Intents e content providers

Programação para dispositivos móveis

Prof. Allan Rodrigo Leite

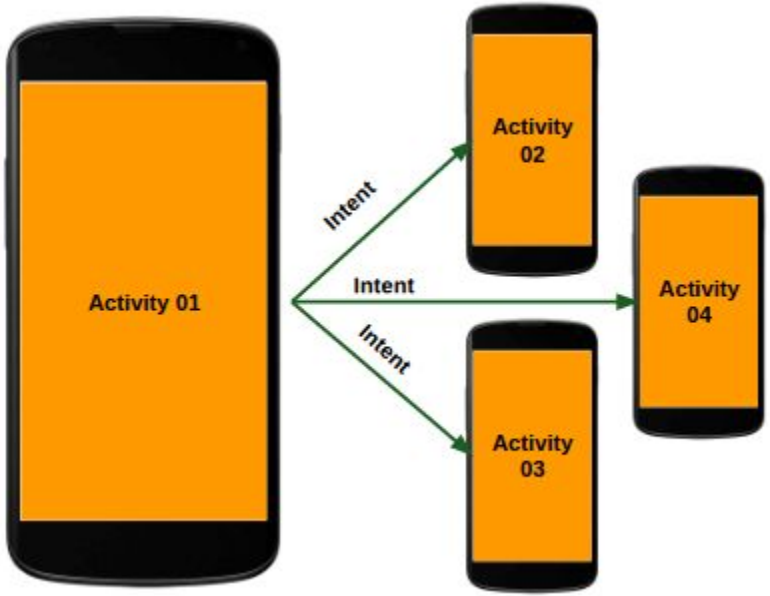
Intents

- São mensagens assíncronas trocadas entre componentes
  - Possibilita a interação entre componentes de um aplicativo
    - Componentes do mesmo aplicativo
    - Componentes contribuídos por outros aplicativos

- Uma Intent pode conter dados dentro dela através de um Bundle
  - Dados serão utilizados pelo componente que está recebendo a Intent
  - Mecanismo bastante utilizado para comunicação entre activities

Intents

- Exemplo de interação entre activities



Estrutura de uma Intent

- Uma Intent é constituída por
  - Nome do componente
    - Se houver nome do componente, a Intent é explícita, do contrário é implícita
  - Ação
    - Define qual ação genérica a ser realizada
  - Dados
    - Define os dados a serem compartilhados entre componentes
  - Categoria
    - Informações do tipo de componente que deve processá-la
  - Extra
    - Pares de valor-chave que carregam informações adicionais

Intents explícitas

- Define explicitamente o componente que deve ser chamado
  - Usa-se a classe Java como identificador
  - São normalmente usadas dentro do próprio aplicativo
    - Visto que as classes em um aplicativo são controladas pelo mesmo

- Ao criar uma nova Intent deve ser fornecido os seguintes parâmetros
  - Contexto de origem: dados sobre a intenção
  - Recurso de destino: pode ser uma activity quanto um service

Intents explícitas

- Os seguintes métodos iniciam uma activity ou service via Intent
  - `startActivity`
  - `startService`

```
Intent intent = new Intent(this, OtherActivity.class);
startActivity(intent);
```

Intents explícitas

- Opcionalmente uma Intent também pode conter dados adicionais
  - Os dados podem
    - Ser fornecidos por meio de uma instância de `Bundle`
    - Diretamente fornecidos na Intent (extra data)

```
Intent intent = new Intent(this, OtherActivity.class);

Bundle bundle = new Bundle();
bundle.putString("message", "Olá mundo");

intent.putExtra("messageBundle", bundle);
intent.putExtra("messageIntent", "Olá mundo com intent");


startActivity(intent);
```

Intents explícitas

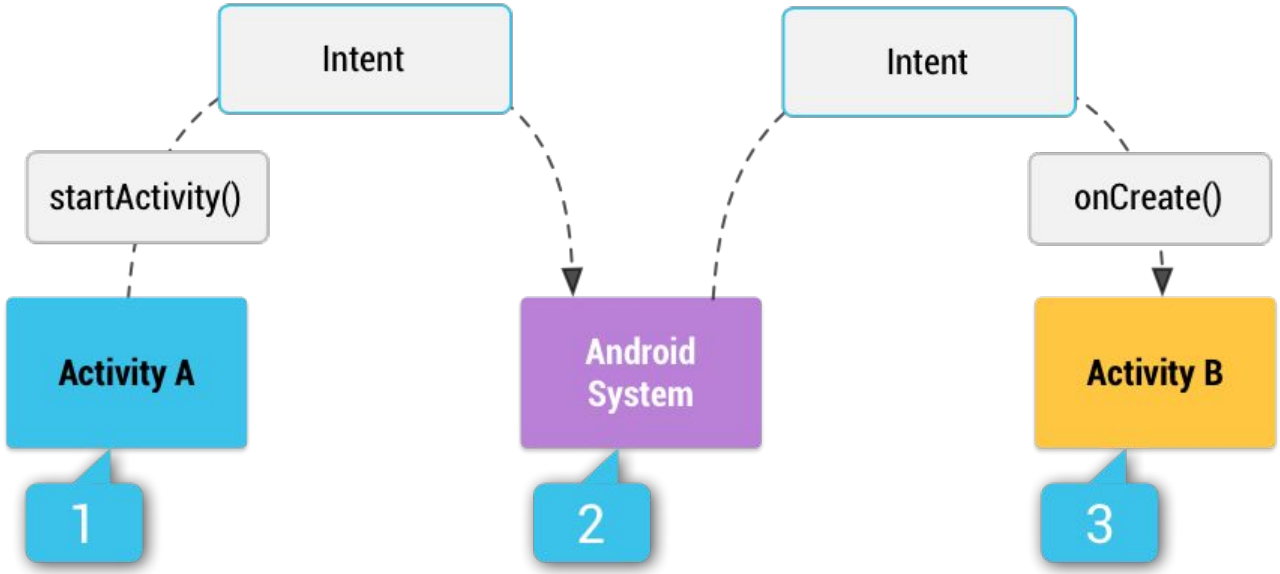
- Recuperando dados na invocação de uma Intent

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_other);

  Intent intent = getIntent();
  Bundle bundle = intent.getBundleExtra("messageBundle");
  String message = bundle.getString("message");
  String messageIntent = intent.getStringExtra("messageIntent");
}
```

Intents implícitas

- Não nomeiam nenhum componente específico
  - Declaram apenas uma ação geral a realizar
    - Permite que um componente de outro aplicativo processe a Intent



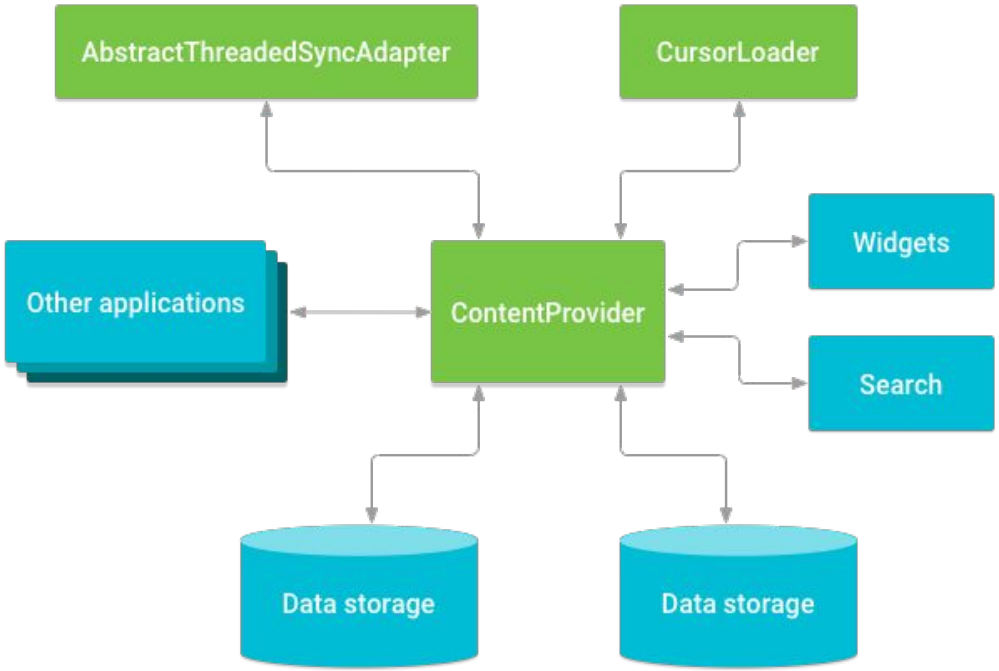
Intents implícitas

- Requer que o aplicativo que irá processar a Intent declare um filtro
  - Também chamado de `intent-filter`
  - Configuração é realizada no `AndroidManifest.xml`

```
<activity
    android:name="org.udesc.otes06.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="com.org.otes06.OtherActivity" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
<activity>
```

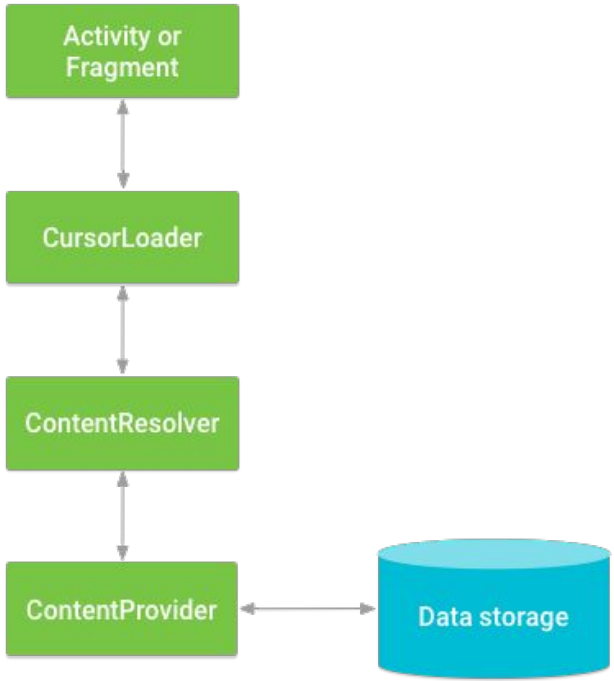
Content providers

- Fornece um mecanismo para acessar dados de outros aplicativos
  - Compartilhamento de dados ou prover dados para outros aplicativos
  - Envio de dados a um widget
  - Sincronização dos dados do aplicativo com seu servidor



Acessando um content providers

- O provedor de conteúdo coordena o acesso à camada de dados
  - Usa-se o `ContentResolver` para se comunicar como cliente
  - Em seguida é obtida a instância de `ContentProvider` do provedor
  - Fornece funções básicas para manipulação de dados
    - CRUD (create, retrieve, update e delete)



Acessando um content providers

- Obter lista de contatos de outro aplicativo
  - Necessário informar que o aplicativo requer permissão aos contatos

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="org.udesc.otes06">

    <uses-permission android:name="android.permission.READ_CONTACTS">
    </uses-permission>

    <application> ... </application>

</manifest>
```

Acessando um content providers

- Todo projeto de aplicativo tem um arquivo `AndroidManifest.xml`
  - Descreve informações essenciais sobre o aplicativo para
    - Ferramentas de compilação do Android
    - Sistema operacional Android
    - Informações para a Google Play

  - Os aplicativos precisam solicitar permissão de acesso aos dados
    - Contatos, SMS, chamadas, etc
    - As permissões são identificadas por um código exclusivo

Acessando um content providers

- Solicitações de permissão são realizadas pelo próprio aplicativo
  - Princípios básicos
    - Solicite uma permissão no contexto, isto é, quando o usuário começar a interagir com o recurso que requer a permissão
    - Não bloqueie o usuário, ofereça a opção de cancelar este fluxo explicando a necessidade da permissão pelo aplicativo
    - Se o usuário negar ou revogar uma permissão necessária para um recurso, desative o recurso que requer a permissão

  - Para conferir se o usuário concedeu a permissão, usa-se o método `ContextCompat.checkSelfPermission()`
    - Sucesso retorna `PERMISSION_GRANTED`
    - Insucesso retorna `PERMISSION_DENIED`

Acessando um content providers

- Solicitações de permissão são realizadas pelo próprio aplicativo
  - Observe que o método para solicitar permissão não é blocante

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  checkContactsReadPermission();
  //...
}

protected void checkContactsReadPermission() {
  String permission = Manifest.permission.READ_CONTACTS;
  int granted = PackageManager.PERMISSION_GRANTED;

  if (ContextCompat.checkSelfPermission(this,permission) != granted) {
    ActivityCompat.requestPermissions(this, new String[] { permission }, 0);
  }
}
```

Acessando um content providers

- Solicitações de permissão são realizadas pelo próprio aplicativo
  - Observe que o método para solicitar permissão não é blocante

```
@Override
protected void onResume() {
  super.onResume();

  String permission = Manifest.permission.READ_CONTACTS;
  int granted = PackageManager.PERMISSION_GRANTED;

  button.setEnabled(ContextCompat.checkSelfPermission(this,permission) == granted);
}
```

Acessando um content providers

- Dados do provider são obtidos pela classe `Cursor`
  - Mas antes é necessário resolver o endereço do serviço de dados
  - Usa-se a classe `ContentResolver` para este propósito
    - Aplicativos padrão (contatos, SMS, ligações, etc.) possuem seus respectivos endereços já definidos pelo SDK do Android

```
ContentResolver contentResolver = getContentResolver();
Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;

Cursor cursor = contentResolver.query(uri, null, null, null, null);

if (cursor.getCount() > 0) {
  while (cursor.moveToNext()) {
    // Iterando sobre os contatos
  }
}
```

Acessando um content providers

- Classe `Cursor`
  - Similar a um cursor de uma consulta em um banco de dados
  - Coleção de dados com métodos para navegação sobre eles

```
if (cursor.getCount() > 0) {
  while (cursor.moveToNext()) {
    String name = cursor.getString(
      cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));

    String number = cursor.getString(
      cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));

    // Faz algo com os dados
  }
}
```

Consumindo API REST

- Primeiramente é necessário habilitar uso da Internet pelo aplicativo
  - Sem isto o aplicativo não poderá acessar recursos da Internet
  - Configuração é realizada no `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="org.udesc.otes06">

    <uses-permission android:name="android.permission.INTERNET">
    </uses-permission>

    <application> ... </application>

</manifest>
```

Consumindo API REST

- Uso de bibliotecas adicionais
  - Adição de bibliotecas realizada no `build.gradle`
    - GSON: responsável por converter objetos em JSON e vice-versa
    - OkHttp: responsável por realizar requisições HTTP

```
dependencies {
    ...

    implementation 'com.google.code.gson:gson:2.10.1'
    implementation 'com.squareup.okhttp3:okhttp:4.11.0'

    ...
}
```

Consumindo API REST

- Realizando a requisição HTTP para um serviço de CEP

```
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
    .url("https://viacep.com.br/ws/89221140/json/")
    .build();

Call call = client.newCall(request);
Response response = call.execute();

Gson gson = new Gson();
Map map = gson.fromJson(response.body().string(), Map.class);
```

Intents e content providers

Programação para dispositivos móveis

Prof. Allan Rodrigo Leite