27 DE SEPTIEMBRE DE 2024

INVESTIGACION

SAMUEL ELÍ DELGADO RODRÍGUEZ

Cómo enviar al usuario a otra app

Una de las funciones más importantes de Android es la capacidad que tiene una app de enviar al usuario a otra en función de una "acción" que le gustaría realizar. Por ejemplo, si tu app tiene la dirección de una empresa que quieres mostrar en un mapa, no necesitas compilar una actividad en tu app para mostrar el mapa. En su lugar, puedes crear una solicitud para ver la dirección mediante un Intent. Luego, el sistema Android inicia una app que puede mostrar la dirección en un mapa.

Como se explicó en la primera clase, Cómo compilar tu primera app, debes usar intents para navegar entre las actividades de tu propia app. En general, *debes utilizar un intent explícito*, que define el nombre exacto de la clase de componente que deseas iniciar. Sin embargo, si quieres que otra app realice una acción, como "mostrar un mapa", debes usar un *intent implícito*.

En esta lección, se muestra cómo crear un intent implícito para una acción determinada y cómo utilizarlo con el objetivo de iniciar una actividad que realice la acción en otra app. Además, mira el video incorporado a fin de entender por qué es importante que incluyas verificaciones del tiempo de ejecución para tus intents implícito

Cómo crear un intent implícito

Los intents implícitos no declaran el nombre de clase del componente que se iniciará, sino que declaran una acción que se llevará a cabo. La acción especifica lo que deseas hacer, por ejemplo, *ver*, *editar*, *enviar* u *obtener* algo.

Cómo asociar acciones de intent con datos

Los intents también suelen incluir datos asociados con la acción, como la dirección que quieres ver o el mensaje de correo electrónico que quieres enviar. Según el intent que quieras crear, los datos pueden ser un Uri o uno de varios tipos de datos diferentes. También es posible que el intent no necesite ningún dato.

Si tus datos son un Uri, existe un simple constructor Intent() que puedes usar para definir la acción y los datos.

Por ejemplo, se muestra la manera de crear un intent a fin de iniciar una llamada telefónica usando los datos de un Uri para especificar el número de teléfono:

Kotlin Java

```kotlin
val callIntent: Intent = Uri.parse("tel:5551234").let { number ->
    Intent(Intent.ACTION_DIAL, number)
}
```

Cuando tu app invoca este intent llamando a startActivity(), la app de teléfono inicia una llamada al número de teléfono especificado.

Ver un mapa

Kotlin Java

```kotlin
// Map point based on address
val mapIntent: Intent = Uri.parse(
        "geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California"
).let { location ->
    // Or map point based on latitude/longitude
    // val location: Uri = Uri.parse("geo:37.422219,-122.08364?z=14") // z param is zoom level
    Intent(Intent.ACTION_VIEW, location)
}
```

Ver una página web

Kotlin Java

```kotlin
val webIntent: Intent = Uri.parse("https://www.android.com").let { webpage ->
    Intent(Intent.ACTION_VIEW, webpage)
}
```

Cómo agregar objetos adicionales a un intent

Otros tipos de intents implícitos requieren datos "adicionales" que proporcionan diferentes tipos de datos, como una string. Puedes agregar uno o más datos adicionales usando los diferentes métodos putExtra().

De forma predeterminada, el sistema define el tipo de MIME requerido por un intent según los datos de Uri incluidos. Si no incluyes un Uri en el intent, deberás utilizar un elemento setType() para especificar el tipo de datos relacionados con el intent. Cuando se configura el tipo de MIME, también se especifica qué tipos de actividades deben recibir el intent.

Cómo crear un evento de calendario

Nota: Este intent para un evento de calendario se admite solamente con el nivel de API 14 o versiones superiores.

Kotlin Java

```kotlin
// Event is on January 23, 2021 -- from 7:30 AM to 10:30 AM.
Intent(Intent.ACTION_INSERT, Events.CONTENT_URI).apply {
    val beginTime: Calendar = Calendar.getInstance().apply {
        set(2021, 0, 23, 7, 30)
    }
    val endTime = Calendar.getInstance().apply {
        set(2021, 0, 23, 10, 30)
    }
    putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, beginTime.timeInMillis)
    putExtra(CalendarContract.EXTRA_EVENT_END_TIME, endTime.timeInMillis)
    putExtra(Events.TITLE, "Ninja class")
    putExtra(Events.EVENT_LOCATION, "Secret dojo")
}
```

Nota: Es importante que definas un **`Intent`** lo más específico posible. Por ejemplo, si deseas mostrar una imagen usando el intent **`ACTION_VIEW`**, debes especificar un MIME de tipo **`image/*`**, lo que evita que el intent active las apps que pueden "ver" otros tipos de datos (como una app de mapas).

Cómo iniciar una actividad con el intent

Una vez que hayas creado tu Intent y establecido la información adicional, invoca

Cómo enviar un correo electrónico con un archivo adjunto

Kotlin Java

```
startActivity(intent)
```

```
        type = "text/plain"
        putExtra(Intent.EXTRA_EMAIL, arrayOf("jan@example.com")) // recipients
        putExtra(Intent.EXTRA_SUBJECT, "Email subject")
        putExtra(Intent.EXTRA_TEXT, "Email message text")
        putExtra(Intent.EXTRA_STREAM, Uri.parse("content://path/to/email/attachment"))
        // You can also attach multiple items by passing an ArrayList of Uris
    }
```

a startActivity() para enviar el elemento al sistema.

Cómo procesar una situación en la que ninguna app puede recibir un intent

Aunque varios intents se procesan correctamente mediante otra app instalada en el dispositivo (como una aplicación de teléfono, correo electrónico o calendario), la tuya debe prepararse para la situación en la que ninguna actividad pueda controlar el intent de tu app. Siempre que invoques un intent, debes tener todo listo para capturar una ActivityNotFoundException, que se produce si no hay ninguna otra actividad que pueda procesar el intent de tu app.

Kotlin Java

```
try {
    startActivity(intent)
} catch (e: ActivityNotFoundException) {
    // Define what your app should do if no activity can handle the intent.
}
```

Diálogo de desambiguación.

Si el sistema identifica más de una actividad que puede procesar el intent, se mostrará un diálogo (a veces denominado "diálogo de desambiguación") para que el usuario seleccione qué app usar, como se muestra en la figura 1. Si existe una única actividad que puede procesar el intent, el sistema la iniciará de inmediato.

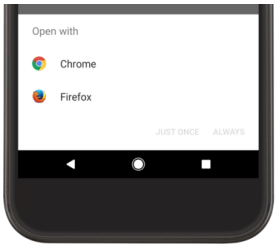

Figura 1: Ejemplo del diálogo de selección que aparece cuando más de una app puede procesar un intent

Ejemplo completo

Se incluye un ejemplo completo que muestra cómo crear un intent para ver un mapa, verificar que exista una app para controlar el intent y luego iniciarlo:

Kotlin Java

```kotlin
// Build the intent.
val location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California")
val mapIntent = Intent(Intent.ACTION_VIEW, location)

// Try to invoke the intent.
try {
    startActivity(mapIntent)
} catch (e: ActivityNotFoundException) {
    // Define what your app should do if no activity can handle the intent.
}
```

Muestra un selector de apps

Ten presente que, cuando inicias una actividad transfiriendo tu Intent a startActivity() y existe más de una app que responde al intent, el usuario puede seleccionar la app que se usará de forma predeterminada (seleccionando la casilla de verificación de la parte inferior del diálogo; consulta la figura 1). Esto es útil cuando se realiza una acción para la que el usuario quiere utilizar siempre la misma app, por ejemplo, cuando se abre una página web (los usuarios suelen usar un solo navegador web) o se toma una fotografía (suelen preferir una cámara).

Sin embargo, si la acción que se realizará se puede controlar mediante varias apps y el usuario quizás prefiere utilizar una diferente cada vez, por ejemplo, una acción de "compartir" un elemento, debes mostrar explícitamente un diálogo de selección, como se indica en la figura. El diálogo de selección obliga al usuario a seleccionar qué app quiere utilizar para la acción en cada caso (no puede elegir una app predeterminada para la acción).

Para mostrar el diálogo de selección, crea un Intent usando createChooser() y transfiérelo a startActivity(). Por ejemplo:

Kotlin Java

```kotlin
val intent = Intent(Intent.ACTION_SEND)

// Create intent to show chooser
val chooser = Intent.createChooser(intent, /* title */ null)

// Try to invoke the intent.
try {
    startActivity(chooser)
} catch (e: ActivityNotFoundException) {
    // Define what your app should do if no activity can handle the intent.
}
```

Se mostrará un diálogo con una lista de apps que responden al intent que se pasó al método `createChooser()`. Se puede proporcionar el parámetro `title` si la acción no es `ACTION_SEND` ni `ACTION_SEND_MULTIPLE`.