ONE PAGE SUMMARY

IT & Software › Android Development › Dependency Injection

Introduction to Dependency Injection

Master Dependency Injection for Cleaner Android Code!

DEFINITION

Dependency Injection (DI) is a design pattern that allows a program to create and manage its dependencies. It promotes loose coupling and easier testing by providing components with their required dependencies from an external source.

KEY CONCEPTS

Inversion of Control

Inversion of Control (IoC) shifts the responsibility of creating dependencies from the class itself to an external entity.

Service Locator

A Service Locator is a design pattern that provides a way to retrieve dependencies without needing to know how they are created.

Constructor Injection

Constructor Injection involves passing dependencies through a class constructor, ensuring that all required dependencies are provided at instantiation.

Field Injection

Field Injection allows dependencies to be set directly on class fields, often using annotations, but can lead to less clear code.

Scope Management

Scope Management defines the lifecycle of dependencies, ensuring they are created and destroyed at the appropriate times.

EXAMPLES

→ Using Dagger 2 for dependency management in an Android app.

→ Injecting Retrofit service into an Activity using constructor injection.

→ Setting up a ViewModel with dependencies in a Fragment.

MEMORY TIPS

★ Think of DI as 'delegating responsibility' for creating dependencies.

★ Remember IoC: 'Inversion of Control' means you control the flow of dependencies.

★ Use the acronym 'SICF' for Service Locator, Inversion of Control, Constructor Injection, Field Injection.

COMMON MISTAKES

✗ Overusing Field Injection, leading to tightly coupled code.

✗ Neglecting to manage the lifecycle of dependencies properly.

✗ Confusing DI with Service Locator, which can lead to design flaws.

QUICK RECAP

Dependency Injection simplifies the management of dependencies in Android development. By using patterns like Constructor Injection and Service Locator, you can create cleaner, more testable code.