devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

ESPAÑOL

Spring framework 5 : Bean Aware interfaces

BY RAIDENTRANCE ON FEBRERO 14, 2019 · (DEJA UN COMENTARIO)

Rate This

Aware interface significa en español interfaz consciente, este tipo de interfaces actúan como listeners para ciertos eventos que suceden en Spring framework.

Las siguientes son las Aware interfaces en Spring framework:

Aware interface	Recurso objetivo
BeanNameAware	El nombre del bean de las instancias configuradas en el contenedor
BeanFactoryAware	El bean factory actual a través del cual se invoca el contenedor
ApplicationContextAware	El application context actual a través del cual puedes invocar el contenedor
MessageSourceAware	El message source a través del cual se pueden resolver mensajes

ApplicationeventPublisherAware	El even publisher a través del cual se pueden publicar eventos de la aplicación
ResourceLoaderAware	El resource loader a través del cual puedes cargar recursos externos
EnvironmentAware	La instancia de Environment asociada con el ApplicationContext

Las Aware Interfaces definen métodos setters los cuales se invocan por Spring después de que las propiedades de Spring se asignaron pero antes de que los método callback (postConstruct y preDestroy) se invoquen.

Ejemplo utilizando BeanNameAware

Para entender como funcionan las aware interfaces crearemos el siguiente bean:

```
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;

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

@Component
public class UserService implements BeanNameAware {

        @Value("fakeUser")
        private String user;

        private static final Logger log = LoggerFact

        @PostConstruct
        public void init() {
                log.info("Post construct callback {}
        }

        @Override
        public void setBeanName(String name) {
                log.info("Being aware of {} {}", nam
        }
```

```
        @PreDestroy
        public void destroy() {
                log.info("Pre destroy callback");
        }
}
```

Al ejecutar la aplicación tendremos la siguiente salida:

```
2019-02-14 16:01:55.319  INFO 16934 --- [
2019-02-14 16:01:55.321  INFO 16934 --- [
2019-02-14 16:01:55.686  INFO 16934 --- [
2019-02-14 16:01:55.687  INFO 16934 --- [
2019-02-14 16:01:55.833  INFO 16934 --- [
2019-02-14 16:01:55.836  INFO 16934 --- [       Thre
```

Este tipo de interfaces se utilizan para logging o para realizar integraciones con código legado, en la mayoría de los casos debemos evitar utilizarlas debido a que acoplan nuestra aplicación a Spring framework.

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/)