# **Preguntas** **de la entrevista de Spring Boot**

**0.** **Spring vs Spring Boot**

|  |  |
| --- | --- |
| Spring | Spring Boot |
| Un framework de aplicaciones web basado en Java | Un modulo de Spring |
| Proporciona herramientas y bibliotecas para crear aplicaciones web personalizadas. | Se utiliza para crear un proyecto de aplicación Spring que solo se puede run/execute |
| Spring es más complejo que Spring Boot | Spring Boot es menos complejo que Spring Framework |
| Adopta una opinión obstinada | Tomar como punto de vista obstinado de una plataforma |

**1¿Qué es Spring Boot y mencionar la necesidad de ello?**

Spring Boot es un módulo Spring que tiene como objetivo simplificar el uso del marco Spring para el desarrollo de Java. Se utiliza para crear aplicaciones independientes basadas en Spring que puede ejecutar.

Entonces, básicamente elimina muchas configuraciones y dependencias. Con el objetivo del desarrollo rápido de aplicaciones, el marco Spring Boot viene con la resolución de dependencia automática, servidores HTTP integrados, configuración automática, puntos finales de administración y Spring Boot CLI.

Entonces, si me pregunta por qué alguien debería usar Spring Boot, entonces diría que Spring Boot no solo mejora la productividad, sino que también brinda muchas comodidades para escribir su propia lógica comercial.

**2.** **Mencione las ventajas de Spring Boot**

**-** Proporciona configuración automática para cargar un conjunto de configuración predeterminada para un inicio rápido de la aplicación

- Crea aplicaciones independientes con una gama de características no funcionales que son comunes a grandes clases de proyectos

- Viene con tomcat integrado, embarcadero de contenedores de servlet para evitar el uso de archivos WAR

- Spring Boot proporciona una vista obstinada para reducir el esfuerzo del desarrollador y simplificar las configuraciones de maven

- Proporciona una herramienta CLI para desarrollar y probar aplicaciones

- Viene con arrancadores de Spring Boot para garantizar la gestión de dependencias y también proporciona varias métricas de seguridad

- Consiste en una amplia gama de API para monitorear y administrar aplicaciones en desarrollo y producción.

- Se integra con Spring Ecosystem como Spring JDBC, Spring ORM, Spring Data, Spring Security fácilmente al evitar el código repetitivo.

**3**. **Mencione algunas características de Spring Boot.**

1. Spring CLI: Spring Boot CLI le permite Groovy para escribir la aplicación de arranque Spring y evita el código repetitivo.

2. Dependencia de inicio:

Con la ayuda de esta característica, Spring Boot agrega dependencias comunes y finalmente mejora la productividad.

3. Configuración automática:

La función de configuración automática de Spring Boot ayuda a cargar las configuraciones predeterminadas de acuerdo con el proyecto en el que está trabajando. De esta manera, puede evitar archivos WAR innecesarios.

4. Spring Initializer:

Esta es básicamente una aplicación web, que puede crear una estructura de proyecto interna para usted. Por lo tanto, no tiene que configurar manualmente la estructura del proyecto, sino que puede usar esta función.

5. Spring Actuator:

Esta característica proporciona ayuda mientras se ejecutan aplicaciones Spring Boot.

6. Registro y seguridad:

La función de registro y seguridad de Spring Boot garantiza que todas las aplicaciones creadas con Spring Boot estén debidamente protegidas sin problemas.

**4**. **Explicar cómo crear una aplicación Spring Boot usando Maven.**

Bueno, hay varios enfoques para crear una aplicación Spring Boot usando maven, pero si tengo que nombrar algunos, las siguientes son las formas de crear un proyecto/aplicación Spring Boot usando maven:

- CLI de arranque de primavera

- Asistente para proyectos Spring Starter

- Spring Initializr

- Proyecto Spring Maven

**5**. **Mencione las posibles fuentes de configuración externa.**

No hay duda de que Spring Boot permite a los desarrolladores ejecutar la misma aplicación en diferentes entornos. Bueno, esto se hace con el soporte que brinda para la configuración externa. Utiliza variables de entorno, archivos de propiedades, argumentos de línea de comandos, archivos YAML y propiedades del sistema para mencionar las propiedades de configuración requeridas. Además, la anotación @value se usa para obtener acceso a las propiedades. Por lo tanto, las fuentes más posibles de configuración externa son las siguientes:

- Propiedades de la aplicación :

De forma predeterminada, Spring Boot busca el archivo de propiedades de la aplicación o su archivo YAML en el directorio actual, raíz de classpath o directorio de configuración para cargar las propiedades.

- Propiedades de línea de comandos :

Spring Boot proporciona argumentos de línea de comandos y convierte estos argumentos en propiedades. Luego los agrega al conjunto de propiedades del entorno.

- Propiedades específicas del perfil :

Estas propiedades se cargan desde el archivo application-{profile}.properties o su archivo YAML. Este archivo se encuentra en la misma ubicación que los archivos de propiedades no específicas y el marcador de posición {profile} hace referencia a un perfil activo.

**6**. **¿Puede explicar qué sucede en segundo plano cuando una aplicación Spring Boot es "Ejecutar como aplicación Java"?**

Cuando una aplicación Spring Boot se ejecuta como "Ejecutar como aplicación Java", automáticamente inicia el servidor tomcat tan pronto como ve que está desarrollando una aplicación web.

**7**. **¿Qué son los arrancadores Spring Boot y cuáles son los arrancadores disponibles?**

Los iniciadores de Spring Boot son un conjunto de proveedores de administración de dependencias convenientes que se pueden usar en la aplicación para habilitar las dependencias. Estos iniciadores hacen que el desarrollo sea fácil y rápido. Todos los titulares disponibles están bajo la organización. marco de resorte. grupo de arranque. Algunos de los entrantes populares son los siguientes:

° spring-boot-starter:

Este es el iniciador principal e incluye registro, soporte de configuración automática y YAML.

° spring-boot-starter-jdbc:

Este iniciador se usa para el grupo de conexiones HikariCP con JDBC

° spring-boot-starter-web:

Este es el iniciador para crear aplicaciones web, incluidas las aplicaciones RESTful, que utilizan Spring MVC

° spring-boot-starter-data-jpa:

Es el iniciador para usar Spring Data JPA con Hibernate

° spring-boot-starter-security:

Es el iniciador utilizado para Spring Security

° spring-boot-starter-aop:

Este iniciador se utiliza para la programación orientada a aspectos con AspectJ y Spring AOP

° spring-boot-starter-test:

Es el iniciador para probar aplicaciones Spring Boot

**8**. **Explicar Spring Actuator y sus ventajas.**

Spring Actuator es una característica genial de Spring Boot con la ayuda de la cual puede ver lo que sucede dentro de una aplicación en ejecución.

Entonces, cada vez que desee depurar su aplicación y necesite analizar los registros, necesita comprender qué está sucediendo en la aplicación, ¿verdad? En tal escenario, Spring Actuator proporciona un fácil acceso a funciones como la identificación de beans, el uso de CPU, etc.

Spring Actuator proporciona una manera muy fácil de acceder a los puntos REST listos para producción y obtener todo tipo de información de la web. Estos puntos se aseguran mediante la estrategia de negociación de contenido de Spring Security.

**9**. **¿Qué es la gestión de dependencias de Spring Boot?**

La administración de dependencias de Spring Boot se usa básicamente para administrar las dependencias y la configuración automáticamente sin que usted especifique la versión para ninguna de esas dependencias.

## **P11. Mencione los requisitos mínimos para un sistema de arranque Spring.**

Spring Boot 2.1.7.RELEASE requiere

* Java 8 +
* marco de primavera 5.1.9 +

Soporte de compilación explícito

* Experto 3.3+
* Gradle 4.4+

Compatibilidad con contenedores de servlet

* Tomcat 9.0: versión de servlet 4.0
* Embarcadero 9.4 — Servlet Versión 3.1
* Resaca 2.0 — Servlet Versión 4.0

**P12**. **Explique qué es la hoja de tomillo y cómo usar la hoja de tomillo.**

Thymeleaf es un motor de plantillas Java del lado del servidor que se utiliza para aplicaciones web. Su objetivo es traer una plantilla natural para su aplicación web y puede integrarse bien con Spring Framework y aplicaciones web HTML5 Java. Para usar Thymeleaf, debe agregar el siguiente código en el archivo pom.xml:

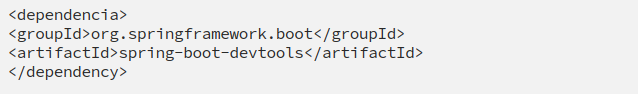
## 

## **P13. ¿Podemos cambiar el puerto del servidor Tomcat incorporado en Spring boot?**

Sí, podemos cambiar el puerto del servidor tomcat integrado mediante el archivo de propiedades de la aplicación. En este archivo, debe agregar una propiedad de "servidor.puerto" y asignarlo a cualquier puerto que desee. Por ejemplo, si desea asignarlo a 8081, debe mencionar server.port=8081. Una vez que mencione el número de puerto, Spring Boot cargará automáticamente el archivo de propiedades de la aplicación y las configuraciones requeridas se aplicarán a la aplicación.

## **P14. ¿Cuál es la necesidad de Spring Boot DevTools?**

## Spring Boot Dev Tools son un conjunto elaborado de herramientas y tienen como objetivo facilitar el proceso de desarrollo de una aplicación. Si la aplicación se ejecuta en producción, este módulo se desactiva automáticamente, el reempaquetado de archivos también se excluye de forma predeterminada. Por lo tanto, Spring Boot Developer Tools aplica propiedades a los respectivos entornos de desarrollo. Para incluir DevTools, solo tiene que agregar la siguiente dependencia en el archivo pom.xml:



## **P15. Mencione los pasos para crear un proyecto Spring Boot usando Spring Initializer.**

## Spring Initializr es una herramienta web proporcionada por Spring. Con la ayuda de esta herramienta, puede crear proyectos de Spring Boot simplemente proporcionando los detalles del proyecto. Se deben seguir los siguientes pasos para crear un proyecto Spring Boot usando Spring Initializer:

## - Elija el proyecto maven y las dependencias requeridas. Luego, complete los otros detalles requeridos como Grupo, Artefacto y luego haga clic en Generar proyecto.

## - Una vez descargado el proyecto, extráigalo a su sistema

## - A continuación, debe importar este proyecto utilizando la opción de importación en Spring Tool Suite IDE

- Al importar el proyecto, recuerde que debe elegir el tipo de proyecto para que sea Maven y el proyecto de origen debe contener el archivo pom.xml.

Una vez que haya seguido todos los pasos anteriores, verá que el proyecto Spring Boot se crea con todas las dependencias requeridas.

**P16. Mencione los pasos para conectar la aplicación Spring Boot a una base de datos usando JDBC.**

Los proyectos iniciales de Spring Boot proporcionan las bibliotecas necesarias para conectar la aplicación con JDBC. Entonces, por ejemplo, si solo tiene que crear una aplicación y conectarla con la base de datos MySQL, puede seguir los pasos a continuación:

**Paso 1:** Crea una base de datos en MySQL

CREATE DATABASE example;

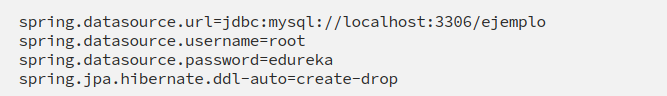
**Paso 2:** luego debe crear una tabla dentro de esta base de datos.

CREATE TABLE clients(customerid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT, customername VARCHAR(255));

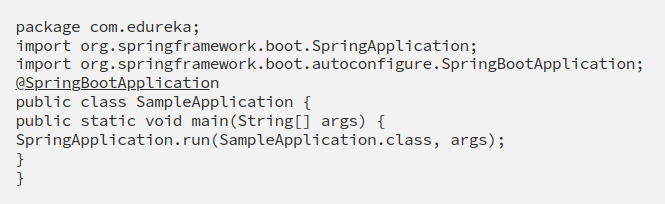
**Paso 3**: ahora, cree un proyecto Spring Boot y proporcione los detalles requeridos

**Paso 4**: agregue las dependencias JDBC, MySQL y web.

**Paso 5**: una vez que se crea el proyecto, debe configurar la base de datos en las propiedades de la aplicación

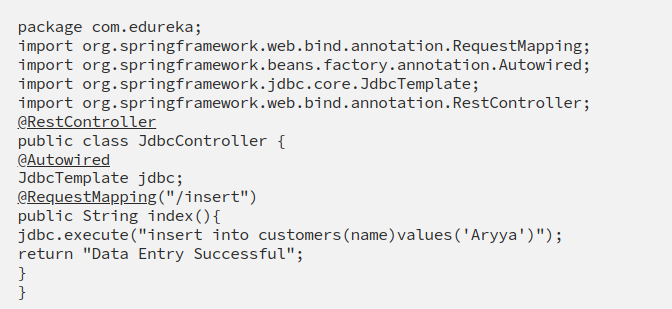


**Paso 6:** La clase principal application.java debe tener el siguiente código:



Paso 7: A continuación, debe crear un controlador para manejar las solicitudes HTTP, mencionando el siguiente

código:



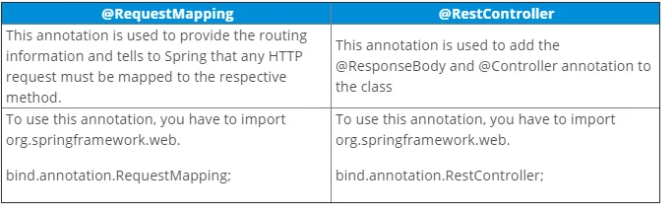
Paso 8: Finalmente, ejecute este proyecto como una aplicación Java.

Paso 9: A continuación, abra la URL (localhost:8080/insert) y verá el resultado como Entrada de datos exitosa. También puede avanzar y verificar si los datos se ingresaron en la tabla.

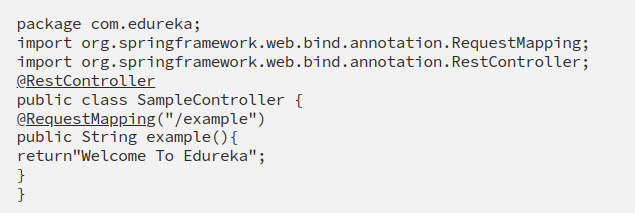
**P17. ¿Cómo habilitar la compatibilidad con HTTP/2 en Spring Boot?**

Puede habilitar la compatibilidad con HTTP/2 en Spring Boot:server.http2.enabled=true

**P18. ¿Para qué se usan las anotaciones @RequestMapping y @RestController en Spring Boot?**

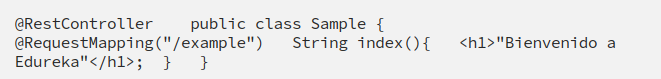


Ejemplo: Considere que tiene un método ejemplo() que debería mapearse con /URL de ejemplo.



## **P19. ¿Qué es Spring Boot CLI y cómo ejecutar el proyecto Spring Boot usando boot CLI?**

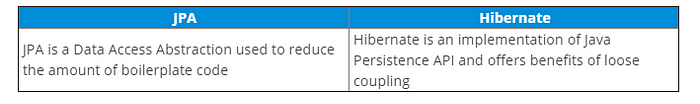
## Spring Boot CLI es una herramienta compatible con Spring Framework oficial. Los pasos para ejecutar un proyecto Spring Boot son los siguientes:

* Descargue la herramienta CLI del sitio oficial y extraiga el archivo zip. La carpeta bin presente en la configuración de Spring se usa para ejecutar la aplicación Spring Boot.
* Dado que Spring Boot CLI ejecuta archivos maravillosos, debe crear un archivo maravilloso para la aplicación Spring Boot. Entonces, para hacer eso, abra la terminal y cambie el directorio actual a la carpeta bin. Ahora, abre un archivo groovy (por ejemplo, Sample. groovy)
* En este archivo, cree un controlador de la siguiente manera:
* Luego ejecute el archivo groovy mencionando:



Una vez que se ejecuta el proyecto, vaya a la URL (localhost: 8080: / ejemplo) y verá el resultado como Bienvenido a Edureka

## **Q20**. **Mencione las diferencias entre JPA e Hibernate.**



## **P21. ¿Cómo podemos crear un punto final personalizado en Spring Boot Actuator?**

Para crear un punto final personalizado en Spring Boot 2.x, puede usar la anotación @Endpoint. Spring Boot también expone puntos finales usando @WebEndpointor, @WebEndpointExtension sobre HTTP con la ayuda de Spring MVC, Jersey, etc.

## **P22. Explique los datos de Spring.**

Spring Data tiene como objetivo facilitar a los desarrolladores el uso de bases de datos relacionales y no relacionales, servicios de datos basados ​​en la nube y otras tecnologías de acceso a datos. Entonces, básicamente, facilita el acceso a los datos y aún conserva los datos subyacentes.

## **P23. ¿Qué entiendes por autoconfiguración en Spring Boot y cómo deshabilitar la autoconfiguración?**

La configuración automática se utiliza para configurar automáticamente la configuración necesaria para la aplicación. Por ejemplo, si tiene un bean de fuente de datos presente en el classpath de la aplicación, automáticamente configura la plantilla JDBC. Con la ayuda de la configuración automática, puede crear una aplicación Java de una manera fácil, ya que configura automáticamente los beans, controladores, etc. necesarios.

Para deshabilitar la propiedad de configuración automática, debe excluir el atributo de @EnableAutoConfiguration, en el escenario en el que no desea que se aplique.

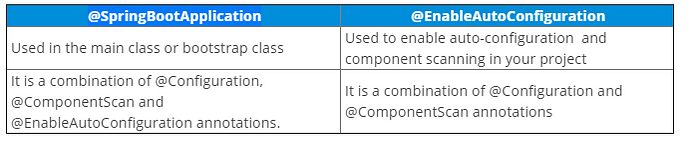
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})

Si la clase no está en el classpath, para excluir la configuración automática, debe mencionar el siguiente código:

@EnableAutoConfiguration(excludeName={Sample.class})

Aparte de esto, Spring Boot también ofrece la posibilidad de excluir una lista de clases de configuración automática mediante el uso de la spring.autoconfigure.excludepropiedad. Puede continuar y agregarlo en la aplicación. properties o agregue varias clases separadas por comas.

## **P24. ¿Cuáles son las diferencias entre la anotación @SpringBootApplication y @EnableAutoConfiguration?**



**P25. ¿Cuáles son los pasos para implementar aplicaciones web Spring Boot como archivos JAR y WAR?**

Para implementar una aplicación web Spring Boot, solo debe agregar el siguiente complemento en el archivo pom.xml:

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

Al usar el complemento anterior, obtendrá un JAR que ejecuta la fase del paquete. Este JAR contendrá todas las bibliotecas y dependencias necesarias requeridas. También contendrá un servidor integrado. Entonces, básicamente puede ejecutar la aplicación como un archivo JAR ordinario.

Nota: El elemento de empaquetado en el archivo pom.xml debe establecerse en jar para crear un archivo JAR como se muestra a continuación:

<packaging>jar</packaging>

Del mismo modo, si desea crear un archivo WAR, mencionará

<packaging>war</packaging>

## **P.26 ¿Puede dar un ejemplo para ReadOnly como verdadero en la gestión de transacciones?**

Un ejemplo de ReadOnly como TRUE en la gestión de transacciones podría ser el siguiente:

Considere un escenario en el que tiene que leer datos de la base de datos. Por ejemplo, supongamos que tiene una base de datos de clientes y desea leer los detalles del cliente, como el cliente y el nombre del cliente. Para hacerlo, establecerá la transacción en modo de solo lectura, ya que no queremos verificar los cambios en las entidades.

## **P27. ¿Puede explicar cómo implementar en un servidor diferente con Spring Boot?**

Para implementar un servidor diferente con Spring Boot, siga los pasos a continuación:

* Generar un WAR del proyecto
* Luego, implemente el archivo WAR en su servidor favorito

Nota: Los pasos para implementar el archivo WAR en el servidor dependen del servidor que elija.

## **P28: ¿Cuál es la mejor manera de exponer la configuración de aplicaciones personalizadas con Spring Boot?**

Una forma de exponer la [**configuración de la aplicación personalizada en Spring**](https://www.edureka.co/blog/spring-tutorial/) Boot es usar la anotación @Value . Pero el único problema con esta anotación es que todos los valores de configuración se distribuirán por toda la aplicación. En su lugar, puede utilizar un enfoque centralizado.

Por enfoque centralizado, quiero decir que puede definir un componente de configuración utilizando @ConfigurationProperties de la siguiente manera:

[**@ConfigurationPropert**](http://twitter.com/ConfigurationPropert) ies("ejemplo")

public class SampleConfiguration {

private int number;

valor booleano privado;

mensaje de cadena privado;

Según el fragmento anterior, los valores configurados en application. propiedades serán las siguientes:

ejemplo.número: 100

ejemplo.valor: verdadero

ejemplo.mensaje: Mensaje dinámico

## **P29. ¿Podemos crear una aplicación no web en Spring Boot?**

Sí, podemos crear una aplicación no web eliminando las dependencias web del classpath y cambiando la forma en que Spring Boot crea el contexto de la aplicación.

## **P 30. ¿Cuáles son los pasos para conectar una base de datos externa como MySQL u Oracle?**

Para conectar una base de datos externa, debe seguir los siguientes pasos:

* Comience agregando la dependencia de MySQL Connector a pom.xml
* Luego elimine la dependencia H2 de pom.xml
* Ahora, configure su [**base de datos MySQL**](https://www.edureka.co/blog/mysql-tutorial/) y configure su conexión a la base de datos MySQL
* Reinicia tu proyecto

## **P31. Mencione las ventajas del archivo YAML sobre el archivo de propiedades y las diferentes formas de cargar el archivo YAML en Spring Boot.**

La ventaja del archivo YAML sobre un archivo de propiedades es que los datos se almacenan en un formato jerárquico. Por lo tanto, se vuelve muy fácil para los desarrolladores depurar si hay un problema. La clase SpringApplication admite el archivo YAML como una alternativa a las propiedades cada vez que usa la biblioteca SnakeYAML en su classpath. Las diferentes formas de cargar un archivo YAML en Spring Boot son las siguientes:

* Use YamlMapFactoryBean para cargar YAML como un mapa
* Use YamlPropertiesFactoryBean para cargar YAML como propiedades

## **P32. ¿Cómo se elige Hibernate como la implementación predeterminada para JPA sin ninguna configuración?**

Cuando usamos la configuración automática de Spring Boot, automáticamente se agrega la dependencia spring -boot-starter-data-jpa al archivo pom.xml. Ahora, dado que esta dependencia tiene una dependencia transitiva en JPA e Hibernate, Spring Boot configura automáticamente Hibernate como la implementación predeterminada para JPA, cada vez que ve Hibernate en el classpath.

## **P33. ¿Qué entiendes por Spring Data REST?**

Spring Data REST se utiliza para exponer los recursos RESTful en torno a los repositorios de Spring Data. Considere el siguiente ejemplo:

[**@RepositoryRestResour**](http://twitter.com/RepositoryRestResour) ce(collectionResourceRel = "muestra", ruta = "muestra")

interfaz pública SampleRepository

extends CustomerRepository<muestra, Long> {

Ahora, para exponer los servicios REST, puede usar el método POST de la siguiente manera:

{

"nombre del cliente": "Rohit"

}

Contenido de la respuesta

{

"nombre del cliente": "Rohit"

"\_links": {

"self": {

"href": " <a href=" http://localhost:8080/sample/1 "> [**http://localhost:8080/sample /1**](http://localhost:8080/sample/1) </a>"

},

"muestra": {

"href": " <a href=" http://localhost:8080/muestra/1 "> [**http://localhost:8080/muestra/1**](http://localhost:8080/sample/1) </ a>"

}

}

Observe que el contenido de la respuesta contiene el href del recurso recién creado.

## **P34. ¿Cuál es la diferencia entre RequestMapping y GetMapping?**

@GetMapping es una anotación compuesta que actúa como acceso directo para @RequestMapping(method = RequestMethod.GET). Ambos métodos apoyan a los consumidores. Las opciones de consumo son:

consume = “texto/simple”  
consume = {“texto/simple”, “aplicación/\*”}

## **P35. ¿En qué capa debe comenzar el límite de una transacción?**

El límite de la transacción debe comenzar desde la capa de servicio ya que la lógica de la transacción comercial está presente en esta misma capa.

## **P36. ¿Cómo funciona path=”sample”, collectionResourceRel=”sample” con Spring Data Rest?**

## [**@RepositoryRestResour**](http://twitter.com/RepositoryRestResour) ce(collectionResourceRel = "muestra", ruta = "muestra")

## interfaz pública SampleRepository extiende

## PagingAndSortingRepository<Sample, Long>

* ruta: esta sección se usa para mencionar el segmento bajo el cual se exportará el recurso.
* collectionResourceRel: este valor se usa para generar enlaces al recurso de la colección.

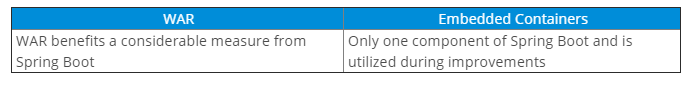
## **P37. Explicar cómo registrar una configuración automática personalizada.**

Para registrar una clase de configuración automática, debe mencionar el nombre completo en la clave @EnableAutoConfiguration META-INF/spring. archivo de fábricas. Además, si construimos con maven, entonces este archivo debe colocarse en el directorio resources/META-INT.

## **P38. ¿Cómo se configura Log4j para iniciar sesión?**

Dado que Spring Boot admite Log4j2 para registrar una configuración, debe excluir Logback e incluir Log4j2 para iniciar sesión. Esto solo se puede hacer si está utilizando el proyecto de inicio.

## **P39. Mencione las diferencias entre WAR y los contenedores integrados**



## **P40. ¿Cuál crees que es la necesidad de Perfiles?**

Los perfiles se utilizan para proporcionar una forma de segregar las diferentes partes de la configuración de la aplicación y hacer que esté disponible para varios entornos. Entonces, básicamente, cualquier @Component o una @Configuración se puede marcar con un @Perfil para limitar a medida que se carga. Considere que tiene varios entornos,

* desarrollador
* control de calidad
* Escenario
* Producción

Ahora, digamos que desea tener diferentes configuraciones de aplicaciones en cada uno de los entornos, puede usar perfiles para tener diferentes configuraciones de aplicaciones para diferentes entornos. Entonces, básicamente, Spring y Spring Boot brindan funciones a través de las cuales puede especificar:

* El perfil activo para un entorno específico
* La configuración de varios entornos para varios perfiles.

## **P41. ¿Cuáles son los pasos para agregar un código JS personalizado con Spring Boot?**

Los pasos para agregar un código JS personalizado con Spring Boot son los siguientes:

* Ahora, cree una carpeta y asígnele un nombre estático en la carpeta de recursos
* En esta carpeta, puede poner el contenido estático en esa carpeta

Nota: Por si acaso, el navegador arroja un error no autorizado, deshabilita la seguridad o busca la contraseña en el archivo de registro y, finalmente, la pasa en el encabezado de la solicitud.

## **P42. ¿Cómo instruir a una configuración automática para que retroceda cuando existe un bean?**

Para indicar a una clase de configuración automática que retroceda cuando existe un bean, debe usar la anotación @ConditionalOnMissingBean. Los atributos de esta anotación son los siguientes:

* valor: este atributo almacena el tipo de frijoles que se verificarán
* nombre: este atributo almacena el nombre de los frijoles que se verificarán

## **P43. ¿Por qué Spring Data REST no se recomienda en aplicaciones del mundo real?**

Spring Data REST no se recomienda en aplicaciones del mundo real, ya que está exponiendo las entidades de su base de datos directamente como servicios REST. Al diseñar servicios RESTful, las dos cosas más importantes que consideramos son el modelo de dominio y los consumidores. Pero, al usar Spring Data REST, ninguno de estos parámetros se considera. Las entidades están directamente expuestas. Entonces, solo diría que puede usar Spring Data REST para la evolución inicial del proyecto.

## **P44. ¿Cuál es el error que ves si H2 no está en el classpath?**

Si H2 no está presente en el classpath, verá el siguiente error:

Cannot determine embedded database driver class for database type NONE

Para resolver este error, agregue H2 al archivo pom.xml y reinicie su servidor.  
Se puede agregar el siguiente fragmento de código para agregar la dependencia:

<dependencia>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<ámbito>tiempo de ejecución</ámbito>

</dependencia>

# **P45. ¿Cuál es la forma de usar perfiles para configurar la configuración específica del entorno con Spring Boot?**

Dado que es un hecho conocido que un perfil no es más que una clave para identificar un entorno, consideremos los siguientes dos perfiles en el ejemplo:

* desarrollador
* pinchar
* Considere las siguientes propiedades presentes en el archivo de propiedades de la aplicación:

ejemplo.número: 100

ejemplo.valor: verdadero

ejemplo.mensaje: Mensaje dinámico

Ahora, digamos que desea personalizar application.properties para el perfil de desarrollo, luego debe crear un archivo con el nombre application-dev.properties y anular las propiedades que desea personalizar. Puedes mencionar el siguiente código:

ejemplo.mensaje: mensaje dinámico en desarrollo

Del mismo modo, si desea personalizar application.properties para el perfil de producción, puede mencionar el siguiente fragmento de código:

ejemplo.mensaje: Mensaje dinámico en Prod

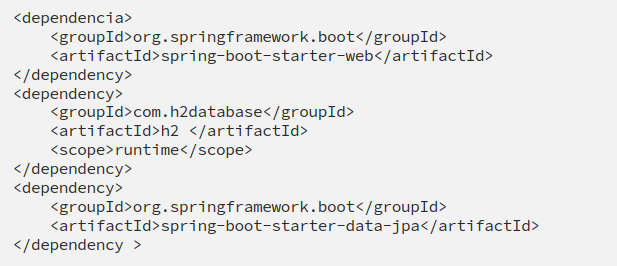
Una vez que haya terminado con la configuración específica del perfil, debe establecer el perfil activo en un entorno. Para hacer eso, puedes

* Usar -Dspring.profiles.active=proden argumentos
* Usar spring.profiles.active=proden el archivo application.properties

## **P46. ¿Mencione las dependencias necesarias para iniciar una aplicación JPA y conectarse a la base de datos en memoria H2 con Spring Boot?**

Las dependencias son necesarias para iniciar una aplicación JPA y conectarse a la base de datos H2 en memoria con Spring Boot

* arrancador web
* h2
* arrancador de datos JPA
* Para incluir las dependencias, consulte el siguiente código:



## **P47. ¿Qué entiendes por Spring Boot que admite la unión relajada?**

El enlace relajado es una forma en la que el nombre de la propiedad no necesita coincidir con la clave de la propiedad del entorno. En Spring Boot, el enlace relajado es aplicable al enlace seguro de tipo de las propiedades de configuración. Por ejemplo, si una propiedad en una clase de bean con la anotación @ConfigurationPropertie se usa sampleProp, entonces se puede vincular a cualquiera de las siguientes propiedades de entorno:

* muestraProp
* muestra-Prop
* muestra\_Prop
* MUESTRA\_PROP

## **P48. ¿Dónde se especifica la información de conexión de la base de datos y cómo se conecta automáticamente a H2?**

Es debido a la configuración automática de Spring Boot que configura las dependencias de la aplicación. Por lo tanto, la propiedad de configuración automática realiza la información de conexión de la base de datos y la conexión automática de la base de datos a H2.

## **P49. ¿Cuál es el nombre de la base de datos H2 predeterminada configurada por Spring Boot?**

El nombre de la base de datos H2 predeterminada es testdb. Consulte a continuación:

spring.datasource.name=testdb # Nombre de la fuente de datos.

Nota: Por si acaso, si está utilizando una base de datos en memoria H2, ese es exactamente el nombre de Spring Boot que se utiliza para configurar su base de datos H2.

## **Q50. ¿Crees que puedes usar jetty en lugar de tomcat en spring-boot-starter-web?**

Sí, podemos usar jetty en lugar de tomcat en spring-boot-starter-web, eliminando la dependencia existente e incluyendo lo siguiente:

Con esto, llegamos al final de este artículo sobre las preguntas de la entrevista de Spring Boot.

