Spring Boot 2.7.18 im WebSphere Liberty Profile

Wenn Sie Spring Boot 2.7.18 verwenden, um eine Anwendung auf WebSphere Liberty Profile (WLP) zu deployen, wie in der bereitgestellten Anleitung beschrieben, ist die gute Nachricht, dass keine Änderungen an der `DemoApplication`-Klasse oder der Verwendung von `SpringBootServletInitializer` erforderlich sind. Spring Boot 2.7.18 ist ein Patch-Release innerhalb der 2.7.x-Serie, was bedeutet, dass es Fehlerbehebungen und kleinere Verbesserungen enthält, aber keine Änderungen, die die Standardkonfiguration für die WAR-Deployment, wie in der Anleitung beschrieben, beeinflussen würden. Im Folgenden werde ich erklären, warum dies der Fall ist, und bestätigen, dass die Einrichtung weiterhin vollständig kompatibel ist.

Verständnis der Einrichtung

Die Anleitung konfiguriert eine Spring Boot-Anwendung, die als WAR-Datei auf WLP deployt wird, indem:

1. **Erweiterung von `SpringBootServletInitializer`:**
   - Die `DemoApplication`-Klasse erweitert `SpringBootServletInitializer`, um die Bereitstellung auf einem externen Servlet-Container wie WLP zu ermöglichen. Diese Klasse ist notwendig, da Spring Boot standardmäßig als eigenständige JAR-Datei mit einem eingebetteten Server läuft, während die Bereitstellung auf WLP eine Integration mit dessen Servlet-Container erfordert.
   - Die `configure`-Methode wird überschrieben, um die Anwendungsquellen zu spezifizieren und sicherzustellen, dass die Anwendung im externen Container korrekt initialisiert wird.

   Hier ist der Code aus der Anleitung:

```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;

@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);
    }
}
```

2. **Verpacken als WAR-Datei:**
   - Die `pom.xml` spezifiziert `<packaging>war</packaging>` und markiert die `spring-boot-starter-tomcat`-Abhängigkeit als `<scope>provided</scope>`, um den eingebetteten Tomcat-Server auszuschließen und stattdessen auf den Servlet-Container von WLP zu vertrauen.
3. **Bereitstellen auf WLP:**
   - Die WAR-Datei wird im `dropins`-Verzeichnis von WLP platziert, um eine automatische Bereitstellung zu ermöglichen, und das `javaee-8.0`-Feature von WLP bietet Unterstützung für Servlet 4.0, was mit den Anforderungen von Spring Boot kompatibel ist.

Warum keine Änderungen mit Spring Boot 2.7.18 erforderlich sind

Spring Boot 2.7.18 ist Teil der 2.7.x-Serie, und bedeutende Änderungen an den Bereitstellungsmechanismen oder APIs treten normalerweise zwischen den Hauptversionen (z.B. 2.x zu 3.x) und nicht innerhalb von Minor- oder Patch-Releases auf. Hier ist eine detaillierte Analyse:

1. Kompatibilität mit `SpringBootServletInitializer`

- **Zweck:** `SpringBootServletInitializer` bleibt die Standardmethode zur Konfiguration einer Spring Boot-Anwendung für die WAR-Bereitstellung in der 2.x-Serie. Es integriert sich in den externen Servlet-Container, indem es einen Haken bereitstellt, um den Anwendungskontext einzurichten.
- **Stabilität in 2.7.18:** Es gibt keine Abwertungen oder Ersatzlösungen für `SpringBootServletInitializer` in Spring Boot 2.7.18. Wichtige Änderungen, wie der Wechsel zu Jakarta EE (Ersetzen der Java EE-APIs), treten in Spring Boot 3.x auf, was auch Java 17 erfordert. Da 2.7.18 innerhalb der 2.x-Serie bleibt und Java EE verwendet, bleibt die bestehende Implementierung in `DemoApplication` gültig und unverändert.

2. Kompatibilität des Servlet-Containers

- **Spring Boot-Anforderungen:** Spring Boot 2.7.x erfordert Servlet 3.1 oder höher. Die Anleitung verwendet WLP mit dem `javaee-8.0`-Feature, das Servlet 4.0 enthält—eine noch neuere Spezifikation. Dies stellt die volle Kompatibilität sicher.
- **Keine Änderungen in 2.7.18:** Patch-Releases wie 2.7.18 ändern die Servlet-Kompatibilität nicht oder führen neue Anforderungen ein, die die Interaktion von `SpringBootServletInitializer` mit WLP beeinflussen würden.

3. Abhängigkeits- und Verpackungskonfiguration

- **Tomcat als `provided`:** Die Anleitung setzt `spring-boot-starter-tomcat` in `pom.xml` korrekt auf `<scope>provided</scope>`, wodurch der eingebettete Tomcat aus der WAR-Datei ausgeschlossen wird. Dies ist eine Standardpraxis für WAR-Bereitstellungen auf externen Containern und bleibt in 2.7.18 unverändert.
- **Maven-Konfiguration:** Der Verpackungstyp (`war`) und die Abhängigkeitsaufstellung sind mit den Spring Boot 2.7.x-Konventionen konsistent, und es sind keine Updates spezifisch für 2.7.18 erforderlich.

4. WLP-Bereitstellungsspezifika

- **Dropins-Verzeichnis:** Der `dropins`-Bereitstellungsmechanismus von WLP wird durch die Spring Boot-Version nicht beeinflusst, da er auf der WAR-Dateistruktur und den Servlet-Spezifikationen basiert, die beide kompatibel bleiben.
- **Kontextwurzel und Port:** Die Kontextwurzel (z.B. `/myapp`) und der Standardport (z.B. `9080`) werden durch WLP und den Namen der WAR-Datei bestimmt, nicht durch die Spring Boot-Version, sodass sie gleich bleiben.

5. Potenzielle neue Funktionen oder Abwertungen

- **Betrachtung der Versionshinweise:** Die Versionshinweise von Spring Boot 2.7.x (einschließlich 2.7.18) decken normalerweise Abhängigkeitsaktualisierungen, Sicherheitsfixes und kleinere Verbesserungen ab, die jedoch die WAR-Bereitstellung oder `SpringBootServletInitializer` nicht beeinflussen. Zum Beispiel wurden in 2.7.18 keine neuen Annotationen oder Konfigurationsmethoden eingeführt, die die bestehende Vorgehensweise ersetzen.
- **Kontrast zu 3.x:** Im Gegensatz dazu übernimmt Spring Boot 3.x Jakarta EE und könnte Anpassungen erfordern (z.B. unterschiedliche Basisklassen oder aktualisierte Servlet-APIs), aber 2.7.18 überbrückt diese Änderungen nicht.

Fazit

Für Spring Boot 2.7.18 sind keine Änderungen an der `DemoApplication`-Klasse oder der Verwendung von `SpringBootServletInitializer`, wie in der Anleitung gezeigt, erforderlich. Die Konfiguration ist vollständig kompatibel mit:

- WAR-Verpackung und Bereitstellung auf WLP.
- Das `javaee-8.0`-Feature in WLP (Servlet 4.0).
- Die Servlet- und Bereitstellungsanforderungen von Spring Boot 2.7.x.

Sie können mit den exakten Schritten und dem Code aus der Anleitung—unter Verwendung von Spring Boot 2.7.18 anstelle einer generischen 2.7.x-Version—fortfahren und erwarten, dass die Anwendung erfolgreich auf WLP deployt wird und “Hello World!”unter `http://localhost:9080/myapp/` zurückgibt. Keine zusätzlichen Konfigurationen, Eigenschaftsänderungen oder Codeanpassungen sind erforderlich.