


Android

Disponibilizando Informações com Content Providers




Tópicos Abordados

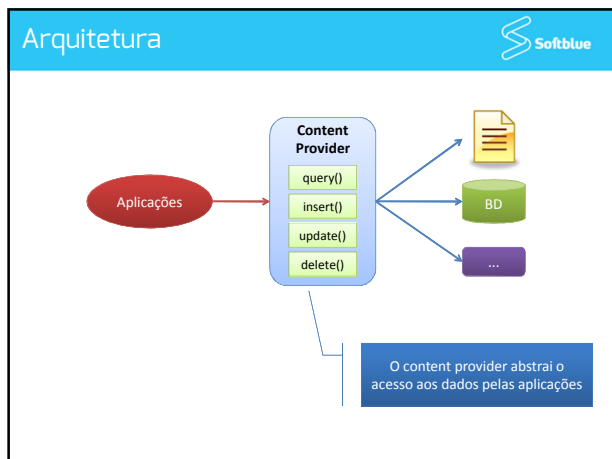


- O que são content providers
- Modelo de dados
- URI
- Convenções
- Consultando um content provider
- Executando queries e modificando dados
- Criando content providers
- Content providers nativos do Android
- Loaders
- Adapters e Cursores

O Que São



- Os content providers são uma maneira de expor dados de uma forma uniforme para diversas aplicações
 - São a forma correta para compartilhar dados entre aplicações no Android
- Permitem consultar, inserir, alterar e excluir dados

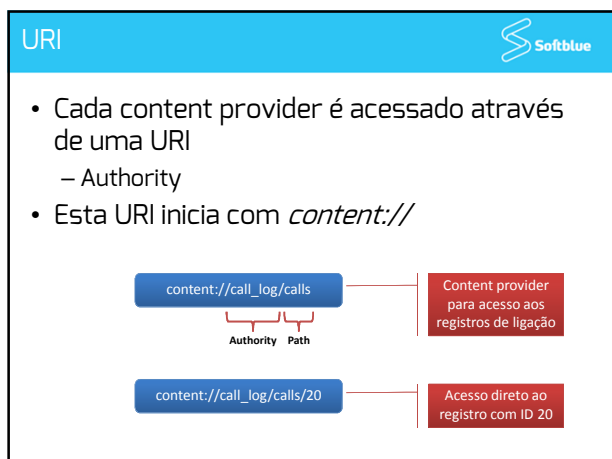


Modelo de Dados

- Os dados retornados por um content provider são retornados em forma de uma tabela
- Semelhante ao que acontece com consultas a bancos de dados

_ID	DATE	DURATION	NUMBER	TYPE
1	01/01/11 13:00:00	120	1234 5678	OUTGOING_TYPE
2	01/01/11 13:30:00	60	3344 5566	MISSED_TYPE
3	01/01/11 15:50:00	15	6677 2222	INCOMING_TYPE

Uma query retorna um objeto do tipo **Cursor**, que pode ser usado para navegar pelos registros



Convenções



- Quem vai utilizar o content provider precisa conhecer detalhes sobre ele para poder acessá-lo da forma correta
 - Authority
 - Nomes das colunas disponíveis
 - Tipos de dados das colunas
- Existem algumas convenções para facilitar este trabalho de acesso

Convenções



- A montagem da URI é feita com base no pacote da sua aplicação
- A URI é normalmente exposta como uma constante pública *CONTENT_URI*

```
public static final Uri CONTENT_URI =  
    Uri.parse("content://com.mypackage.provider.MyApplication");
```

- Também são criadas constantes para expor o nome das colunas que podem ser acessadas pelo cursor
 - Classe de contrato
- Tudo deve estar bem documentado

Executando Queries




- Para consultar um content provider é preciso executar queries
 - Bastante semelhante a queries em bancos de dados

```
Cursor c = cr.query(  
    CallLog.Calls.CONTENT_URI,  
    new String[]{ CallLog.Calls.NUMBER, CallLog.Calls.DURATION },  
    CallLog.Calls.NUMBER + " LIKE ?",  
    new String[]{ "%%" },  
    CallLog.Calls.DURATION + " DESC");
```

```
while (c.moveToNext()) {  
    String number = c.getString(c.getColumnIndex(CallLog.Calls.NUMBER));  
    long seconds = c.getLong(c.getColumnIndex(CallLog.Calls.DURATION));  
}
```


Buscando Registros por ID



- Para buscar registros por ID, basta adicioná-lo no final do path da URI

```
Uri.parse("content://call_log/calls/40");
Uri.withAppendedPath(CallLog.Calls.CONTENT_URI, "40");
ContentUris.withAppendedId(CallLog.Calls.CONTENT_URI, 40);
```

Modificação de Dados



- Os dados a serem modificados devem ser colocados em um objeto **ContentValues**

```
ContentValues values = new ContentValues();
values.put("CAMPO1", 10);
values.put("CAMPO2", "algo");

ContentResolver resolver = getContentResolver();
Uri newUri = resolver.insert(Application.CONTENT_URI, values);
```

Retorna a URI do registro criado

Insere um registro

Modificação de Dados



```
ContentValues values = new ContentValues();
values.put("CAMPO1", 10);
values.put("CAMPO2", "algo");

ContentResolver resolver = getContentResolver();
int rows = resolver.update(
    Application.CONTENT_URI, values, "CAMPO3 = ?", new String[] { "2" }, null);
```

Retorna o número de linhas alteradas

Atualiza registros

```
Uri uri = ContentUris.withAppendedId(Application.CONTENT_URI, 10);
ContentResolver resolver = getContentResolver();
int rows = resolver.delete(uri, null, null);
```

Retorna o número de linhas excluídas

Exclui registros

Criando Content Providers



- A criação de um content provider deve passar por três etapas
 - Especificar a estrutura e local do armazenamento dos dados
 - Normalmente em tabelas de um banco de dados
 - Estender a classe *ContentProvider*
 - Declarar o content provider no arquivo *AndroidManifest.xml*

A Classe *ContentProvider*



- É necessário implementar alguns métodos

```
public class MyContentProvider extends ContentProvider {  
    public boolean onCreate() { }  
  
    public Cursor query(Uri uri, String[] projection,  
        String selection, String[] selectionArgs, String sortOrder) { }  
  
    public Uri insert(Uri uri, ContentValues values) { }  
  
    public int update(Uri uri, ContentValues values,  
        String selection, String[] selectionArgs) { }  
  
    public int delete(Uri uri, String selection,  
        String[] selectionArgs) { }  
  
    public String getType(Uri uri) { }  
}
```

Declarando o Content Provider




- O content provider criado deve ser declarado no *AndroidManifest.xml*

```
<application>  
    ...  
    <provider  
        android:name="com.mypackage.provider.MyProvider">  
        android:authorities="com.mypackage.provider.MyApplication">  
    </provider>  
    ...  
</application>
```


Classe do content provider

URI authority


Content Providers Nativos


- O Android possui um conjunto de content providers nativos para serem utilizados

Content Provider	Descrição
Browser	Bookmarks, histórico de navegação e buscas
CallLog	Ligações feitas e recebidas
ContactsContract	Contatos
MediaStore	Conteúdo multimídia do dispositivo (áudio, vídeo e imagens)
Settings	Preferências do usuário
UserDictionary	Dicionário do usuário (usado para prever digitação de textos)
CalendarContract	Agenda

Loaders


- Disponibilizados a partir do Android 3.0 (API Level 11)
- Permitem carregar dados de forma assíncrona
- Gerenciam o ciclo de vida do cursor usado para ler os dados
- Monitoram a fonte dos dados e são capazes de atualizarem os dados automaticamente
- Podem ser usados em activities e fragments

Usando Loaders


- Loaders são comumente usados para ler dados de content providers
- O objeto **LoaderManager** associado à activity/fragment é usado para inicializar o loader

Parâmetros

```

LoaderManager lm = getLoaderManager();
lm.initLoader(0, null, this);

```

ID do loader

LoaderManager.LoaderCallbacks

A Interface *LoaderCallbacks*



- Possui métodos que são chamados durante o ciclo de vida de um loader
- Para buscar dados em content providers, um **CursorLoader<Cursor>** é normalmente utilizado

```
public class MainActivity extends Activity implements
    LoaderManager.LoaderCallbacks<Cursor> {

    public Loader<Cursor> onCreateLoader(int id, Bundle args) { }
    public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) { }
    public void onLoaderReset(Loader<Cursor> loader) { }
}
```

Criando um *CursorLoader<Cursor>*



- A criação é feita em *onCreateLoader()*

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    return new CursorLoader(this, uri, projection, selection,
        selectionArgs, sortOrder);
}
```

Os parâmetros são basicamente os mesmos passados para o método *query()* do content provider

Adapters e o Uso de Cursores



- Algumas views fazem o uso de adapters para poderem exibir dados
 - Ex: *ListView*, *GridView*
- A classe **SimpleCursorAdapter** representa um adapter que extrai dados de um cursor
 - Utilizado para exibir dados vindos de content providers e bancos de dados

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
    android.R.layout.simple_list_item_1,
    null, projection, new int[]{ android.R.id.text1 }, 0);
```

Um *CursorLoader* pode usar um *SimpleCursorAdapter* para exibir dados em uma view