Spring Framework 6

Beginner to Guru

Basics of Dependency Injection

Dependency Injection for 5 year olds

“When you go and get things out of the refrigerator for yourself, you can cause problems. You might leave the door open, you might get something Mommy or Daddy doesn't want you to have. You might even be looking for something we don't even have or which has expired.

What you should be doing is stating a need, "I need something to drink with lunch," and then we will make sure you have something when you sit down to eat.”

John Munsch, 28 October 2009

Dependency Injection

- Dependency Injection is where a needed dependency is injected by another object.
- Can be at instantiation via constructor, or after via setter
- The class being injected has no responsibility in instantiating the object being injected.
- Some say you avoid declaring objects using ‘new’
  - Not 100% correct…
  - Be pragmatic in what is and is not being managed in the Spring Context

Types of Dependency Injection

- By class properties - least preferred
  - Can be public or private properties
  - Using private properties is EVIL
    - Spring can use reflection to set private properties
    - “Works” but is slow & makes testing difficult
- By Setters - Area of much debate
- By Constructor - Most Preferred

Concrete Classes vs Interfaces

- DI can be done with Concrete Classes or with Interfaces
- Generally DI with Concrete Classes should be avoided
- DI via Interfaces is highly preferred
  - Allows runtime to decide implementation to inject
  - Follows Interface Segregation Principle of SOLID
  - Also, makes your code more testable - mocking becomes trivial

Inversion of Control

- Inversion of Control - aka IoC
- Is a technique to allow dependencies to be injected at runtime
- Dependencies are not predetermined
- Allows the framework to compose the application by controlling which implementation is injected
  - Example - H2 in memory data source or MySQL data source

Inversion of Control

“One important characteristic of a framework is that the methods defined by the user to tailor the framework will often be called from within the framework itself, rather than from the user's application code. The framework often plays the role of the main program in coordinating and sequencing application activity. This inversion of control gives frameworks the power to serve as extensible skeletons. The methods supplied by the user tailor the generic algorithms defined in the framework for a particular application.”

Ralph Johnson and Brian Foote

IoC vs Dependency Injection

- IoC and DI are easily confused
- DI refers much to the composition of your classes
  - ie - you compose your classes with DI in mind
  - You might write code to ‘inject’ a dependency
- IoC is the runtime environment of your code
  - Control of Dependency Injection is inverted to the framework
  - Spring is in control of the injection of dependencies

Best Practices with Dependency Injection

- Favor using Constructor Injection over Setter Injection
- Use final properties for injected components
  - Declare property **private final** and initialize in the constructor
- Whenever practical, code to an interface

SPRING FRAMEWORK

GURU