

Implementing Dependency Injection with Spring

Spring Feature Description

Dependency Injection (DI) is a design pattern where an object receives its dependencies from an external source rather than creating them itself. In Spring, DI is facilitated by the Inversion of Control (IoC) container, which manages the lifecycle and configuration of application components (beans). This promotes loose coupling and easier testing.

Why Do We Need It in Flight Application?

In the current Flight Application, object dependencies are manually managed, leading to tightly coupled code and difficulties in testing and maintenance. By using Spring's DI, we can decouple components, making the application more modular, testable, and easier to manage.

Refactoring Application

We will refactor the application to leverage Spring's IoC container for managing dependencies. This involves:

- Defining beans for our components.
- Configuring Spring to manage these beans.
- Injecting dependencies using annotations.

Update The Code

- **Annotate DataService and CSVDataLoader Classes with @Service and @Component**

Use `@Service` to mark your service classes and `@Component` for other components like data loaders. Inject dependencies using constructor injection and optionally `@Autowired` (which is not required for constructor injection in Spring 4.3+).

- **Update Main Application to Use Spring Context**

Modify the `FlightsApplication` class to use Spring's `ApplicationContext` and retrieve beans using the `SpringApplication.run()` method.

Ensure to use `@ComponentScan` to scan your component classes.

Benefits of Using Dependency Injection

- **Loose Coupling:** Components are less dependent on each other.
- **Easier Testing:** Dependencies can be easily mocked or stubbed.

- **Better Maintainability:** Changes in one component have minimal impact on others.
- **Configuration Flexibility:** Easily switch between implementations.

Further Advices

Consider using constructor injection for mandatory dependencies and setter injection for optional ones. Avoid field injection to keep code testable and maintainable.