Android Fundamentals

Лекция 4

Saving Data

Storing data

- Saving arbitrary files in Android's **file system**
- Saving key-value pairs of simple data types in a **shared preferences** file
- Using **databases** managed by SQLite

Internal storage

- It's always available.
- Files saved here are accessible by only your app by default.
- When the user uninstalls your app, the system removes all your app's files from internal storage.

Internal storage

```
File appDirectory = getContext().getFilesDir()
```

```
File cacheDirectory = getContext().getCacheDir()
```

External storage

- It's not always available, because the user can remove it from the device.
- It's world-readable, so files saved here may be read outside of your control.
- When the user uninstalls your app, the system removes your app's files from here only if you save them in the directory from getExternalFilesDir().

Permissions

```
<manifest ...>
    <uses-permission android:name="android.permission.
READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

External storage

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

External storage

Environment.getExternalStoragePublicDirectory

(Environment.DIRECTORY_PICTURES)

getContext().getExternalFilesDir(Environment.

DIRECTORY_PICTURES)

SharedPreferences

```
Context context = getActivity();
SharedPreferences sharedPref =
        context.getSharedPreferences(
                "com.example.myapp.PREFERENCE_FILE_NAME",
                Context.MODE_PRIVATE);
```

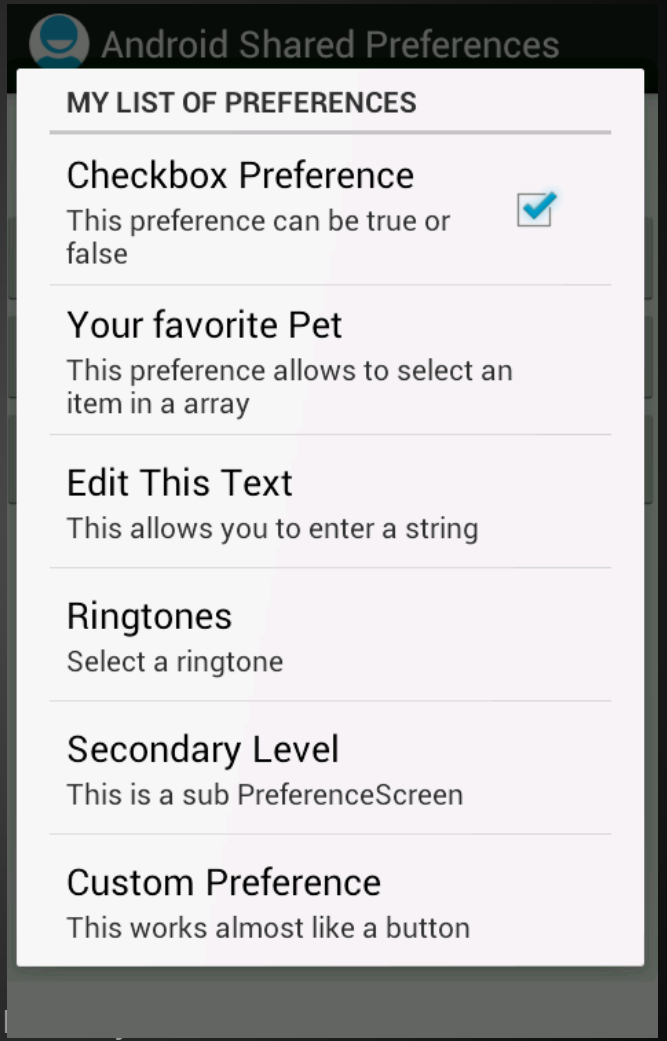
Saving key-value

```
int newHighScore = 42;

SharedPreferences.Editor editor = sharedPref.edit();

editor.putInt(«new_high_score_key", newHighScore);

editor.commit();
```

Retrieving key-value

```
int newHighScore = sharedPref.getInt
(«new_high_score_key", 0);
```

Preferences Activity



Preferences Activity

```
<PreferenceCategory
        android:title="@string/dialog_based_preferences">

    <EditTextPreference
            android:key="edittext_preference"
            android:title="@string/title_edittext_preference"
            android:summary="@string/summary_edittext_preference"
            android:dialogTitle="@string/dialog_title_edittext_preference" />

    <ListPreference
            android:key="list_preference"
            android:title="@string/title_list_preference"
            android:summary="@string/summary_list_preference"
            android:entries="@array/entries_list_preference"
            android:entryValues="@array/entryvalues_list_preference"
            android:dialogTitle="@string/dialog_title_list_preference" />

</PreferenceCategory>
```

Preferences Activity

~~setContantView(R.layout.main_activity);~~

addPreferencesFromResource(R.xml.pref);

Database in Android

SQLite. Table

```
public class CoreObjectTable {
    public static final String TABLE_CORE_OBJECT = "coreObject";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_URL = "url";

    private static final String DATABASE_CREATE = "create table "
            + TABLE_CORE_OBJECT + "("
            + COLUMN_ID + " integer primary key autoincrement, "
            + COLUMN_NAME+ " text not null, "
            + COLUMN_URL + " text not null);";

    public static void onCreate(SQLiteDatabase database) { database.execSQL(DATABASE_

    public static void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(SQLiteHelper.class.getName(),
                "Upgrading database from version " + oldVersion + " to "
                        + newVersion + ", which will destroy all old data"
        );
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CORE_OBJECT);
        onCreate(db);
    }
}
```

SQLite. Helper

```
public class SQLiteHelper extends SQLiteOpenHelper {


    private static final String DATABASE_NAME = "cursor_research.db";
    private static final int DATABASE_VERSION = 1;

    // Database creation sql statement


    public SQLiteHelper(Context context) { super(context, DATABASE_NAME, null, DA

    @Override
    public void onCreate(SQLiteDatabase db) { CoreObjectTable.onCreate(db); }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        CoreObjectTable.onUpgrade(db, oldVersion, newVersion);
    }
}
```

SQLite. Working with db

```
public void open() { database = dbHelper.getWritableDatabase(); }

//DO SOMETHING

public void close() { dbHelper.close(); }
```

SQLite. Create/Query

```
public CoreObject createCoreObject(String name, String url) {
    ContentValues values = new ContentValues();
    values.put(CoreObjectTable.COLUMN_NAME, name);
    values.put(CoreObjectTable.COLUMN_URL, url);
    long insertId = database.insert(CoreObjectTable.TABLE_CORE_OBJECT, null,
            values);
    Cursor cursor = database.query(CoreObjectTable.TABLE_CORE_OBJECT,
            allColumns, CoreObjectTable.COLUMN_ID + " = " + insertId, null,
            null, null, null);
    cursor.moveToFirst();
    CoreObject newCoreObject = cursorToCoreObject(cursor);
    cursor.close();
    return newCoreObject;
}
```

SQLite. Query

```
public Cursor query (String table,
String[] columns,
String selection, String[] selectionArgs,
String groupBy,
String having,
String orderBy,
String limit)
```

SQLite. Cursor foreach

```
cursor.moveToFirst();
while (!cursor.isAfterLast()) {
    CoreObject comment = cursorToCoreObject(cursor);
    comments.add(comment);
    cursor.moveToNext();
}
// make sure to close the cursor
cursor.close();
```

CursorAdapter

```
@Override
public void bindView(View view, Context context, Cursor cursor)


@Override
public View newView(Context context, Cursor cursor, ViewGroup parent)
```

ContentProvider

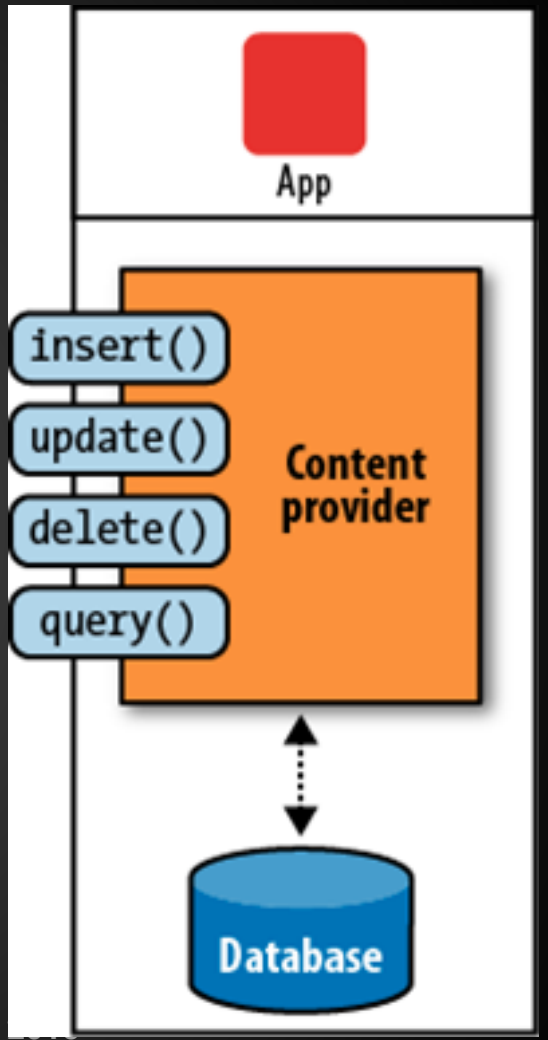
ContentProvider

You don't need to develop your own provider if you don't intend to share your data with other applications. However, you do need your own provider to provide custom search suggestions in your own application. You also need your own provider if you want to copy and paste complex data or files from your application to other applications.

Manifest

```
<manifest ...>
    <provider
        android:name=".db.MyContentProvider"
        android:authorities="@string/content_provider_authorities"
        android:exported="false" />
    ...
</manifest>
```

ContentProvider methods



Query cursor

```
public final Cursor query(
        Uri uri,
        String[] projection,
        String selection,
        String[] selectionArgs,
        String sortOrder)
```

URI

content://<authority>/<path>/<id>

content://com.example.app.provider/table1

content://com.example.app.provider/table2/dataset1

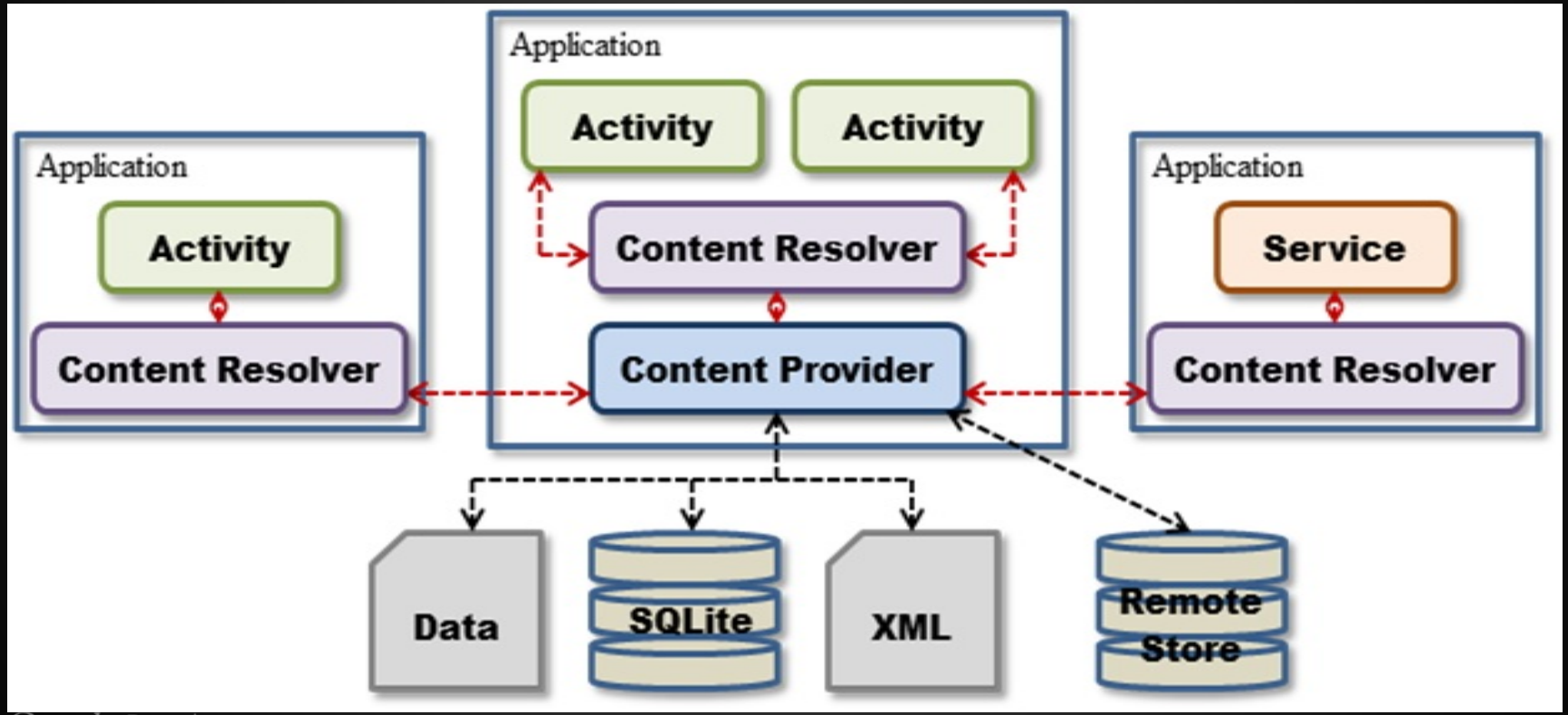
content://com.example.app.provider/table3

content://com.example.app.provider/table3/6

content://com.example.app.provider/table2/#

content://com.example.app.provider/*

Content Provider Architecture



Insert value

```
ContentValues mNewValues = new ContentValues();
mNewValues.put(APP_ID_ROW, "example.user");
mNewValues.put(LOCALE_ROW, "en_US");
mNewValues.put(WORD_ROW, "insert");
mNewValues.put(FREQUENCY_ROW, "100");

mNewUri = getContentResolver().insert(CONTENT_URI,
mNewValues);
```

Loaders

Loaders

- Available to every Activity and Fragment.
- Provide asynchronous loading of data.
- Monitor the source of their data and deliver new results when the content changes.
- Automatically reconnect to the last loader's cursor when being recreated after a configuration change. Thus, they don't need to re-query their data.

InitLoader

```
Loader getLoaderManager().initLoader(
        int loaderId, Bundle args,
                LoaderManager.LoaderCallbacks callback);
```

LoaderCallbacks

```
public interface LoaderCallbacks<D> {
    public Loader<D> onCreateLoader(int id, Bundle args);

    public void onLoadFinished(Loader<D> loader, D data);

    public void onLoaderReset(Loader<D> loader);
}
```

CursorLoader

```
public CursorLoader(
        Context context, Uri uri,
        String[] projection,
        String selection,
        String[] selectionArgs,
        String sortOrder)
```

Observer

```
getContext().getContentResolver()
                    .notifyChange(URI, null);
```

Home work

Сохранить погоду в БД
Adapter заменить на CursorAdapter
Подгрузить курсор с погодой в Adapter при помощи Loader

Time for Q&A

Lesson 4