



devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

[HOME](#)

[ABOUT](#)

[CONTACT](#)



Anuncios

AUTOMATTIC

We're hiring backend developers.
Join us!

APPLY



REPORT THIS AD

Spring framework 5 : Dependency Injection utilizando Spring



HACE 6 DÍAS



1 COMENTARIO

1 Vote

Una vez que entendimos como funciona dependency injection e inversion of control el siguiente paso es aprender a utilizarlo con Spring framework.


Crear un Service de ejemplo

El primer paso será crear un service de ejemplo, a continuación se muestra la interfaz y su implementación:

MathService.java

```
/**
 *
 * @author raidentrance
 *
 */
```

```
*/  
  
public interface MathService {  
  
    /**  
     * Receives a set of numbers and return the  
     * @param values  
     * @return  
     */  
    double sum(double... values);  
  
}
```



MathServiceImpl.java

```
import org.springframework.stereotype.Service;  
  
/**  
 * @author raidentrance  
 *  
 */  
@Service  
public class MathServiceImpl implements MathService  
  
    @Override  
    public double sum(double... values) {  
        double sum = 0.0;  
        for (double value : values) {  
            sum += value;  
        }  
        return sum;  
    }  
}
```

Como vemos la clase `MathServiceImpl` esta marcada con la anotación `@Service`, esta nos permitirá indicarle a spring que debe crear una instancia de esta y mantenerla dentro de su contexto.

Accediendo al objeto creado

Una vez que creamos nuestras clases `MathService` y `MathServiceImpl` el siguiente paso será utilizarlas en nuestro código, para esto haremos lo siguiente dentro de nuestra clase de aplicación:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.Sprin
import org.springframework.context.ConfigurableAppl

import com.devs4j.service.MathService;
import com.devs4j.service.MathServiceImpl;

@SpringBootApplication
public class Devs4jSpringCoreApplication {

    public static void main(String[] args) {
        ConfigurableApplicationContext appl
            args);

        /**
         * Get bean by name
         */
        MathService bean = (MathService) ap
        System.out.println(bean.sum(1.1, 2.

        /**
         * Get bean by type
         */
        MathService bean2 = applicationCont
```

```
System.out.println(bean2.sum(1, 2.3));

/**
 * Get bean by name and type
 */
MathService bean3 = applicationCont
System.out.println(bean3.sum(1, 2.3));
}

}
```

Como podemos ver al iniciar la aplicación Spring framework devolverá un objeto de tipo **ConfigurableApplicationContext** el cual nos permitirá acceder a los beans que existen dentro del contexto de Spring, para hacerlo podremos alguno de los siguientes métodos:

- **getBean**(String name) : Devuelve el objeto que tenga el nombre que se pasa como parámetro, por default el nombre del bean será el mismo al de la clase pero iniciando con minúsculas, si se desea cambiar se puede poner el nuevo nombre dentro de la anotación `@Service("nuevoNombreDelBean")`.
- **getBean**(Class<T> requiredType) : Devuelve un objeto del tipo que se esta solicitando.
- **getBean**(String name, Class<T> requiredType) : Devuelve un objeto del tipo que se esta solicitando con el nombre que se está solicitando.

Algunas de las siguientes excepciones se pueden generar al utilizar los métodos anteriores:

- **NoUniqueBeanDefinitionException** : En caso de que existan más de un bean del mismo tipo.

- **NoSuchBeanDefinitionException** : En caso de que no se encuentre el bean que se está solicitando.
- **BeansException** : En caso de que no se pueda crear el bean solicitado.
- **BeanNotOfRequiredTypeException** : En caso de que el bean con el nombre solicitado no sea del tipo especificado.

Injectando beans en otros beans

Una vez que ya sabemos como agregar un bean al contexto de spring y como obtenerlo, el siguiente paso es como hacer dependencias entre ellos, existen 3 formas de hacerlo, veamos una por una injectando el servicio MathService que creamos previamente en la siguiente interfaz.

```
/**
 *
 * @author raidentrance
 *
 */
public interface ComplexCalculatorService {

    /**
     * Calculates the avergae based on a set of
     *
     * @param values
     * @return
     */
    double average(double... values);

}
```



Inyección por constructor

La primera forma que veremos será inyección por constructor:

```
import org.springframework.beans.factory.annotation
import org.springframework.stereotype.Service;

/**
 * @author raidentrance
 *
 */
@Service
public class ComplexCalculatorServiceImpl implements

    private MathService mathService;

    @Autowired
    public ComplexCalculatorServiceImpl(MathService mathService) {
        this.mathService = mathService;
    }

    @Override
    public double average(double... values) {
        double result = mathService.sum(values);
        return result / values.length;
    }
}
```



De acuerdo a lo anterior podemos notar que:

- La clase en la que inyectaremos el service se debe encontrar en el contexto de spring, por esto `ComplexCalculatorServiceImpl` tiene la anotación **@Service**
- La inyección de dependencias se hace a través del constructor, es por esto que le colocamos la anotación

@Autowired en el

Inyección por setter

La siguiente forma de hacerlo será inyección por setter:

```
import org.springframework.beans.factory.annotation
import org.springframework.stereotype.Service;

/**
 * @author raidentrance
 *
 */
@Service
public class ComplexCalculatorServiceImpl implement

    private MathService mathService;

    @Autowired
    public void setMathService(MathService math
        this.mathService = mathService;
    }

    @Override
    public double average(double... values) {
        double result = mathService.sum(val
        return result / values.length;
    }
}
```

A diferencia de la inyección por constructor en la inyección por setter la anotación **@Autowired** se colocará en el método setter.

Inyección por atributo

La siguiente forma de inyectar un atributo es a través de inyección por atributos:

```
import org.springframework.beans.factory.annotation
import org.springframework.stereotype.Service;

/**
 * @author raidentrance
 *
 */
@Service
public class ComplexCalculatorServiceImpl implements

    @Autowired
    private MathService mathService;

    @Override
    public double average(double... values) {
        double result = mathService.sum(val
        return result / values.length;
    }
}
```

Como podemos ver haciendo inyección por atributo la clase luce más simple y limpia, el problema de utilizar este método de inyección es que en el se utiliza reflection dado que el atributo es privado y no se puede acceder desde fuera de la clase.

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://twitter.com/raidentrance)

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

Contacto: raidentrance@gmail.com

Anuncios

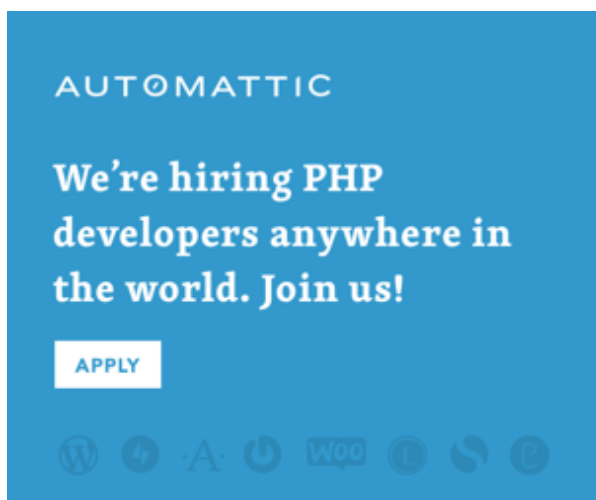


Earn money off your WordPress site

WordAds

REPORT THIS AD

This advertisement features a blue background. On the left, the text "Earn money off your WordPress site" is written in white. Below this text is a black speech bubble containing the word "WordAds" in white. On the right side, there is a white laptop with a blue screen displaying a colorful abstract pattern. At the bottom right, there is a small link that says "REPORT THIS AD".



AUTOMATTIC

We're hiring PHP developers anywhere in the world. Join us!

APPLY

REPORT THIS AD

This advertisement has a solid blue background. At the top, the word "AUTOMATTIC" is written in white, all-caps. Below it, the text "We're hiring PHP developers anywhere in the world. Join us!" is written in white. Underneath this is a white rectangular button with the word "APPLY" in blue. At the bottom, there is a row of seven circular icons: WordPress, GitHub, a stylized 'A', a power button, a speech bubble with "WOO", a person icon, and a circular arrow. At the bottom right, there is a small link that says "REPORT THIS AD".



1 comentario »



Pingback: [Spring framework 5 : Uso de la anotación
@Qualifier – devs4j](#)
