Spring Boot 2.7.18 in WebSphere Liberty Profile

When using Spring Boot 2.7.18 for deploying an application to WebSphere Liberty Profile (WLP) as described in the provided guide, the good news is that no changes are required to the `DemoApplication` class or the use of `SpringBootServletInitializer`. Spring Boot 2.7.18 is a patch release within the 2.7.x series, which means it includes bug fixes and minor improvements but does not introduce breaking changes that would affect the standard configuration for WAR deployment outlined in the guide. Below, I'll explain why this is the case and confirm that the setup remains fully compatible.

Understanding the Setup

The guide configures a Spring Boot application to be deployed as a WAR file on WLP by:

1. **Extending `SpringBootServletInitializer`:**
   - The `DemoApplication` class extends `SpringBootServletInitializer` to enable deployment to an external servlet container like WLP. This class is necessary because, while Spring Boot defaults to running as a standalone JAR with an embedded server, deploying to WLP requires integration with its servlet container.
   - The `configure` method is overridden to specify the application sources, ensuring the application bootstraps correctly in the external container.

   Here's the code from the guide:

```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. **Packaging as a WAR File:**

   - The `pom.xml` specifies `<packaging>war</packaging>` and marks the `spring-boot-starter-tomcat` dependency as `<scope>provided</scope>` to exclude the embedded Tomcat server, relying instead on WLP's servlet container.

3. **Deploying to WLP:**

   - The WAR file is placed in WLP's `dropins` directory for automatic deployment, and WLP's `javaee-8.0` feature provides Servlet 4.0 support, which is compatible with Spring Boot's requirements.

Why No Changes Are Needed with Spring Boot 2.7.18

Spring Boot 2.7.18 is part of the 2.7.x series, and significant changes to deployment mechanisms or APIs typically occur between major versions (e.g., 2.x to 3.x), not within minor or patch releases. Here's a detailed analysis:

1. Compatibility with `SpringBootServletInitializer`

- **Purpose:** `SpringBootServletInitializer` remains the standard way to configure a Spring Boot application for WAR deployment in the 2.x series. It integrates with the external servlet container by providing a hook to set up the application context.
- **Stability in 2.7.18:** There are no deprecations or replacements for `SpringBootServletInitializer` in Spring Boot 2.7.18. Major changes, such as the shift to Jakarta EE (replacing Java EE APIs), occur in Spring Boot 3.x, which also requires Java 17. Since 2.7.18 stays within the 2.x series and uses Java EE, the existing implementation in `DemoApplication` remains valid and unchanged.

2. Servlet Container Compatibility

- **Spring Boot Requirements:** Spring Boot 2.7.x requires Servlet 3.1 or higher. The guide uses WLP with the `javaee-8.0` feature, which includes Servlet 4.0—an even newer specification. This ensures full compatibility.
- **No Changes in 2.7.18:** Patch releases like 2.7.18 do not alter servlet compatibility or introduce new requirements that would affect how `SpringBootServletInitializer` interacts with WLP.

3. Dependency and Packaging Configuration

- **Tomcat as `provided`:** The guide correctly sets `spring-boot-starter-tomcat` to `<scope>provided</scope>` in `pom.xml`, ensuring the embedded Tomcat is excluded from the WAR file. This is a standard practice for WAR deployments to external containers and remains unchanged in 2.7.18.
- **Maven Configuration:** The packaging type (`war`) and dependency setup are consistent with Spring Boot 2.7.x conventions, and no updates specific to 2.7.18 are needed.

4. WLP Deployment Specifics

- **Dropins Directory:** WLP's `dropins` deployment mechanism is unaffected by the Spring Boot version, as it relies on the WAR file structure and servlet specifications, both of which remain compatible.
- **Context Root and Port:** The context root (e.g., `/myapp`) and default port (e.g., `9080`) are determined by WLP and the WAR file name, not Spring Boot's version, so they stay the same.

5. Potential New Features or Deprecations

- **Release Notes Consideration:** Spring Boot 2.7.x release notes (including 2.7.18) typically cover dependency updates, security fixes, and minor enhancements, but none impact WAR deployment or `SpringBootServletInitializer`. For example, no new annotations or configuration methods were introduced in 2.7.18 to replace the existing approach.
- **Contrast with 3.x:** In contrast, Spring Boot 3.x adopts Jakarta EE and might require adjustments (e.g., different base classes or updated servlet APIs), but 2.7.18 does not bridge to those changes.

Conclusion

For Spring Boot 2.7.18, the `DemoApplication` class and the use of `SpringBootServletInitializer` as shown in the guide do not require any modifications. The configuration is fully compatible with:

- WAR packaging and deployment to WLP.
- The `javaee-8.0` feature in WLP (Servlet 4.0).
- Spring Boot 2.7.x's servlet and deployment requirements.

You can proceed with the exact steps and code from the guide—using Spring Boot 2.7.18 instead of a generic 2.7.x version—and expect the application to deploy and run successfully on WLP, returning "Hello World!" at `http://localhost:9080/myapp/`. No additional configurations, property changes, or code adjustments are necessary.