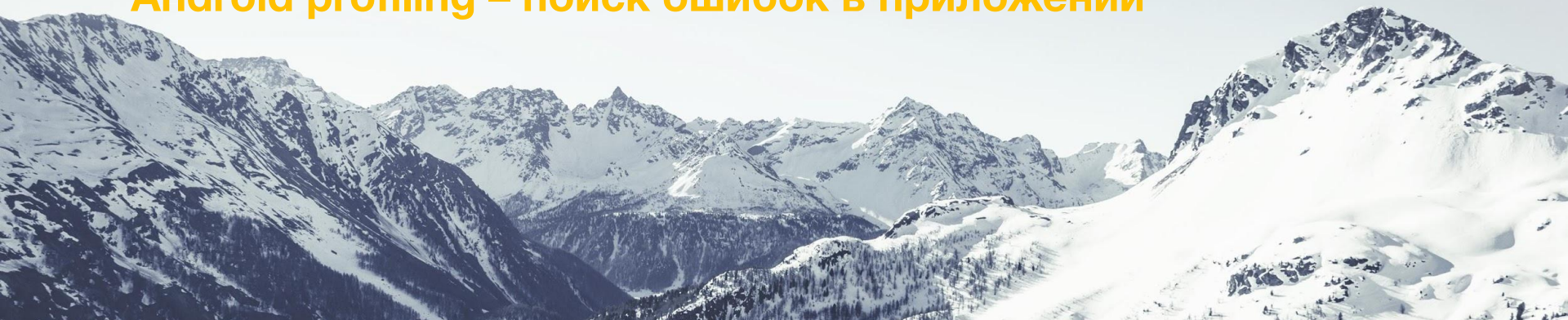
<TeachMeSkills/>

курс

Android разработчик

Android profiling – поиск ошибок в приложении



Агенда занятия:

Profiler
Утечки памяти
ANRs
Performance

- **Profiler**
  - Утечки памяти
  - ANRs
  - Performance

Profiler

Android Profiler — инструмент Android Studio для анализа:

- CPU (нагрузка, вызовы функций)
- Memory (утечки, аллокации)
- Network (запросы, payload, скорость)
- Energy (энергопотребление)

Как использовать:

- В Android Studio: View > Tool Windows > Profiler
- Запускаем приложение на реальном устройстве/emulator
- Анализируем графики, аллокации, вызовы

Memory Profiler

Что отслеживает:

- Количество занятой памяти (Java/Kotlin heap, native, graphics, etc);
- Аллокации объектов;
- Garbage Collector активности;
- Утечки (через heap dump).

Memory Profiler

Возможность	Описание
🔍 Live memory usage graph	Видишь, как растет/падает использование памяти во времени
🧬 Record memory allocations	Смотришь, какие классы создаются, как часто и сколько весят
💣 Dump Java heap	Снимок памяти → позволяет искать утечки, висячие Activity
👓 Class list view	Видишь количество и размер экземпляров каждого класса
🔁 GC события	Отмечаются точки, где сработал сборщик мусора

Memory Profiler

Пример использования:

- Нажми Record → запусти действия в приложении → останови запись;
- Нажми Dump Java Heap;
- Ищи объекты, которые не должны оставаться (например, MainActivity);
- Открой дерево ссылок, чтобы найти, кто их удерживает.

CPU Profiler

Что отслеживает:

- Загрузка CPU (время, когда главный и рабочие потоки были активны)
- Стек вызовов функций (Call stack)
- Частота вызова методов
- Задержки и блокировки

CPU Profiler

Возможность	Описание
🧬 Record method tracing	Запись всех вызовов методов (с указанием времени)
🔍 Call chart / Flame chart	Визуализация нагрузки и времени вызова по стеку
🧵 Thread activity timeline	Видно, когда и какие потоки работали, были заблокированы или спали
⛓️ Trace CPU scheduling	Распределение задач между потоками (для ANR/Performance анализа)
Возможность	Описание

CPU Profiler

Пример использования:

При открытии сложного экрана приложение фризит.

- Включи запись CPU
- Открой экран → останови запись
- Найди метод, который выполнялся дольше всего
- Оптимизируй алгоритм, используй Dispatchers.IO, AsyncListDiffer и т.п.

Network Profiler

Что отслеживает:

- Все HTTP-запросы (Retrofit, OkHttp, Volley, HttpURLConnection и пр.);
- Размеры запросов/ответов;
- Заголовки, параметры, payload;
- Время подключения, ожидания, загрузки.

Network Profiler

Возможность	Описание
🌍 Request timeline	Все запросы с точками старта и длительностью
🧾 Details per request	Заголовки, код ответа, тело запроса/ответа
📦 Payload viewer	JSON/XML/HTML просмотр, форматированный
⚠️ Error tracking	Видны ошибки сети (timeout, 404, 500)
Возможность	Описание

Network Profiler

Пример использования:

Приложение иногда долго загружает данные.

- Открой Network Profiler;
- Посмотри, какие запросы занимают >1 сек;
- Проверь, где именно задержка (DNS, Connect, Server wait);
- Добавь кэш, сжатие, или загрузи в фоне.

Energy Profiler (в Android Studio Arctic Fox и выше)

Что отслеживает:

- Энергопотребление: CPU, Radio (сеть), GPS, Sensor;
- WakeLocks;
- Фоновые задачи, JobScheduler, AlarmManager.

Energy Profiler (в Android Studio Arctic Fox и выше)

Возможность	Описание
🔌 Energy timeline	Отображает пики использования энергии
⏰ WakeLock visualization	Видно, когда и как долго работал WakeLock
📡 JobScheduler timeline	Показ запланированных фоновых задач
📍 Location requests	Запросы на GPS и геолокацию
Возможность	Описание

Profiler

Смотрим на практике

Profiler

▶ **Утечки памяти**

ANRs

Performance

Утечки памяти

Утечка памяти (memory leak) — это ситуация, при которой объекты, больше не нужные приложению, не освобождаются сборщиком мусора, потому что на них всё ещё есть ссылки.

В результате:

- Память расходуется впустую;
- Устройство может замедляться;
- В тяжёлых случаях происходит OutOfMemoryError.

Основные причины утечек памяти в Android

- Неправильные ссылки на Context
- Хранение Activity, View или Context в static полях, singleton-ах, Handler, ViewModel
- Утечки через View в Fragment:
- Observer без удаления
- LiveData, BroadcastReceiver, EventBus, RxJava подписки без отписки ведут к утечкам.
- Кеширование больших объектов без лимитов

Смотрим на практике

Как найти утечки памяти

1. Memory Profiler (Android Studio);
2. LeakCanary (автоматический детектор утечек)

LeakCanary — библиотека от Square, интегрируется за 5 минут и показывает всплывающее уведомление при утечке.
Автоматически отслеживает утечки Activity, Fragment, ViewBinding, Context.

Profiler

Утечки памяти

▶ **ANRs**

Performance

ANRs (Application Not Responding)

ANR происходит, если Главный поток (UI thread) блокируется более чем на 5 сек.

Частые причины:

- Сетевые вызовы в UI-потоке;
- Долгие вычисления;
- Синхронные операции с БД;
- Некорректные lock-и (deadlock/livelock).

.

Пример: ANR при загрузке

Проблема: После нажатия на кнопку "Загрузить данные" UI замирает на 5+ секунд → ANR.

Диагностика:

- Logcat → ANR in com.example.app + stack trace;
- Профайлер → spike в CPU и UI thread блок.

Пример: ANR при загрузке

Причина: синхронный Retrofit-вызов в UI thread.

Решение:

- Использовать асинхронные вызовы (enqueue, suspend);
- Перенести тяжелые задачи на Dispatchers.IO.

Profiler

Утечки памяти

ANRs

- **Performance**

Performance

Метрики производительности:

- FPS (Frames per Second);
- Time to first draw;
- Startup time;
- CPU/GPU load;
- Battery usage.

Инструменты:

- Layout Inspector: анализ иерархии вёрстки, поиска "тяжёлых" layout'ов.
- Systrace: глубокий анализ событий системы.
- StrictMode: выявление неэффективных операций.

Layout Inspector

Layout Inspector — это инструмент Android Studio, который позволяет в реальном времени исследовать иерархию вёрстки (View hierarchy) запущенного приложения на устройстве или эмуляторе.
С его помощью можно визуализировать структуру текущего экрана, свойства View, размеры, padding, margin, видимость, а также выявлять неэффективные или неправильные участки UI.

Layout Inspector

Для чего используется?

- Поиск слишком глубокой или сложной иерархии (Nested layouts).
- Анализ размеров и расположения элементов.
- Диагностика "невидимых" View, перекрытий, неправильных размеров.
- Проверка, как выглядят элементы на разных устройствах и состояниях.
- Быстрое выявление причин лагов и низкой производительности UI.

Layout Inspector

Как пользоваться?

- Запусти приложение на устройстве или эмуляторе через Android Studio.
- Открой вкладку Layout Inspector.
- View → Tool Windows → Layout Inspector.
- или через меню Tools → Layout Inspector.
- Выбери нужный процесс приложения.

Layout Inspector

Layout Inspector покажет:

- Дерево View всех видимых элементов на экране.
- 2D/3D визуализацию layout.
- Свойства выбранного View (размер, id, visibility, margin и др.).
- Скриншот экрана с выделением выбранного элемента.

Советы

1. Старайтесь держать глубину иерархии менее 10 (желательно 5–7).
2. Используйте ConstraintLayout вместо вложенных LinearLayout/RelativeLayout (при вложенности больше 3).
3. Удаляйте неиспользуемые View и Layout.
4. Проверяйте состояние layout на разных состояниях (пусто, полно, ошибка).

StrictMode

StrictMode — это инструмент Android, который позволяет выявлять потенциально опасные или неэффективные операции в приложении во время разработки (например, долгие операции на UI-потоке, утечки ресурсов, неправильное использование памяти).

Помогает обнаружить:

- Долгие операции в main thread (I/O, работа с БД, сеть).
- Утечки файловых дескрипторов, сокетов, ресурсов.
- Несвоевременное освобождение объектов.
- Доступ к диску или сети на главном потоке (причина ANR и лагов).
- Особенно полезен на этапе debug-разработки.

StrictMode

Основные возможности и типы проверок:
detectDiskReads() обнаруживает чтение с диска в UI-потоке.
detectDiskWrites() обнаруживает запись на диск в UI-потоке.
detectNetwork() обнаруживает сетевые запросы в UI-потоке.
detectLeakedClosableObjects() — утечки объектов, требующих закрытия (File, Cursor).
detectLeakedSqlLiteObjects() — утечки баз данных.
detectActivityLeaks() — утечки Activity.
penaltyLog() логирует проблему.
penaltyDialog() показывает диалог (только для ThreadPolicy).
penaltyDeath() аварийно завершает приложение при ошибке (только для тестирования).

Как включить StrictMode?

Обычно включают в Application или в onCreate() Activity только в debug-сборках:

```
if (BuildConfig.DEBUG) {
    StrictMode.setThreadPolicy(
        StrictMode.ThreadPolicy.Builder()
            .detectAll() // или .detectDiskReads(), .detectNetwork() и т.д.
            .penaltyLog() // выводит предупреждения в Logcat
            .penaltyDialog() // всплывающее окно (для Activity)
            .build()
    )
    StrictMode.setVmPolicy(
        StrictMode.VmPolicy.Builder()
            .detectAll() // проверяет утечки Activity, неправильные вызовы и т.д.
            .penaltyLog()
            .build()
    )
}
```

Практика

Задача 1.

Найти причину ANR и устранить ее.

Задача 2.

Найти причину утечки памяти и устранить ее.

Задача 3.

Изучить верстку в Layout Inspector и оптимизировать.

Домашнее задание

Задача 1. Поиск и устранение проблем

Найти все ошибки, утечки памяти и оптимизировать использование ресурсов.

Ссылка на гит.

Q&A

Ваши вопросы

Спасибо

<TeachMeSkills/>