Problemas de Inicio de Aplicación en WebSphere Liberty Profile

Parece que después de degradar su aplicación Spring Boot a la versión 2.7.18 y desplegarla en WebSphere Liberty Profile (WLP), está encontrando un problema donde la aplicación falla al iniciar dentro del período de tiempo de espera predeterminado, como se indica en el mensaje en `console.log`: *"la aplicación myapp no ha iniciado en 30.xx segundos"*. Esto sugiere que la aplicación está tardando demasiado en inicializarse o fallando por completo. A continuación, le guiaré a través de un proceso paso a paso para diagnosticar y solucionar este problema.

Paso 1: Investigar los Registros del Servidor en Busca de Pistas

Lo primero que debe hacer es examinar los registros del servidor en más detalle para identificar la causa raíz del fallo en el inicio. El archivo `console.log` proporciona el mensaje de tiempo de espera, pero puede que no contenga toda la historia. Verifique los siguientes archivos de registro en el directorio de registros de su servidor WLP (`/opt/ibm/wlp/usr/servers/myServer/logs/`):

- `messages.log`: Este archivo a menudo contiene mensajes de ERROR o WARNING que pueden identificar problemas como dependencias faltantes, errores de configuración o excepciones durante el inicio.
- `trace.log`: Si se ha habilitado el rastreo detallado, este archivo puede proporcionar más contexto sobre lo que está sucediendo durante el despliegue.

Busque: - Trazas de pila o excepciones (por ejemplo, `ClassNotFoundException`, `NoSuchBeanDefinitionException`). - Mensajes sobre recursos faltantes o bibliotecas incompatibles. - Indicaciones de que el contexto de la aplicación falló en inicializarse.

Si no ve suficiente detalle, puede aumentar el nivel de registro en WLP modificando el archivo `server.xml`. Agregue o actualice el elemento `<logging>` de la siguiente manera:

```
<logging traceSpecification="*=info:com.ibm.ws.webcontainer*=all" />
```

Reinicie el servidor después de realizar este cambio, vuelva a desplegar su aplicación y verifique los registros nuevamente en busca de más información.

Paso 2: Verificar el Inicio de la Aplicación con Registro

Dado que esta es una aplicación Spring Boot, el problema podría estar relacionado con el contexto de la aplicación que falla en inicializarse. Para determinar hasta dónde llega el proceso de inicio, agregue una simple declaración de registro a su clase principal de la aplicación utilizando un método `@PostConstruct`. Aquí hay un ejemplo:

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import javax.annotation.PostConstruct;

@SpringBootApplication
public class DemoApplication extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(DemoApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @PostConstruct
    public void init() {
        System.out.println("Contexto de la aplicación inicializado");
    }
}
```

- Reconstruya su aplicación (`mvn clean package`).
- Vuelva a desplegar el archivo WAR en el directorio `dropins` de WLP.
- Verifique `console.log` en busca del mensaje `"Contexto de la aplicación inicializado"`.

Si este mensaje aparece, el contexto de la aplicación se está cargando correctamente y el problema podría estar relacionado con componentes web o la inicialización de servlets. Si no aparece, el problema ocurre antes durante la inicialización del contexto.

Paso 3: Habilitar el Registro de Depuración en Spring Boot

Para obtener más visibilidad en el proceso de inicio de Spring Boot, habilite el registro de depuración agregando un archivo de configuración. Cree o edite `src/main/resources/application.properties` con lo siguiente:

```
debug=true
```

- Reconstruya y vuelva a desplegar la aplicación.
- Verifique `console.log` (u otros registros) en busca de salida de depuración detallada de Spring Boot.

Esto registrará información sobre la creación de beans, la auto-configuración y cualquier error que ocurra durante el inicio. Busque pistas sobre qué podría estar colgando o fallando.

Paso 4: Verificar el Archivo WAR y la Configuración de Dependencias

Dado que está desplegando en WLP, que proporciona su propio contenedor Servlet, asegúrese de que su archivo WAR esté correctamente configurado para un servidor externo:

- **Empaquetado WAR**: En su `pom.xml`, confirme que el empaquetado está configurado como `war`:

```
<packaging>war</packaging>
```

- **Tomcat como Proporcionado**: Asegúrese de que el Tomcat incrustado esté excluido del archivo WAR, ya que WLP proporcionará el contenedor Servlet. Verifique su `pom.xml` para:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
```

- **Compatibilidad de API Servlet**: Spring Boot 2.7.18 usa `javax.servlet:javax.servlet-api:4.0.1`, que es compatible con la característica `javaee-8.0` de WLP (Servlet 4.0). Para confirmar que no hay dependencias en conflicto, ejecute:

```
mvn dependency:tree
```

Busque cualquier versión inesperada de la API Servlet (por ejemplo, `jakarta.servlet-api`, que se usa en Spring Boot 3.x e incompatible con `javaee-8.0`).

Si sospecha de problemas de dependencia, descomprima el archivo WAR e inspeccione `WEB-INF/lib` para asegurarse de que no se incluyan JARs relacionados con Servlet inesperados.

Paso 5: Probar Localmente para Aislar el Problema

Para determinar si el problema es específico de WLP o de la aplicación misma, pruebe la aplicación localmente utilizando el Tomcat incrustado:

```
mvn spring-boot:run
```

Si se inicia correctamente y puede acceder a sus puntos finales (por ejemplo, un simple controlador REST `"Hello World!"`), el problema es probable que esté relacionado con el despliegue en WLP en lugar del código de la aplicación.

Paso 6: Ajustar el Tiempo de Espera de Inicio de WLP (Solución Temporal)

Si los registros sugieren que la aplicación está iniciando pero tarda más de 30 segundos, puede aumentar el tiempo de espera de inicio en el `server.xml` de WLP:

```
<applicationMonitor startTimeout="60s" />
```

- Vuelva a desplegar la aplicación y monitoree los registros.
- Si se inicia después del tiempo de espera extendido, esto confirma un proceso de inicio lento y debería optimizar la aplicación (por ejemplo, reducir el escaneo de componentes o tareas de inicialización).

Sin embargo, esto es una solución temporal: idealmente, una aplicación simple debería iniciar en 30 segundos, por lo que continúe investigando la causa raíz.

Paso 7: Simplificar y Comparar con un Nuevo Proyecto

Si el problema persiste, cree un proyecto Spring Boot 2.7.18 mínimo para probar el despliegue en WLP: 1. Use Spring Initializr con: - Spring Boot 2.7.18 - Java (coincidiendo con la versión de WLP, por ejemplo, 8 o 11) - Dependencia: Spring Web 2. Agregue un controlador REST básico:

```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
```

```
public class HelloController {
    @GetMapping("/")
    public String hello() {
        return "Hello World!";
    }
}
```

3. Configúrelo para el despliegue WAR (extienda `SpringBootServletInitializer` como se mostró anteriormente).
4. Construya el archivo WAR (`mvn clean package`) y despliéguelo en el directorio `dropins` de WLP.

Si este nuevo proyecto se inicia correctamente, compare su `pom.xml`, clase principal y configuración con su proyecto original para identificar diferencias que causen el fallo.

Paso 8: Solución de Problemas Adicionales de WLP

Si ninguno de los pasos anteriores resuelve el problema, considere estas verificaciones específicas de WLP: - **Limpiar el Directorio de Trabajo**: Elimine los datos en caché que podrían interferir con el despliegue: `bash rm -rf /opt/ibm/wlp/usr/servers/myServer/workarea` Luego, reinicie el servidor y vuelva a desplegar. - **Verificar la Versión de WLP**: Ejecute `/opt/ibm/wlp/bin/productInfo version` para confirmar que su versión de WLP admite despliegues de Spring Boot 2.7.x con `javaee-8.0`. - **Versión de Java**: Asegúrese de que la versión de Java utilizada por WLP (verifique con `java -version` en el entorno de WLP) coincida con una versión compatible con Spring Boot 2.7.18 (Java 8, 11 o 17).

Solución Recomendada

Basado en la información proporcionada, el problema más probable es un problema durante la inicialización del contexto de la aplicación o una discrepancia sutil de configuración con WLP. Aquí hay cómo solucionarlo:

1. **Comience con el Registro**:
    - Agregue el registro `@PostConstruct` (Paso 2) y habilite `debug=true` (Paso 3).
    - Vuelva a desplegar y verifique `console.log` y `messages.log` en busca de errores o donde el inicio se detiene.
2. **Valide la Configuración**:
    - Confirme la configuración del archivo WAR y las dependencias (Paso 4).
    - Pruebe localmente (Paso 5) para descartar problemas a nivel de aplicación.

3. **Iterar y Simplificar**:
    - Si es necesario, pruebe un proyecto mínimo (Paso 7) para aislar el problema.

Siguiendo estos pasos, debería poder identificar la causa: ya sea un conflicto de dependencias, un componente que tarda en iniciarse o un problema de configuración de WLP, y resolverlo en consecuencia. ¡Si aún encuentra dificultades, comparta mensajes de error específicos de los registros para obtener más asistencia!