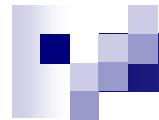




NTI / PAA PROGRAMOVÁNÍ MOBILNÍCH APLIKACÍ

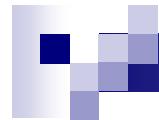
8. ContentProvider & ContentResolver

Ing. Igor Kopetschke – TUL, NTI
<http://www.nti.tul.cz>



Android – Content Provider

- Prostředek pro poskytování dat
- Data uložena v SQLite DB, souborech, na webu apod.
 - Závisí na konkrétní implementaci
- Jediný možný způsob, jak sdílet data napříč aplikacemi
 - Výjimkou je:
 - Mode - při přístupu k některým souborům
 - IPC - Android Interface Definition Language (AIDL)
- Možné využívat i pro přístup k datům z vlastní aplikace
 - Výhodou je zkrácení kódu



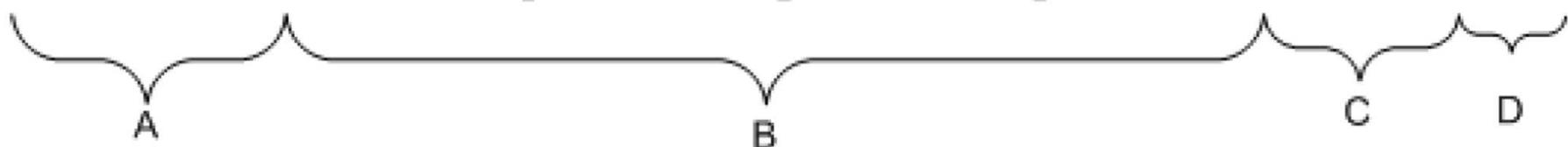
Android – Content Provider

- Každá aplikace může poskytovat svoje data přes Content Provider definované pomocí URI
- Pro přístup k datům poskytovaným ContentProviderem slouží ContentResolver
- Příklady (systémových) content providerů
 - Kontakty
 - SMS
 - Zmeškané hovory
 - Multimedia library

Android – Content Provider - URI

- Musí se definovat unikátní URI

`content://com.example.transportationprovider/trains/122`



- A. Scheme - standardní předpona indikující, že data mají být zpracována Content Providerem
- B. Authority - identifikátor Provideru
 - Musí být deklarována v AndroidManifestu
 - Měla by obsahovat jméno balíčku, aby se zajistila jedinečnost
- C. Data type path - vymezení obsahu
 - Může mít několik úrovní (`land/bus`, `land/train`, `sea/ship...`)
- D. Instance identifier - ID položky, kterou požadujeme
 - Pokud chceme více záznamů, vynecháme

Povinná je pouze část A a B

Android – Content Provider - URI

- Definice provideru v AndroidManifestu

```
1. <provider  
2.     android:name=".MyProvider"  
3.     android:authorities="cz.cvut.exampleprovider" />
```

Privátní CP - android:exported="false"

- Třída MyProvider.java

```
1. public class MyProvider extends ContentProvider{  
2.     public static final Uri CONTENT_URI =  
3.         Uri.parse("content://cz.cvut.exampleprovider");  
4.     ...  
5. }
```



Android – Content Provider

- Je potřeba vždy implementovat 6 základních metod
 - `onCreate` – inicializace provideru, nastavení přístupu např. k DB
 - `getType` – vrácení MIME dat u konkrétní URI
 - MIME formát ve tvaru `vnd.X.cursor.dir/Y` nebo `vnd.X.cursor.item/Y`
 - X – název společnosti, projektu apod., Y – název typu oddělený tečkami
 - `query` – výběr dat
 - `insert` – přidání dat
 - `update` – aktualizace dat
 - `delete` – smazání dat

Android – Content Provider – UriMatcher

- Pokud Provider poskytuje více typů dat, je potřeba pro každý typ vytvořit proměnnou a nastavit UriMatcher

```
content://cz.cvut.exampleprovider/auta  
content://cz.cvut.exampleprovider/auta/3  
content://cz.cvut.exampleprovider/lode  
content://cz.cvut.exampleprovider/lode/7
```

```
1. private static final int AUTA_ALL_ROWS = 1;  
2. private static final int AUTA_SINGLE_ROW = 2;  
3. private static final int LODE_ALL_ROWS = 3;  
4. private static final int LODE_SINGLE_ROW = 4;  
5.  
6. private static final UriMatcher uriMatcher;  
7. static {  
8.     uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);  
9.     uriMatcher.addURI("cz.cvut.exampleprovider", "auta", AUTA_ALL_ROWS);  
10.    uriMatcher.addURI("cz.cvut.exampleprovider", "auta/#", AUTA_SINGLE_ROW);  
11.    uriMatcher.addURI("cz.cvut.exampleprovider", "lode", LODE_ALL_ROWS);  
12.    uriMatcher.addURI("cz.cvut.exampleprovider", "lode/#", LODE_SINGLE_ROW);  
13. }
```

Android – Content Provider - getType

- Vrací MIME požadavku v aktuální URI
- Formát:
 - 1 položka: vnd.<autor>.cursor.item/<typ obsahu>
 - Všechny položky: vnd.<autor>.cursor.dir/<typ obsahu>

```
1.  @Override
2.  public String getType(Uri uri) {
3.      switch (uriMatcher.match(uri)) {
4.          case AUTA_ALL_ROWS:
5.              return "vnd.cvut.cursor.dir/auta";
6.          case AUTA_SINGLE_ROW:
7.              return "vnd.cvut.cursor.item/auta";
8.          case LODE_ALL_ROWS:
9.              return "vnd.cvut.cursor.dir/lode";
10.         case LODE_SINGLE_ROW:
11.             return "vnd.cvut.cursor.item/lode";
12.         default:
13.             throw new IllegalArgumentException("Unsupported URI: " + uri);
14.     }
15. }
```

Android – Content Provider - query

- Vrací Cursor s výsledky dotazu
- Analogicky se chovají ostatní metody (insert, update, delete)

```
1.  @Override
2.  public Cursor query(Uri uri, String[] projection, String selection,
3.  String[] selectionArgs, String sortOrder) {
4.  switch (uriMatcher.match(uri)) {
5.  case AURA_ALL_ROWS:
6.      return db.fetchAllCars();
7.  case AURA_SINGLE_ROW:
8.      String rowNumber = uri.getLastPathSegment();
9.      int i = Integer.parseInt(rowNumber);
10.     return db.fetchCar(i);
11. ...
12. }
```

Většinou se používá SQLiteQueryBuilder

Android – Content Provider - query

- Získání Cursoru za použití SQLiteQueryBuilderu

```
1.  @Override
2.  public Cursor query(Uri uri, String[] projection, String selection,
3.  String[] selectionArgs, String sortOrder) {
4.  SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
5.  switch (uriMatcher.match(uri)) {
6.  case AUTA_ALL_ROWS:
7.      queryBuilder.setTables(TABLE_CAR);
8.      break;
9.  case AUTA_SINGLE_ROW:
10.     String rowNum = uri.getLastPathSegment();
11.     queryBuilder.setTables(TABLE_CAR);
12.     queryBuilder.appendWhere(ID + "=" + rowNum);
13.     break;
14. ...//přístup do DB atd.
15. return queryBuilder.query(db, projection, selection, selectionArgs,
   orderBy, null, sortOrder);
16. }
```

Android – Content Resolver

- Slouží k přístupu do ContentProvideru
- Referenci získáme `context.getContentResolver();`
- Chová se podobně jako DB
 - Update, insert, delete a query jsou stejné s výjimkou prvního parametru
 - První parametr je URI požadovaného obsahu

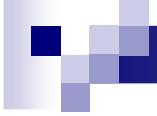
```
1. Cursor c = getContentResolver().query(Contact.CONTENT_URI, null, null, null, null);
2. startManagingCursor(c);
3. int colName = c.getColumnIndex(Contact.DISPLAY_NAME);
4. int colId = c.getColumnIndex(Contact._ID);
5. if(c.moveToFirst()){
6.     do{
7.         String jmeno = c.getString(colName);
8.         String id = c.getString(colId);
9.         ...
10.    }while(c.moveToNext());
11. }
12. stopManagingCursor(c);
```

Android – Content Resolver

- startManagingCursor & stopManagingCursor
 - Metody které provážou životní cyklus Cursoru s životním cyklem Activity
 - Pokud se activita zastaví, Cursor se deaktivuje
 - Když se activita opět obnoví, zavolá se `requery()` ;
 - Po skončení activity se Cursor uzavře
- Pro přístupu k systémovému ContentProvideru je většinou zapotřebí oprávnění

Oprávnění pro přístup ke kontaktům (`AndroidManifest.xml`)

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



Použité a doporučené zdroje

- <http://developer.android.com/>
- <http://www.zdrojak.cz/serialy/vyvijime-pro-android/>
- <http://www.itnetwork.cz/java/android>
- <https://users.fit.cvut.cz/cermaond/dokuwiki>
- Google...



.. A to je pro dnešek vše

DĚKUJI ZA POZORNOST