[0051] 上述弹框提示的触发事件可以包括终端系统监听到的需要面向用户输出弹框提示的事件。其中，在不同的应用场景中，上述弹框提示的触发事件可以分别包括不同类型的事件。

[0052] 例如，在单点登录的应用场景中，上述弹框提示的触发事件可以包括单点登录事件。在该应用场景中，当用户使用相同登录账号在其它终端上登录时，当前终端可以退出登录状态，并面向用户输出一个提示框提示本次退出登录状态的原因。

[0053] 又如，在呼叫来电接听的应用场景中，上述弹框提示的触发事件可以包括呼叫来电事件。当终端的系统监听到呼叫来电时，可以向用户输出一个提示框提示用户接收到了来电，用户此时可以针对该提示框进行相应的操作(比如向上滑动或者向下滑动)来接听来电或者挂断来电。

[0054] 在本例中，终端在向用户进行弹框时，仍然可以利用预设的Activity弹框组件来实现。当终端监听到弹框提示的触发事件后，可以启动Activity弹框组件，创建一个用于输出提示框的Activity实例，然后基于该实例将提示文案填充到提示框中向用户输出。

[0055] 在Android系统中，Activity组件的启动模式(launchMode)通常包standard模式，singleTop模式，singleTask模式(即单任务模式)以及singleInstance模式等四种模式。

[0056] 其中，当Activity组件的启动模式为singleTask模式时，该Activity弹框组件将只允许唯一的Activity实例运行。

[0057] 在singleTask模式下，该Activity弹框组件被启动后，系统会判断Activity堆栈中是否已经存在与该Activity弹框组件对应的Activity实例，由于当前Activity堆栈中已经存在与该Activity弹框组件对应的Activity实例，因此在singleTask模式下系统将不会再新建Activity实例，会将想要覆盖该Activity弹框组件的其它Activity组件弹出Activity堆栈，并将已经存在的该Activity示例所对应的Activity弹框组件重新移至Activity堆栈的顶部。在这种情况下，基于该Activity实例输出的提示框将会重新在系统前台显示而不被覆盖。

[0058] 基于此，在本例中，为了使Activity弹框组件输出的提示框不被其它提示框或者页面所覆盖，可以在初始状态下，将该Activity弹框组件的启动模式设置为singleTask模式。

[0059] 步骤102，判断该Activity弹框组件是否被其它Activity组件覆盖；

[0060] 在Android系统中，Activity组件的生命周期中通常包括running状态(运行状态)、Pause状态(暂停状态)、Stop状态(停止状态)以及Destroy状态(关闭)。

[0061] 其中，running状态，是指Activity组件处于可见并且可以和用户进行交互的状态。当Activity组件启动后，初始状态会处于running状态，此时会被放入在Activity堆栈的顶部，在系统前台(屏幕最前端)显示。

[0062] Pause状态，是指Activity组件被其它的非全屏或者半透明的Activity组件所覆盖，不再处于Activity堆栈顶部，但仍然处于可见并且可以和用户进行交互的状态。

[0063] Stop状态，是指Activity组件被其它的Activity组件完全覆盖，不再处于Activity堆栈顶部，该Activity组件的交互界面被完全被隐藏的状态。

[0064] Destroy状态，只指Activity组件被关闭销毁的状态。

[0065] 其中，以上四种运行状态可以通过用户针对该Activity组件的操作来进行转换。

[0066] 在Android系统中，当Activity组件启动后，如果用户通过操作启动了新的Activity组件，那么该新的Activity组件会被移入Activity堆栈顶部，对原有的该Activity组件进行覆盖，此时系统会调用onPause方法将原有的Activity组件转换为Pause状态。

[0067] 当该原有的Activity组件被完全覆盖后，此时系统会调用onStop方法将该原有的Activity组件转换为Stop状态。

[0068] 其中，当该原有的Activity组件处于Pause状态或者Stop状态时，系统会在系统内存不足的时候，通过调用onDestroy方法将该Activity组件转换为Destroy状态对该Activity组件执行关闭。

[0069] 基于此，在示出的一种实施方式中，系统在判断该Activity弹框组件是否被其它Activity组件覆盖时，可以通过判断该Activity弹框组件的onPause或者onStop方法是否被调用来实现。

[0070] 当该Activity弹框组件的onPause或者onStop方法被调用时，则可以确定当前该Activity弹框组件被其它Activity组件覆盖，不再处于堆栈顶部。

[0071] 步骤103，当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

[0072] 在本例中，当系统判断出该Activity弹框组件被其它Activity组件覆盖时，可以重新启动该Activity弹框组件。由于该Activity弹框组件已经被预配置为singleTask模式，因此当该Activity弹框组件被重新启动后，系统会将想要覆盖该Activity弹框组件的其它Activity组件弹出Activity堆栈，此时该Activity弹框组件将会被重新移至Activity堆栈顶部，在这种情况下，基于该Activity弹框组件唯一Activity实例输出的提示框将会重新在系统前台显示而不被覆盖。

[0073] 在示出的一种实施方式中，如果系统判断出该Activity弹框组件的onPause方法被调用，此时系统可以确认该Activity弹框组件被其它Activity组件覆盖。

[0074] 在这种情况下，可以在onPause方法中预先加载针对Activity弹框组件的重启逻辑；比如，开发人员可以在onPause()方法中编辑用于针对Activity弹框组件执行重启的执行代码。当系统调用该onPause方法后，则可以重新启动该Activity弹框组件。

[0075] 在示出的另一种实施方式中，如果系统判断出该Activity弹框组件的onStop方法被调用，此时系统可以确认该Activity弹框组件被其它Activity组件覆盖。其中，该onStop方法也可以预先加载针对Activity弹框组件的重启逻辑。

[0076] 在Android系统中，系统在针对onPause方法和onStop方法进行调用时，通常是先调用onPause方法再调用onStop方法(该调用顺序为Android系统的基本处理机制)；然而，在一些应用场景中，比如，并发弹框的场景(即当该Activity弹框组件输出提示框的同时，就有其它Activity组件对该Activity弹框组件输出的提示框进行覆盖)，系统可能会产生onPause方法和onStop方法调用顺序上的异常，出现在onPause方法未调用的情况下绕过onPause方法而直接调用onStop方法的情况。

[0077] 因此，针对这些异常的应用场景，为了避免通过onStop方法中加载的重启逻辑对该Activity弹框组件进行重复处理，系统在判断出该Activity弹框组件的onStop方法被调用时，可以进一步判断该Activity弹框组件的onPause方法是否已经被调用，如果该Activity弹框组件的onPause方法已经被调用，表明该Activity弹框组件已经通过执行onPause方法中加载的重启逻辑进行了处理，此时系统可以不再重复执行该onStop方法中加载的重启逻辑对该Activity弹框组件进行重复处理。

[0078] 当然，如果该Activity弹框组件的onPause方法未被调用，此时系统可能发生了调用异常，系统可以正常运行onStop方法中加载的重启逻辑，重新启动该Activity弹框组件。

[0079] 其中，需要指出的是，系统在判断该Activity弹框组件的onPause方法是否已经被调用时，可以通过在该Activity弹框组件的类文件中增加标记，然后通过检查类文件中是否存在对应的标记来实现。

[0080] 例如，在实现时，当系统调用onPause方法中加载的重启逻辑对该Activity弹框组件处理完成后，可以在该Activity弹框组件的类文件中增加一个标记，该标记用于表示该Activity弹框组件已经基于onPause方法中加载的重启逻辑处理完成。

[0081] 当系统判断出该Activity弹框组件的onStop方法被调用，可以进一步检查该Activity弹框组件的类文件中是否存在上述标记，来确认该Activity弹框组件的onPause方法是否已经被调用。如果类文件中存在上述标记，表明该Activity弹框组件已经基于onPause方法中加载的重启逻辑处理完成，不再需要重复处理。

[0082] 可见，通过这种方式，可以在onPause方法和onStop方法的调用顺序异常的应用场景中，避免通过onStop方法中加载的重启逻辑对该Activity弹框组件进行重复处理。

[0083] 以上实施例中，描述了系统通过判断Activity弹框组件的onPause或者onStop方法是否被调用，来确认Activity弹框组件是否其它Activity被覆盖，并在判断出Activity弹框组件被其它Activity组件覆盖时，通过执行onPause或者onStop方法加载的重启逻辑重新启动该Activity弹框组件，使得该Activity弹框组件在预配置为singleTask模式的情况下，可以被重新移至Activity堆栈顶部，将输出的提示框重新显示在系统前台不被覆盖的详细过程。

[0084] 然而，在Android系统中，系统除了会在Activity弹框组件被其它Activity组件被其它Activity组件覆盖时，调用onPause方法或者onStop方法以外，当用户通常执行正常的关闭事件，来关闭该Activity弹框组件时，系统也会调用onPause方法或者onStop方法。

[0085] 例如，当该Activity组件处于running状态时，如果系统监听到了用户针对该

Activity组件的关闭事件，系统会按照顺序调用onPause方法和onStop方法将该Activity组件依次装换为Pause状态和Stop状态，然后自动调用onDestroy方法将该Activity组件转换为Destroy状态对该Activity组件执行关闭。

[0086] 其中，上述关闭事件可以包括与用户针对该Activity弹框组件执行的关闭操作对应的事件；比如，以搭载Android系统的触屏智能手机为例，该关闭事件可以包括用户针对手机上的返回键的点击事件，即用户可以通过点击手机上的返回键，来关闭该Activity弹框组件。当手机系统在后台监听到用户针对返回键的点击事件时，可以按照顺序调用onPause方法和onStop方法将该Activity组件依次装换为Pause状态和Stop状态，然后自动调用onDestroy方法针对该Activity组件执行关闭。

[0087] 可见，在用户针对该Activity弹框组件进行正常关闭的应用场景中，由于对onPause方法或onStop方法的调用是由用户的正常关闭操作所触发，而并不是由该Activity弹框组件被其它Activity组件覆盖的事件所触发，因此如果系统仍然通过执行该Activity弹框组件的onPause方法或onStop方法中加载的重启逻辑，重新启动该Activity弹框组件，那么将会导致用户的误操作(即用户想要关闭该Activity弹框组件，但通过上述重启逻辑却重新启动了该Activity弹框组件)。

[0088] 因此，为了避免误操作，当系统判断出该Activity弹框组件的onPause方法或者onStop方法被调用时，系统在基于该onPause方法或者onStop方法中加载的重启逻辑重新启动该Activity弹框组件之前，可以进一步判断该onPause方法或者onStop方法的调用是否是由用户针对该Activity弹框组件的关闭事件所触发的。

[0089] 如果该onPause方法或者onStop方法的调用是由用户针对该Activity弹框组件的关闭事件所触发的，此时为针对该Activity弹框组件的正常关闭流程，系统可以允许该Activity弹框组件正常关闭，并按照针对该Activity弹框组件的正常关闭流程进行响应，自动调用Activity弹框组件的onDestroy方法关闭该ctivity弹框组件。

[0090] 当然，如果该onPause方法或者onStop方法的调用并不是由用户针对该Activity弹框组件的关闭事件所触发的，此时系统可以正常的执行onPause方法或者onStop方法中加载的重启逻辑，重新启动该Activity弹框组件，将该Activity弹框组件移至Activity堆栈的顶部。

[0091] 其中，需要指出的是，系统在判断上述onPause方法或者onStop方法的调用是否是由用户针对该Activity弹框组件的关闭事件所触发的，仍然可以通过在该Activity弹框组件的类文件中增加标记，然后通过检查类文件中是否存在对应的标记来实现。

[0092] 例如，在实现时，当系统监听到用户针对该Activity弹框组件的关闭事件，调用onPause方法以及onStop方法后，可以在该Activity弹框组件的类文件中增加一个标记，该标记用于表示本次onPause方法以及onStop方法的调用是由用户针对该Activity弹框组件的关闭事件触发的。

[0093] 当系统判断出onPause方法或者onStop方法被调用，可以进一步检查该Activity弹框组件的类文件中是否存在上述标记。如果类文件中存在上述标记，则表明本次onPause方法以及onStop方法的调用是由用户针对该该Activity弹框组件的关闭事件触发，此时系统可以按照针对该Activity弹框组件的正常关闭流程进行响应即可。

[0094] 可见，通过这种方式，可以在用户针对该Activity弹框组件执行正常关闭的应用

场景中，避免由于系统通过执行该Activity弹框组件的onPause方法或onStop方法中加载的重启逻辑，重新启动该Activity弹框组件，而导致的误操作的问题。

[0095] 以下结合具体的应用场景对以上实施例中的技术方案进行详细描述。

[0096] 在本例中示出的应用的场景可以包括单点登录的应用场景，以及呼叫来电接听的应用场景。

[0097] 一方面，在示出的单点登录的应用场景中，上述弹框提示的触发事件可以包括单点登录事件。在该应用场景中，用户可以使用登录账号在终端上登录，与该终端对应的登录服务端可以对该登录账号进行验证，当登录服务端对该登录账号验证通过后，此时用户成功登录该终端。

[0098] 当用户使用相同登录账号在其它终端上登录时，登录服务端会向当前的登录终端发送一个通知消息，触发当前的登录终端退出登录状态。当前的登录终端在接收到服务端发送的该通知消息后，可以认为监听到了单点登录事件，此时可以退出登录状态，并面向用户输出一个单点登录提示框，在该提示框中可以填充用于提示本次退出登录状态的原因的提示文案。

[0099] 系统在通过启动Activity弹框组件输出上述单点登录提示框时，为了避免输出的该提示框被其它提示框或者页面覆盖，可以默认将该Activity弹框组件的运行状态设置为singleTask模式。

[0100] 当该Activity弹框组件被其它Activity组件覆盖时，系统会调用onPause方法或者onStop方法，其中在onPause方法或者onStop方法中预先加载了重启逻辑，系统通过执行该onPause方法或者onStop方法的重启逻辑后，可以重新启动该Activity弹框组件。

[0101] 由于该Activity弹框组件被预配置为singleTask模式，因此当该Activity弹框组件被重新启动后，系统会将想要覆盖该Activity弹框组件的其它Activity组件弹出Activity堆栈，该Activity弹框组件会被重新移至Activity堆栈顶部。在这种情况下，上述单点登录提示框将会重新在系统前台显示，而不被其它提示框或者页面覆盖，而且由于想要覆盖该Activity弹框组件的其它Activity组件被直接弹出Activity堆栈，该其它Activity组件所对应的交互界面将无法显示出来，对于用户来说无法感知到页面的变化。

[0102] 可见，在单点登录的应用场景中，通过这种方式，可以避免由于单点登录提示框被覆盖，而造成的提示信息无法及时的提示给用户的问题。

[0103] 另一方面，在示出的呼叫来电接听的应用场景中，上述弹框提示的触发事件可以包括呼叫来电事件。在该应用场景中，当终端的系统监听到呼叫来电时，可以向用户输出一个来电提示框提示用户接收到了来电，在该提示框中可以填充用于提示本次来电呼叫发起方的提示文案。用户可以针对该提示框进行相应的操作，比如向上滑动或者向下滑动，来接听该来电或者挂断该来电。

[0104] 在这种场景中，如果系统在接收到呼叫来电时所输出的来电提示框，被其它提示框或者页面所覆盖，那么可能导致用户无法针对来电提示框进行操作，来接听该来电，甚至还可能会导致该来电提示框被覆盖时，终端的按键(比如Home键)失效等兼容性的问题。

[0105] 系统在通过启动Activity弹框组件输出上述来电提示框时，为了避免输出的该提示框被其它提示框或者页面覆盖，可以默认将该Activity弹框组件的运行状态设置为singleTask模式。

[0106] 当该Activity弹框组件被其它Activity组件覆盖时，系统会调用onPause方法或者onStop方法，其中在onPause方法或者onStop方法中预先加载了重启逻辑，系统通过执行该onPause方法或者onStop方法的重启逻辑后，可以重新启动该Activity弹框组件。

[0107] 由于该Activity弹框组件被预配置为singleTask模式，因此当该Activity弹框组件被重新启动后，系统会将想要覆盖该Activity弹框组件的其它Activity组件弹出Activity堆栈，该Activity弹框组件会被重新移至Activity堆栈顶部。在这种情况下，上述来电提示框将会重新在系统前台显示，而不被其它提示框或者页面覆盖，而且由于想要覆盖该Activity弹框组件的其它Activity组件被直接弹出Activity堆栈，该其它Activity组件所对应的交互界面将无法显示出来，对于用户来说无法感知到页面的变化。

[0108] 可见，在呼叫来电接听的应用场景中，通过这种方式，可以避免由于来电提示框被覆盖，而造成的用户无法接听来电，以及导致的终端按键的兼容性的问题。

[0109] 当然，除了以上示出的应用场景以外，以上实施例中的技术方案还可以应用在其它的类似场景中，即在实际应用中，以上实施例中的技术方案所适用的应用场景，除了以上示出的应用场景以外，还可以包括所有需要面向用户输出弹框提示，并且输出的提示框被其它提示框或者页面覆盖时，可能会对系统中的APP或者基本功能的使用造成影响的应用场景，本申请不再一一列举。

[0110] 在以上实施例中，通过在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；判断该Activity弹框组件是否被其它Activity组件覆盖；当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示；由于在本申请中，Activity弹框组件为单任务模式，因此当该Activity弹框组件被重新启动后，可以立即被移至Activity堆栈顶部，从而在启动Activity弹框组件向用户输出了提示框后，可以确保该提示框不被覆盖，始终在系统前台显示。

[0111] 同时，由于本申请不再基于Android系统中的Window组件进行弹框，不再支持用户对弹框进行设置，因此可以避免由于用户通过手动设置在系统中将弹框禁掉导致无法进行弹框的问题。

[0112] 另外，由于本申请通过Activity弹框组件进行弹框，因此可以实现将系统中任意的Activity组件作为主体，在任意页面中来进行弹框，从而具有很高的灵活性。

[0113] 与上述方法实施例相对应，本申请还提供了装置的实施例。

[0114] 请参见图2，本申请提出一种基于Android的弹框提示装置20，应用于终端；其中，请参见图3，作为承载所述基于Android的弹框提示装置20的终端所涉及的硬件架构中，通常包括CPU、内存、非易失性存储器、网络接口以及内部总线等；以软件实现为例，所述基于Android的弹框提示装置20通常可以理解为加载在内存中的计算机程序，通过CPU运行之后形成的软硬件相结合的逻辑装置，所述装置20包括：

[0115] 启动模块201，用于在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式；

[0116] 判断模块202，用于判断该Activity弹框组件是否被其它Activity组件覆盖；

[0117] 重启模块203，用于在该Activity弹框组件被其它Activity组件覆盖时，重新启动

该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示。

[0118] 在本例中，所述判断模块202具体用于：

[0119] 判断所述Activity弹框组件的onPause或者onStop方法是否被调用；

[0120] 当所述Activity弹框组件的onPause或者onStop方法被调用时，确定该Activity弹框组件被其它Activity组件覆盖。

[0121] 在本例中，所述重启模块203具体用于：

[0122] 当所述Activity弹框组件的onPause方法被调用，基于所述onPause方法重新启动该Activity弹框组件；

[0123] 当所述Activity弹框组件的onStop方法被调用时，判断所述Activity弹框组件的onPause方法是否已被调用，如果所述onPause方法未被调用，则基于所述onStop方法重新启动该Activity弹框组件。

[0124] 在本例中，所述重启模块203进一步用于：

[0125] 判断所述onPause方法或者所述onStop方法的调用是否是由针对所述Activity弹框组件的关闭事件所触发；

[0126] 如果所述onPause方法或者所述onStop方法的调用不是由针对所述Activity弹框组件的关闭事件所触发，则基于所述onPause方法或者所述onStop方法重新启动该Activity弹框组件。

[0127] 在本例中，所述装置20还包括：

[0128] 关闭模块204，用于在所述onPause方法或者所述onStop方法的调用是由针对所述Activity弹框组件的关闭事件所触发时，针对所述Activity弹框组件执行正常关闭。

[0129] 本领域技术人员在考虑说明书及实践这里公开的发明后，将容易想到本申请的其它实施方案。本申请旨在涵盖本申请的任何变型、用途或者适应性变化，这些变型、用途或者适应性变化遵循本申请的一般性原理并包括本申请未公开的本技术领域中的公知常识或惯用技术手段。说明书和实施例仅被视为示例性的，本申请的真正范围和精神由下面的权利要求指出。

[0130] 应当理解的是，本申请并不局限于上面已经描述并在附图中示出的精确结构，并且可以在不脱离其范围进行各种修改和改变。本申请的范围仅由所附的权利要求来限制。

[0131] 以上所述仅为本申请的较佳实施例而已，并不用以限制本申请，凡在本申请的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本申请保护的范围之内。

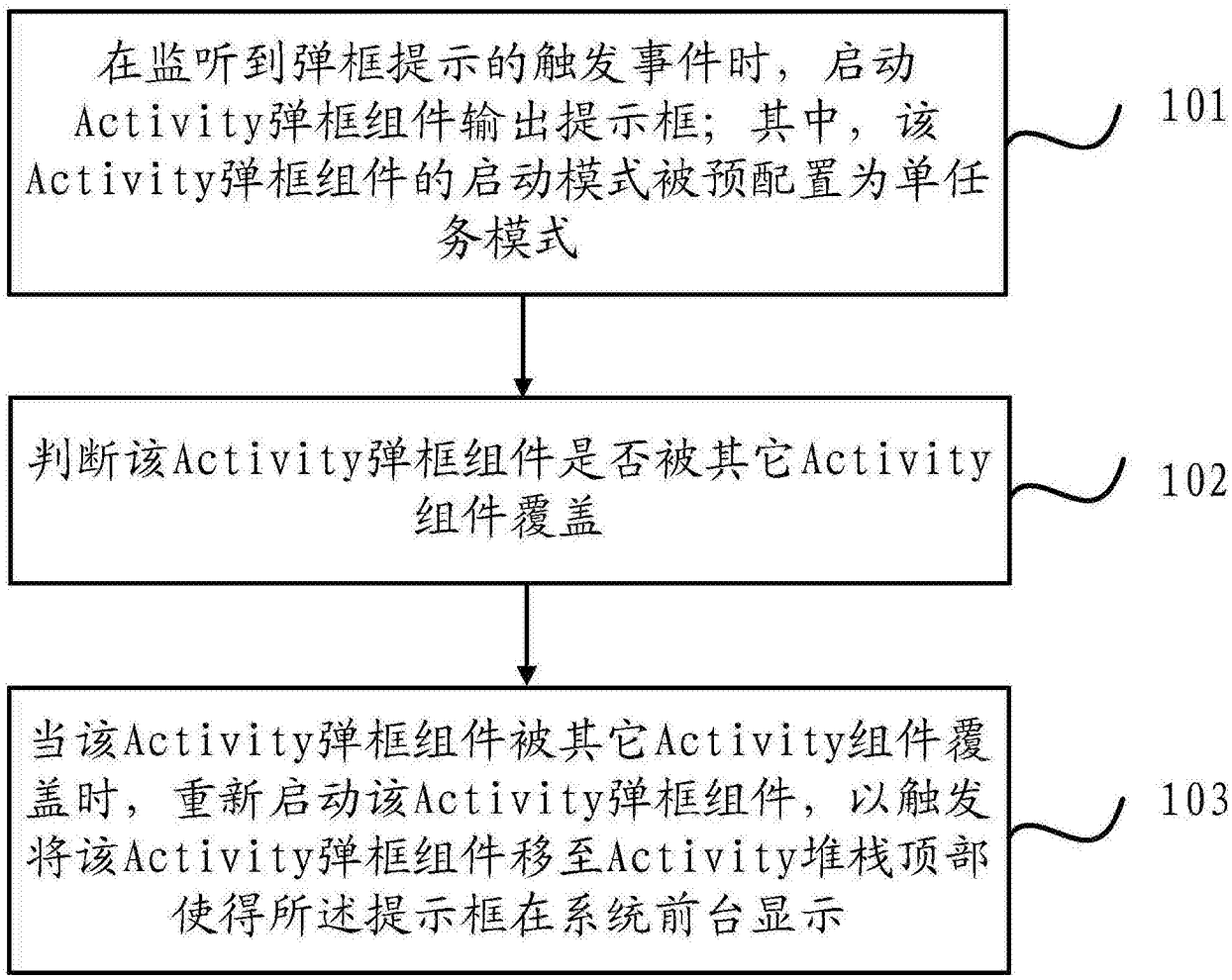

在监听到弹框提示的触发事件时，启动Activity弹框组件输出提示框；其中，该Activity弹框组件的启动模式被预配置为单任务模式
101
判断该Activity弹框组件是否被其它Activity组件覆盖
102
当该Activity弹框组件被其它Activity组件覆盖时，重新启动该Activity弹框组件，以触发将该Activity弹框组件移至Activity堆栈顶部使得所述提示框在系统前台显示
103

图1

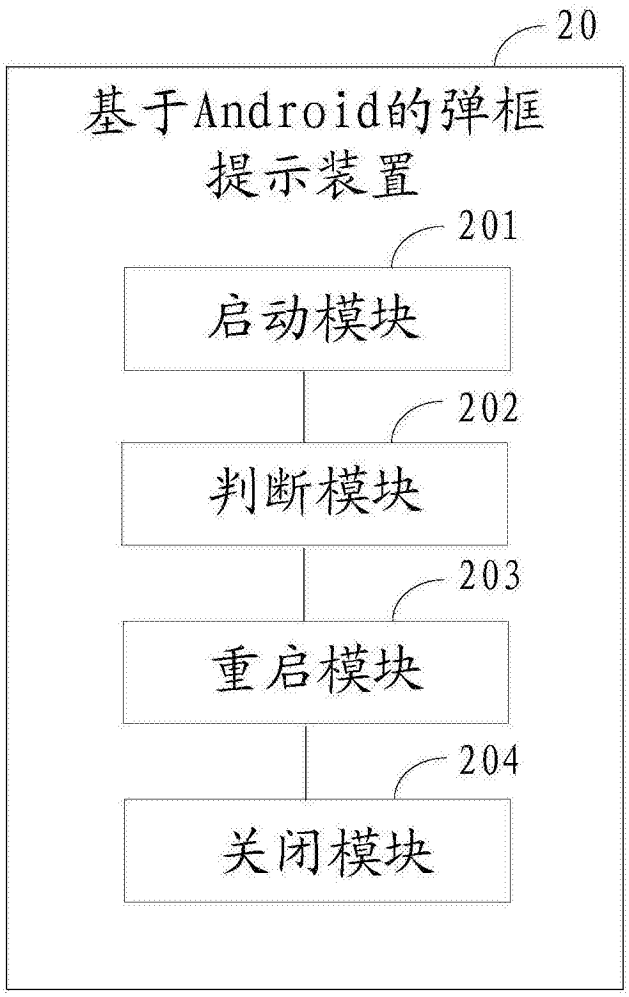


图2

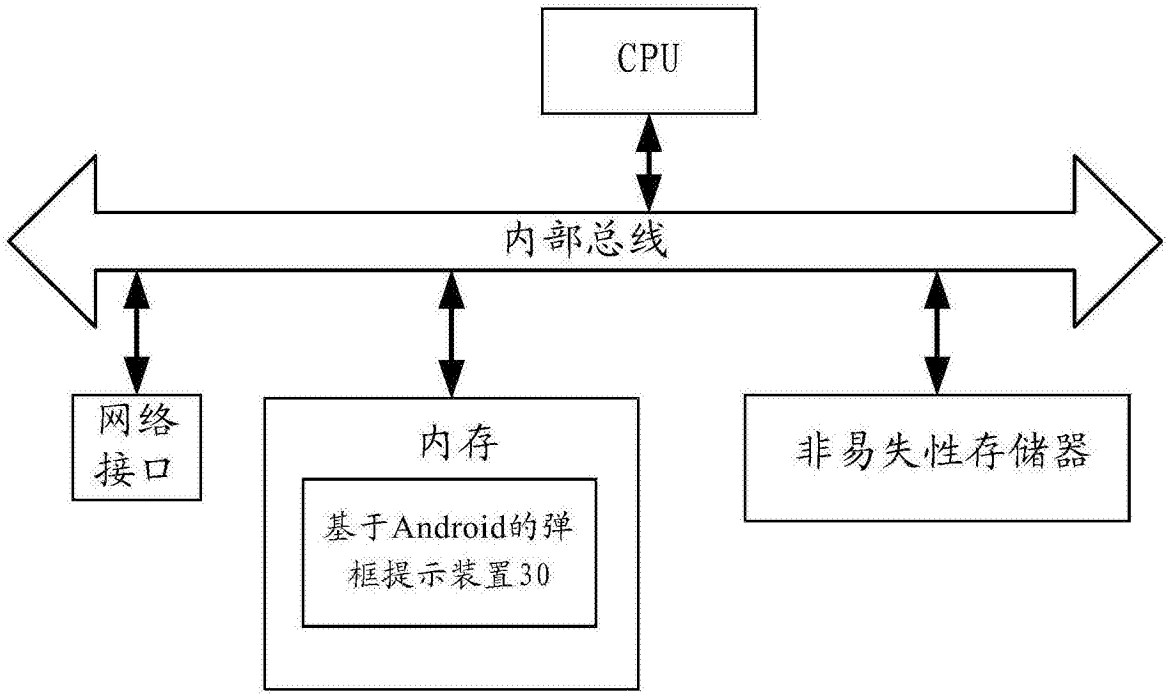


图3