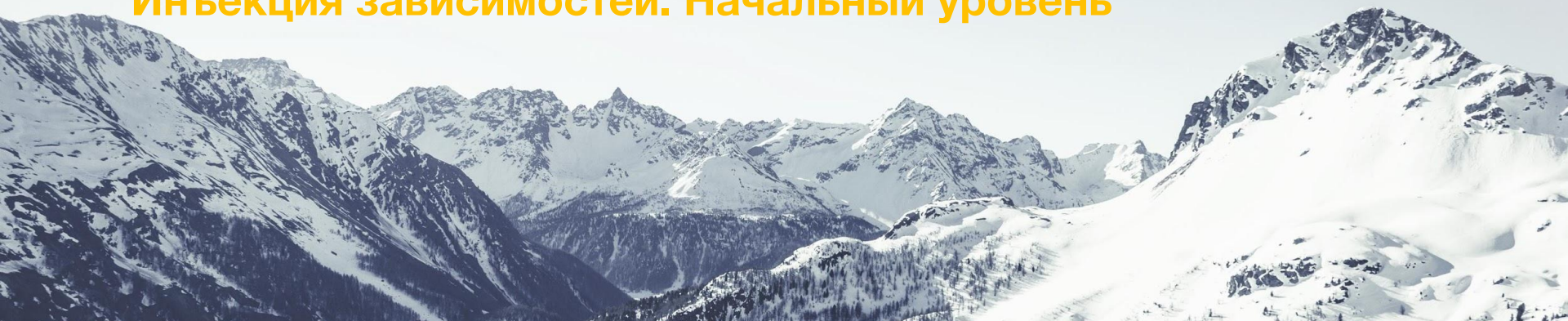
<TeachMeSkills/>

курс

Android разработчик

Инъекция зависимостей. Начальный уровень



Агенда занятия:

Что такое Dependency Injection (DI)

Dagger Hilt — современный стандарт от Google

Обзор других DI-инструментов

- **Что такое Dependency Injection**

  Dagger Hilt — современный стандарт от Google

  Обзор других DI-инструментов

Что такое Dependency Injection?

Инъекция зависимостей (Dependency injection, DI) — это принцип и техника в программировании, при которой объект (или модуль) получает все необходимые зависимости (другие объекты, сервисы, настройки) извне, а не создает их самостоятельно внутри себя.

```
class Car(val engine: Engine) // Engine дают "снаружи"


class CarNoDI {
    val engine = Engine() // Car сам решает, какой Engine использовать
}
```

Зачем это нужно?

- Гибкость: легко менять поведение, подставлять другие реализации.
- Тестируемость: можно подсовывать "фейковые" (mock) объекты при тестах.
- Упрощение кода: нет жесткой связанности между классами.
- Переиспользование: один и тот же класс может работать с разными зависимостями.

Реальный кейс

У вас есть Repository, который использует ApiService и Database. **Без DI** вы сами инициализируете зависимости в каждом месте, где они нужны. Это усложняет повторное использование кода, увеличивает связанность компонентов.

DI позволяет:

- Создать зависимости один раз;
- Прокинуть их туда, где они нужны;
- Управлять временем жизни объектов (scopes).

Как это работает?

Инъекция зависимостей работает так, что объект не создает сам свои зависимости, а получает их извне — обычно через конструктор, методы или поля. Это делается либо вручную, либо с помощью специальных DI-фреймворков.

Кто-то создаёт зависимости и "вкладывает" их. Это может быть главный класс (main), фабрика, специальный контейнер или DI-фреймворк.

DI-фреймворки

В больших проектах используют DI-фреймворки (например, Dagger/Hilt, Koin).
Они сами строят "граф зависимостей" и автоматически создают нужные объекты, подставляя их куда надо.

DI-фреймворки

Как DI-фреймворк решает задачу:

1. Вы описываете зависимости (например, аннотациями или модулями);
2. Фреймворк при запуске анализирует, какие объекты нужны, и создает их в нужном порядке;
3. В нужный момент он "вкладывает" готовые зависимости в конструктор/сеттер/поле вашего объекта.

Итог

- Класс просто объявляет, что ему нужен какой-то объект.
- Кто-то снаружи (main, фабрика, DI-фреймворк) создает эти объекты и передает их.
- Это позволяет легко менять реализации, делать тесты, не зависеть от деталей создания.

Что такое Dependency Injection

▶ **Dagger Hilt — современный стандарт от Google**

Обзор других DI-инструментов

Dagger

Dagger 2 — статически компилируемый фреймворк DI от Google.

- Основан на аннотациях.
- Генерирует быстрый, эффективный код во время компиляции.
- Позволяет строить мощные графы зависимостей.

Dagger

Минусы Dagger:

- Нужно много писать вручную (компоненты, модули).
- Сложная настройка для новичков.
- Неочевидность ошибок на старте.

Dagger Hilt

Hilt — это надстройка над Dagger, созданная специально для Android-разработки.

- Разработан Google.
- Делает DI проще, минимизирует шаблонный код.
- Глубоко интегрируется с жизненным циклом Android-компонентов (Activity, Fragment, ViewModel, Service).

Главные преимущества Hilt

- Автоматически создает компоненты для всех слоёв Android (Application, Activity, Fragment и др.).
- Простая интеграция с ViewModel и WorkManager.
- Автоматическое управление скоупами (областями жизни объектов).
- Меньше ручного кода и аннотаций, чем в чистом Dagger.
- Легкая интеграция с тестированием.

Базовые аннотации Hilt

@HiltAndroidApp ставится на класс Application, включает Hilt во всё приложение.
@AndroidEntryPoint ставится на Activity, Fragment, View, Service, куда нужно внедрять зависимости.
@Inject ставится на конструкторы или поля, чтобы получить зависимость.
@Module и @InstallIn – для предоставления зависимостей, которые нельзя создать через **@Inject** (например, retrofit-клиенты, репозитории).

Базовые аннотации Hilt

@Singleton и другие скоупы — для управления временем жизни объектов. **@HiltViewModel** автоматически интегрирует вашу ViewModel с графом зависимостей Hilt и позволяет использовать конструкторную инъекцию (@Inject). Аннотация сообщает Hilt, что этот класс ViewModel должен управляться Hilt и может получать зависимости через конструктор. Hilt сам создаёт нужную ViewModel и передает зависимости, объявленные в ее конструкторе.

Hilt на примере

Смотрим проект

Что такое Dependency Injection

Dagger Hilt — современный стандарт от Google

▶ **Обзор других DI-инструментов**

Обзор DI-инструментов

Характеристика	Hilt	Dagger 2	Koin	Kodein	Manual DI
Язык	Kotlin / Java	Kotlin / Java	Kotlin	Kotlin	Любой
Тип DI	Compile-time	Compile-time	Runtime	Runtime	-
Автогенерация кода	Да	Да	Нет	Нет	Нет
Аннотации	Да	Да	Нет (DSL)	Нет (DSL)	Нет
Интеграция с Android	Отличная	Хорошая	Хорошая	Хорошая	Зависит от вас
Легкость освоения	★★★☆☆	★★☆☆☆	★★★★☆	★★★★☆	★★★☆☆
Производительность	Высокая	Очень высокая	Средняя	Средняя	Высокая
Гибкость/Кастомизация	Средняя	Высокая	Средняя	Средняя	Абсолютная

Обзор DI-инструментов

Характеристика	Hilt	Dagger 2	Koin	Kodein	Manual DI
Граф зависимостей	Автоматически	Ручной / авто	Автоматически	Автоматически	Ручной
Поддержка ViewModel / Jetpack	Отличная	Хорошая	Отличная	Хорошая	Только вручную
Тестирование	Отлично	Отлично	Хорошо	Хорошо	Хорошо
Документация / Поддержка	Отличная	Отличная	Хорошая	Средняя	-
Поддержка мульти-модуля	Отличная	Отличная	Хорошая	Средняя	Зависит от вас

Практика

Задача 1. Подключить Hilt к проекту

Создать App, ViewModel, Module, Repository и предоставить зависимость в MainActivity.

Создайте GreetingRepository, который возвращает текст приветствия. Используйте Hilt, чтобы внедрить репозиторий в MainViewModel и отобразить приветствие.

Задача 2.

Добавить вторую Activity и ViewModel и предоставить им Repository.

Домашнее задание

Задача 1.

Настройте DI для предыдущего задания из темы “Network. Часть 2”, мини-приложение "FakeStore".

Функции:

- Каталог товаров (GET)
- Экран детали товара (GET /{id})
- Добавление в корзину (POST)
- Удаление из корзины (DELETE)
- Редактирование товара (PUT)

Q&A

Ваши вопросы

Спасибо

<TeachMeSkills/>