Введение в разработку под Android

Лекция 5

Макаров И. О.

16 февраля 2022

Объект Application

https://developer.android.com/reference/android/app/Application.html

Базовый класс для работы с состоянием всего приложения. Данный класс инстанцируется первым при запуске вашего приложения.

Дает вам дополнительную функциональность. Например сигналы: **onTerminate()**, **onTrimMemory(int level)** и другие.

Можно реализовать свой собственный класс. Для этого нужно:

- реализовать свою собственную имплементацию
- в файле манифеста в разделе “<application>” указать атрибут “android:name” полное имя вашего класса

Замечания:

- в большинстве ситуаций не требуется реализовывать свой собственный application
- достаточно только создать singleton-объект, в который передать контекст приложения **Context.getApplicationContext()**
- такой подход позволит достичь схожего поведения и будет более модульным

Работа с app-specific ресурсами

https://developer.android.com/training/data-storage/app-specific

Во многих сценариях ваше приложение создает объекты на файловой системе.

Другие приложения либо не должны иметь доступ к таким объектам, либо просто не нуждаются в нем.

Система предоставляет 2 основных хранилища:

- **internal**: Место для хранения persistent файлов и кеша. Система предотвращает доступ других приложений к хранимым данным. Начиная с android 10 (API level 29) это хранилище шифруется.
- **external**: Отличается тем, что другие приложения (при наличии необходимых разрешений) могут получить доступ к данным. Подразумевается, что хранимыми данными будет пользоваться только ваше приложение. Для хранения данных, доступ к которым должны иметь другие приложения, лучше использовать **shared storage**.

Замечание: все данные удаляются при удалении приложения пользователем. Для хранения данных, время жизни которых не связано с временем жизни вашего приложения (например, фотографии пользователя), необходимо использовать альтернативные хранилища (в случае фотографий **media collection**).

Короткий обзор shared storage

https://developer.android.com/training/data-storage/shared

Используется для хранения объектов, которые не связаны с временем жизни вашего приложения (фото, видео, документов и т.п.).

Система API для хранения следущего контента:

- **media content**: Хранятся в стандартные public-директориях, таким образом пользователь получает некоторое “общее место” для хранения медиа-контента. Доступ осуществляется по средствам **MediaStore**.

- **documents and other files**: Специальные директории для хранения “документов” (например, PDF, EPUB файлов). Доступны через **Storage Access Framework**.

- **datasets**: Начиная с Android 11 (API level 30). Для хранения больших объемов данных в виде бинарных объектов (**BlobStoreManager**). Это может быть полезно, например, для задач машинного обучения.

SharedPreferences

https://developer.android.com/training/data-storage/shared-preferences

Используется для хранения небольшого количества объектов вида **ключ-значение**.

Каждый объект:

- ссылается на некоторый файловый дескриптор (хранящий фактические значения)
- может быть **приватным** (private) и **публичным** (public)
- и управляется фреймворком.

Доступ к объектам (или создание новых) осуществляется по средствам методов:

- **getSharedPreferences** - доступен в контексте приложения (в любом месте, где вам доступен контекст приложения)
- **getPreferences** - доступен в контексте конкретного activity

SharedPreferences

https://developer.android.com/training/data-storage/shared-preferences

Для получения значений:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.integer.saved_high_score_default_key);
int highScore = sharedPref.getInt(getString(R.string.saved_high_score_key), defaultValue);
```

Для записи значений:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score_key), newHighScore);
editor.apply();
```

Замечание: для синхронной записи используйте commit.

SQLite

https://developer.android.com/training/data-storage/sqlite
https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase
https://developer.android.com/training/data-storage/room

Для работы со структурированными данными удобно использовать различные БД.

Android позволяет работать с SQLite “из коробки”. Для этого необходимо подключить соответсвующий пакет **android.database.sqlite**.

Основные подходы для работы с СУБД:

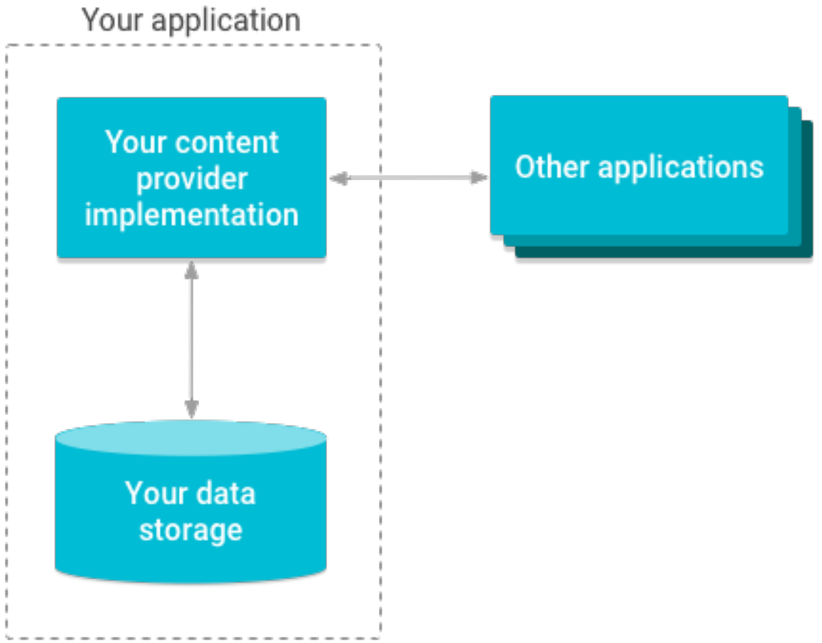
- Низкоуровневое API (классы SQlite, SQLiteDataBase).
- Расширения (Room Library). Рекомендованный способ работы с БД.

ContentProvider

https://developer.android.com/guide/topics/providers/content-provider-basics

Способ взаимодействия и управления:

- собственным контентом
- контентом, сохраненным другими приложениями
- способ описания протокола для получения контента нашего приложения другими приложениями



Замечание: как правило необходим для межпроцессного взаимодействия.

Пример кеширования изображений

https://developer.android.com/topic/performance/graphics/cache-bitmap

Варианты кеширования:

- в памяти
- на файловой системе
- комбинации первых двух
- Soft & Weak references https://habr.com/ru/post/169883/

Замечание: рекомендуется использовать для задач кеширования изображений библиотеку **Glide**