devs4j

EL EJOR SITIO EB SOBRE PROGRA ACI N EN ESPA OL.

ESPAÑOL

Spring framework 5 : Uso de @Autowire para listas de objetos

BY RAIDENTRANCE ON FEBRERO 8, 2019 · (1 COMENTARIO)

En ejemplos anteriores se explicó como inyectar objetos en spring utilizando la anotación @Autowired para objetos simples, en este post explicaremos como inyectar listas de objetos.

Paso 1 Crear las clases a inyectar

El primer paso será crear una lista de clases a inyectar, en este caso tendremos una clase padre y multiples clases hijas, veamos el siguiente código:

```
/**
 * @author raidentrance
 *
 */
public abstract class Figure {

        public abstract double getArea();

}
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Component
public class Circle extends Figure {

	private double radius;

	private static final Logger log = LoggerFact

	public Circle(@Value("10.0") double radius)
		this.radius = radius;
	}

	@Override
	public double getArea() {
		log.info("Calculating the are of a c
		return Math.pow(Math.PI * radius, 2)
	}

}
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Component
public class Square extends Figure {
```

```
        private double side;

        private static final Logger log = LoggerFact

        public Square(@Value("10.0") double side) {
                this.side = side;
        }

        @Override
        public double getArea() {
                log.info("Calculating the are of a s
                return Math.pow(side, 2);
        }

}
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Component
public class Rectangle extends Figure {

        private double width;

        private double height;

        private static final Logger log = LoggerFact

        public Rectangle(@Value("10.0") double width
                this.width = width;
                this.height = height;
        }

        @Override
        public double getArea() {
```

```
                log.info("Calculating the are of a r
                return width * height;
        }

}
```

Se crearon las siguientes clases:

- Figure : Clase abstracta que será padre de las demás clases, define el método abstracto double **getArea**()
- Circle : Define el atributo radio y utiliza la anotación @Value para asignarle el valor de 10.0
- Square : Define el atributo lado y utiliza la anotación @Value para asignarle el valor de 10.0
- Rectangle : Define los atributos ancho y alto y utiliza la anotación @Value para asignarles el valor de 10.0 y 5.0.

Las 3 implementaciones Circle, Square y Rectangle implementan el método **getArea**() definido en la clase Figure.

Inyectando los beans en una referencia de tipo List

Una vez que definimos los beans con sus valores el siguiente paso será inyectarlos en una referencia de tipo List, veamos la siguiente clase:

```
import java.util.List;

import org.springframework.beans.factory.annotation.
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Component
public class AreaCalculator {

        @Autowired
        private List<Figure> figures;

        public double getSumOfAreas() {
                double totalArea = 0.0;
```

```
                for (Figure figure : figures) {
                        totalArea += import java.uti

import org.springframework.beans.factory.annotation.
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Component
public class AreaCalculator {

        @Autowired
        private List<Figure> figures;

        public double getSumOfAreas() {
                double totalArea = 0.0;
                for (Figure figure : figures) {
                        totalArea += figure.getArea
                }
                return totalArea;
        }
}
.getArea();
                }
                return totalArea;
        }
}
```

Como se puede ver la clase AreaCalculator tiene un método que suma el área de todas las figuras, para esto debe inyectar un List<Figure>, con esto Spring inyectará todas las implementaciones de Figure.

Ejecutando la aplicación

El último paso será modificar la clase aplicación para ejecutar el método getSumOfAreas() como se muestra en la siguiente clase:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.Spring
import org.springframework.context.ConfigurableAppli
```

```
import com.devs4j.lists.AreaCalculator;

@SpringBootApplication
public class Devs4jSpringCoreApplication {

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

                AreaCalculator areaCalculator = appl
                System.out.println(areaCalculator.ge
        }

}
```

Al ejecutarlo la salida será la siguiente :

```
2019-02-07 12:01:50.366  INFO 96221 --- [
 2019-02-07 12:01:50.368  INFO 96221 --- [
 2019-02-07 12:01:50.369  INFO 96221 --- [
 2019-02-07 12:01:50.369  INFO 96221 --- [
 1136.960440108936
```

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*

Publicado por raidentrance

Soy @raidentrance en Twitter y en Github, soy egresado de la Facultad de Ingeniería de la UNAM, cuento con 8 certificaciones en diferentes áreas del desarrollo de software, me gustan las cervezas y soy Geek. Ver todas las entradas de raidentrance (https://devs4j.com/author/raidentrance/)

1 reply »

Pingback: Spring framework 5 : Leer información de archivos .properties – devs4j