



Inicio » Enterprise Java » spring » Arranque » Tutorial de anotaciones de arranque de Spring

## SOBRE SANTOSH BALGAR



Es ingeniero de software y trabaja en una organización líder en la industria. Ha completado su licenciatura de la Universidad Tecnológica de Visweswaraya. En su carrera, ha trabajado en el diseño e implementación de varios sistemas de software que involucran Java / J2EE, Spring / Spring Boot, React JS, JQuery, Hibernate y tecnologías de bases de datos relacionadas. Le encanta compartir su conocimiento y siempre espera aprender y explorar nuevas tecnologías. Le encanta pasar su tiempo libre con su familia. Le gusta viajar y le encanta jugar al cricket.

## Tutorial de anotaciones de arranque de primavera

Publicado por: Santosh Balgar en el arranque 28 de febrero de 2020 0 338 reproducciones

En esta publicación, presentamos un completo tutorial de anotaciones de arranque de Spring. Cuando se introdujo Spring inicialmente, los desarrolladores usaban principalmente la configuración basada en XML. Con la introducción del revolucionario framework Spring Boot, ahora los desarrolladores se han alejado completamente de la configuración basada en XML y es difícil imaginar el desarrollo sin usar anotaciones.

En este artículo, voy a discutir las anotaciones básicas de Spring / Spring Boot, @SpringBootApplication, @EnableAutoConfiguration, @Conditional, @ComponentScan, @Bean, @Service, @Component, @Controller, @Autowired

etc.

### ¿Quieres dominar Spring Framework?



Suscríbete a nuestro boletín y descarga el **Spring Framework Cookbook** ahora mismo.

Para ayudarlo a dominar el marco Java innovador y líder, hemos compilado una guía increíble con todas sus características principales y casos de uso. ¡Además de estudiarlos en línea, puede descargar el libro electrónico en formato PDF!

[¡Descargar ahora!](#)

## 1. @SpringBootApplication

La clase principal en la aplicación Spring Boot se anota con

```
@SpringBootApplication
```

La aplicación Spring Boot trata sobre la autoconfiguración de varios recursos. Lo hace por escaneo de componentes. Al escanear clases con

### BOLETIN INFORMATIVO

**¡145,523** personas con **información** privilegiada de actualizaciones semanales **blancos** gratuitos! **Únase a ellos ahora** acceso exclusivo a las en el mundo de Java, así como sobre Android, Scala, Groovy tecnologías relacionadas.

**Dirección de correo electrónico:**

Tu correo electrónico

☒ Reciba alertas de trabajo desarrollador en su área

[Regístrate](#)

**ÚNETE A NOSOTROS**

- `@ComponentScan`

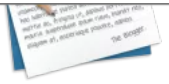
- Esta anotación le dice al marco de Spring Boot que identifique todos los componentes bajo el mismo paquete o todos sus subpaquetes. Opcionalmente, incluso podemos especificar los paquetes a escanear.

- `@EnableAutoConfiguration`

- Esta anotación configura automáticamente todos los beans en el classpath. Prepara los beans inicializando todas las dependencias requeridas.

- `@SpringBootApplication`

- Esta es una anotación de nivel de clase e indica que la clase es una clase de configuración de aplicación. En general, esta clase tiene las definiciones de bean.



## 2. Anotaciones condicionales

Se pueden usar anotaciones condicionales en los componentes. Esto nos permite especificar si la configuración / bean / método anotado es elegible para registrarse en el contenedor o no en función de algunas condiciones. Spring Boot lleva la

`@Conditional`

anotación al siguiente nivel al proporcionar varias

`@Conditional*`

anotaciones predefinidas en el paquete

`org.springframework.boot.autoconfigure.condition`

.

- `@ConditionalOnClass`

y

`@ConditionalOnMissingClass`

- Si alguna clase se cargará solo si hay alguna otra clase disponible, úsela

`@ConditionalOnClass`

. Si una clase se cargará solo si no hay otra clase disponible

`ApplicationContext`

, utilízela

`@ConditionalOnMissingClass`

.

- `@ConditionalOnBean`

y

`@ConditionalOnMissingBean`

- Cargue el bean solo si cierto bean está en el contexto de la aplicación o si falta un cierto bean en el contexto de la aplicación.

- `@ConditionalOnProperty`

- Esta es probablemente la anotación condicional más utilizada. Nos permite cargar ciertos beans solo cuando se establece una propiedad específica en la configuración.

- `@ConditionalOnResource`

- Cargue algunos beans solo si cierto recurso está presente en el classpath. Un caso de uso útil es cargar / habilitar el registro solo cuando `logback.xml` está presente en el classpath.

- `@ConditionalOnWebApplication`

y

`@ConditionalOnNotWebApplication`

- Cargar el bean si estamos ejecutando una aplicación web o cargar cuando no es una aplicación web.

---

Spring Boot tiene soporte para todas las anotaciones básicas de Spring. A continuación se muestran algunas de las anotaciones principales compatibles con Spring / Spring Boot.

- @Component

- Es una anotación genérica para indicar que la clase es administrada por el contenedor Spring

- @Bean

- Esta es una anotación de nivel de método e indica que el método produce un bean administrado por contenedor, esto es un reemplazo para la

```
<bean/>
```

etiqueta en la configuración basada en XML

- @Service

- Esta anotación es una anotación de nivel de clase e indica que la clase posee la lógica empresarial

- @Configuration

- Las clases marcadas con esta anotación son procesadas por el contenedor Spring para generar definiciones de bean

- @Controller

- Esta es una especialización de

```
@Component
```

anotación y generalmente se usa en combinación con la

```
@RequestMapping
```

anotación.

```
@RestController
```

La anotación simplifica la creación del servicio REST.

- @RequestMapping

- Esta anotación asigna las solicitudes HTTP de Web / Rest API a los métodos del controlador.

```
@GetMapping, @PostMapping, @PutMapping
```

son las implementaciones especiales de

```
@RequestMapping
```

.

- @Repository

- El repositorio Spring está muy cerca del patrón DAO y simplifica la implementación de la operación CRUD

- @Autowired

- Esta anotación permite a Spring Framework resolver e inyectar las dependencias. Se puede usar en propiedades, métodos de establecimiento o con constructor

- @Qualifier: se utiliza para resolver los conflictos de nombres entre los beans del mismo tipo

## 4. Anotaciones de prueba

En esta sección, voy a discutir brevemente algunas de las anotaciones de prueba de Spring Boot. La prueba de arranque de Spring requiere dependencia

```
spring-boot-starter-test
```

en el archivo maven. Esto nos permite ejecutar pruebas de Unidad e Integración.

```
@RunWith(SpringRunner.class)
```

proporciona el puente entre Spring Boot y JUnit. Esto es necesario para todas las pruebas de Spring Boot. A continuación se presentan algunas de las anotaciones de Spring Boot Test,

- @DataJpaTest

- `@TestConfiguration`

- Indica que los beans creados con la anotación `@Bean` en las clases de prueba no deben recogerse durante el escaneo

- `@WebMvcTest`

- Para probar las clases del controlador, usamos esta anotación. Configura automáticamente la infraestructura Spring MVC para nuestras pruebas.

- `@SpringBootTest`

- Esto nos permite escribir pruebas de integración. Esto inicia todo el contenedor Spring y crea el

`ApplicationContext`

que se utilizará en las pruebas.

El ejemplo proporcionado en este artículo no cubre las anotaciones de prueba. Puede escribir sus propias pruebas para obtener más información sobre las anotaciones de Spring Boot Test.

## 6. Ejemplo

Este ejemplo muestra algunas de las anotaciones de Spring Boot. Se pueden probar anotaciones adicionales como ejemplo.

### SpringBootApplication

```

1 | @SpringBootApplication
2 | @ComponentScan(basePackages = {"com.jcg.example.controllers", "com.jcg.example.services"})
3 | public class SpringBootApplication {
4 |     public static void main(String[] args) {
5 |         SpringApplication.run(SpringBootApplication.class, args);
6 |     }
7 | }

```

`@SpringBootApplication`

se agrega en la clase principal y realiza la configuración inicial para la aplicación Spring Boot.

`@ComponentScan`

habilita el escaneo automático de clases anotadas.

### HelloWorldController

```

01 | @RestController
02 | public class HelloWorldController {
03 |     private final HelloWorldService service;
04 |
05 |     public HelloWorldController(HelloWorldService service) {
06 |         this.service = service;
07 |     }
08 |
09 |     @GetMapping(value="/hello", produces = MediaType.TEXT_PLAIN_VALUE)
10 |     public String sayHelloWorld() {
11 |         return service.sayMessage();
12 |     }
13 | }

```

La clase anterior está marcada con la anotación

`@RestController`

. Como hemos habilitado la exploración de componentes en el paquete

`com.jcg.example.controllers`

, el contenedor detecta y prepara automáticamente las clases marcadas con anotaciones Spring Boot en este paquete.

Aquí estoy usando inyección de constructor. Si desea utilizar DI basada en setter, puede utilizar la anotación

`@Autowired`

en el bean

`HelloWorldService`

A continuación se muestra el fragmento para usar una de las anotaciones condicionales

`@ConditionalOnResource`

de la carpeta de recursos en el proyecto de ejemplo y vuelva a ejecutar.

MySQLDatabaseService

```
1 | @ConditionalOnResource(  
2 |     resources = "classpath:mysql.properties")  
3 | @Service  
4 | public class MySQLDatabaseService {  
5 |     //This class is available only if mysql.properties is present in the classpath  
6 | }
```

Todos los sabores diferentes de las anotaciones condicionales se pueden probar como ejercicio.

## 7. Descargue el código fuente

### Descargar

Puede descargar el código fuente completo de este ejemplo aquí: **Tutorial de anotaciones de arranque de Spring**

## ¿Quieres saber cómo desarrollar tus habilidades para convertirte en un Rockstar de Java?

¡Suscríbete a nuestro boletín para comenzar a rockear ahora mismo!

Para que comiences, te ofrecemos nuestros eBooks más vendidos

**GRATIS!**

1. JPA Mini libro
2. Guía de resolución de problemas de JVM
3. Tutorial JUnit para pruebas unitarias
4. Tutorial de anotaciones de Java
5. Preguntas de la entrevista Java
6. Preguntas de la entrevista de primavera
7. Diseño de la interfaz de usuario de Android

y muchos más ....

**Dirección de correo electrónico:**

☒ Reciba alertas de trabajo de Java y desarrollador en su área

Regístrate

¿TE GUSTA ESTE ARTÍCULO? LEER MÁS DE JAVA CODE GEEKS

Microservice Framework - Open Api Runtime

Ad grizzly-api.com

Java Annotations Tutorial

javacodegeeks.com

Organic Vapor Full Face Respirator...

Ad Wish

[MEGA DEAL] The 2020 Complete Java Master Class Bundle (96%) | Java Code...

javacodegeeks.com

Real-time API Collaboration

Ad Postman

Docker Basic Commands | Examples Java Code Geeks

javacodegeeks.com


Spring Boot Hello World Example | Examples Java Code Geeks

javacodegeeks.com

Hibernate Ex Code Using Annotations

javacodegeeks.com

Deja una respuesta

 Comience la discusión ...

Este sitio usa Akismet para reducir el spam. Aprenda cómo se procesan sus datos de comentarios .

☒ Suscribirse ▼

KNOWLEDGE BASE

- Courses
- Minibooks
- News
- Resources
- Tutorials

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

HALL OF FAME

- Android Alert Dialog Example
- Android OnClickListener Example
- How to convert Character to String and a String to Character Array in Java
- Java Inheritance example
- Java write to File Example
- java.io.FileNotFoundException – How to solve File Not Found Exception
- java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception
- java.lang.NoClassDefFoundError – How to solve No Class Def Found Error
- JSON Example With Jersey + Jackson
- Spring JdbcTemplate Example

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev. JCGs serve the Java, SOA, Agile and Telecom communities with daily news, domain experts, articles, tutorials, reviews, announcements, code snippets, source projects.

DISCLAIMER

Todas las marcas comerciales y marcas comerciales registradas que aparecen en este sitio son propiedad de sus respectivos dueños. Java es una marca registrada de Oracle Corporation en los Estados Unidos y en otros países. Ejemplos Java Code Geeks no está conectado a Oracle Corporation patrocinado por Oracle Corporation.