devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

Spring framework 5 : Conceptos básicos sobre DI (Dependency Injection)

HACE 7 DÍAS DEJA UN COMENTARIO

1 Vote

En el post anterior se explicó como crear un proyecto de Spring framework 5 desde cero https://devs4j.com/2019/01/28/spring-framework-5-creando-un-proyecto-de-spring (https://devs4j.com/2019/01/28/spring-framework-5-creando-un-proyecto-de-spring), ahora toca el turno de hablar sobre inyección de dependencias en Spring framework 5.

Introducción

La definición más simple de Dependency Injection es cuando una un objeto necesita una dependencia y esta es inyectada por otro objeto, la clase que necesita la dependencia no es responsable sobre la creación del objeto.

Veamos el siguiente ejemplo:

Las clases **UserDao** y **RoleDao** necesitan un objeto de tipo **Datasource**, si no se utilizara inyección de dependencias, ambas clases deberían crear su propia instancia del objeto Datasource y si en el futuro es necesario actualizar la contraseña de la base de datos o incluso el motor de base de datos ese cambio se debería hacer en ambos Daos, lo cual reduce la mantenibilidad de la aplicación.

Haciendo uso de Dependency Injection un componente puede crear el objeto de tipo Datasource solo una vez e inyectarlo en todos los componentes que lo necesiten, esto nos ayuda a que los objetos de tipo **UserDao** y **RoleDao** no tengan la necesidad de crear los objetos a la base de datos, crear conexiones o cerrarlas.

Tipos de dependency injection

Los siguientes son los tipos de inyección de dependencias en Spring:

- Por constructor : Las dependencias de la clase se inyectarán a través del constructor
- Por métodos setters : Las dependencias de la clase se inyectarán a través de métodos setter
- Por atributos de la clase : Las dependencias de la clase se inyectarán haciendo uso del api de reflection, esto debido a que un atributo de una clase puede ser privado y esto hace que no se pueda acceder desde afuera.

Se puede utilizar inyección de dependencias haciendo uso de interfaces o de clases concretas, es preferible hacerlo a través de interfaces porque esto permite:

- Decidir la implementación que se inyectará en tiempo de ejecución
- Seguir los principios de diseño SOLID
- Haces el código más fácil de probar

En el ejemplo anterior hablamos sobre las clases **UserDao** y **RoleDao** utilizando una referencia de tipo **Datasource**, la cual es una interfaz, esto me permite que no importe el tipo de datasource que yo utilice la clase **UserDao** y **RoleDao** funcionarán correctamente.

Inversion of control

Inversion of control es una técnica que permite inyectar dependencias en tiempo de ejecución, esto significa que las dependencias no serán inyectadas de forma predeterminada todo el tiempo sino que pueden variar dependiendo del contexto de ejecución.

Algunas veces las personas confunden Inversion of control con Dependency Injection, la diferencia es que dependency injection se refiere a la composición y estructura de las clases e Inversion of control se refiere más al ambiente de ejecución del código.

En este post hablamos sobre la teoría de dependency injection, en el siguiente veremos como implementarlo utilizando Spring framework 5. Para estar al pendiente sobre nuestro contenido nuevo síguenos en nuestras redes sociales https://www.facebook.com/devs4j/ (https://www.facebook.com/devs4j/) y https://twitter.com/devs4j (https://twitter.com/devs4j).

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance

(https://geeksjavamexico.wordpress.com/mentions/raidentrance/)

Contacto:raidentrance@gmail.com