應用程序啟動問題於 WebSphere Liberty Profile

看來，在將您的 Spring Boot 應用程式降級到版本 2.7.18 並部署到 WebSphere Liberty Profile (WLP) 後，您遇到了應用程式在預設超時期間內未能啟動的問題，如 `console.log` 中的訊息所示：「the application myapp has not started in 30.xx seconds」。這表明應用程式可能初始化時間過長或根本無法啟動。以下是逐步診斷和修復此問題的指南。

第 1 步：調查伺服器日誌以尋找線索

首先，詳細檢查伺服器日誌以識別啟動失敗的根本原因。`console.log` 文件提供了超時訊息，但可能不包含完整的故事。檢查 WLP 伺服器日誌目錄中的以下日誌文件 (`/opt/ibm/wlp/usr/servers/myServer/logs/`):

- `messages.log`：這個文件通常包含 ERROR 或 WARNING 消息，可以指向問題，例如缺少依賴項、配置錯誤或啟動期間的例外。
- `trace.log`：如果啟用了詳細追蹤，這個文件可能會提供有關部署過程中的更多上下文。

查找：- 堆疊跡或例外（例如 `ClassNotFoundException`、`NoSuchBeanDefinitionException`）。- 關於缺少資源或不兼容庫的消息。- 應用程式上下文未能初始化的指示。

如果沒有看到足夠的詳細信息，可以通過修改 `server.xml` 文件來增加 WLP 中的日誌級別。添加或更新 `<logging>` 元素如下：

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

在進行此更改後重新啟動伺服器，重新部署應用程式，並再次檢查日誌以獲取更多信息。

第 2 步：使用日誌驗證應用程式啟動

由於這是一個 Spring Boot 應用程式，問題可能與應用程式上下文未能初始化有關。為了確定啟動過程進展到哪一步，在主應用程式類中添加一個簡單的日誌語句，使用 `@PostConstruct` 方法。以下是一個範例：

```
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("Application context initialized");
    }
}
```

- 重新構建應用程式 (`mvn clean package`)。
- 將 WAR 文件重新部署到 WLP 的 `dropins` 目錄。
- 檢查 `console.log` 中的訊息 `"Application context initialized"`。

如果出現此訊息，則應用程式上下文正在成功加載，問題可能與 Web 組件或 Servlet 初始化有關。如果沒有出現，則問題發生在上下文初始化過程的較早階段。

第 3 步：在 Spring Boot 中啟用調試日誌

為了更好地了解 Spring Boot 的啟動過程，通過添加配置文件來啟用調試日誌。創建或編輯 `src/main/resources/applicat` 文件，添加以下內容：

```
debug=true
```

- 重新構建和重新部署應用程式。
- 檢查 `console.log`（或其他日誌）以獲取 Spring Boot 的詳細調試輸出。

這將記錄有關 Bean 創建、自動配置和啟動期間發生的任何錯誤的信息。查找可能導致啟動過程卡住或失敗的線索。

第 4 步：驗證 WAR 文件和依賴配置

由於您正在將應用程式部署到 WLP，它提供自己的 Servlet 容器，請確保您的 WAR 文件正確配置為外部伺服器：

- **WAR 打包**：在 `pom.xml` 中確認打包設置為 `war`：

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

- **Tomcat 作為提供**：確保嵌入式 Tomcat 從 WAR 文件中排除，因為 WLP 將提供 Servlet 容器。檢查您的 `pom.xml` 中是否有以下內容：

```
<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>
```

- **Servlet API 兼容性**：Spring Boot 2.7.18 使用 `javax.servlet:javax.servlet-api:4.0.1`，這與 WLP 的 `javaee-8.0` 功能（Servlet 4.0）兼容。為了確認沒有衝突的依賴項，運行：

```
mvn dependency:tree
```

查找任何意外的 Servlet API 版本（例如 `jakarta.servlet-api`，這在 Spring Boot 3.x 中使用，與 `javaee-8.0` 不兼容）。

如果懷疑依賴項問題，解壓縮 WAR 文件並檢查 `WEB-INF/lib`，以確保沒有包含意外的 Servlet 相關 JAR。

第 5 步：本地測試以隔離問題

為了確定問題是否特定於 WLP 或應用程式本身，使用嵌入式 Tomcat 本地測試應用程式：

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

如果成功啟動並且可以訪問您的端點（例如，簡單的 `"Hello World!"` REST 控制器），則問題可能與 WLP 部署有關，而不是應用程式代碼。

第 6 步：調整 WLP 啟動超時（臨時解決方案）

如果日誌顯示應用程式正在啟動但需要超過 30 秒，可以在 WLP 的 `server.xml` 中增加啟動超時：

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

- 重新部署應用程式並監控日誌。
- 如果在延長的超時後啟動，這確認了慢啟動過程，應該優化應用程式（例如，減少組件掃描或初始化任務）。

然而，這是一個臨時解決方案——理想情況下，簡單的應用程式應該在 30 秒內啟動，因此應繼續調查根本原因。

第 7 步：簡化並與新項目進行比較

如果問題持續存在，創建一個最小的 Spring Boot 2.7.18 項目來測試在 WLP 上的部署：1. 使用 Spring Initializr 並選擇：- Spring Boot 2.7.18 - Java（與 WLP 版本匹配，例如 8 或 11）- 依賴項：Spring Web 2. 添加基本的 REST 控制器：

```
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. 將其配置為 WAR 部署（如上所述擴展 `SpringBootServletInitializer`）。
4. 構建 WAR 文件 (`mvn clean package`) 並將其部署到 WLP 的 `dropins` 目錄。

如果此新項目成功啟動，請將其 `pom.xml`、主類和配置與原始項目進行比較，以識別導致失敗的差異。

第 8 步：額外的 WLP 疑難排解

如果以上方法都無法解決問題，請考慮這些特定於 WLP 的檢查：- **清理工作目錄**：刪除可能干擾部署的快取數據：`bash  rm -rf /opt/ibm/wlp/usr/servers/myServer/workarea` 然後重新啟動伺服器並重新部署。- **檢查 WLP 版本**：運行 `/opt/ibm/wlp/bin/productInfo version` 以確認您的 WLP 版本支持 Spring Boot 2.7.x 部署，使用 `javaee-8.0`。- **Java 版本**：確保 WLP 使用的 Java 版本（在 WLP 環境中使用 `java -version` 進行檢查）與 Spring Boot 2.7.18 支持的版本匹配（Java 8、11 或 17）。

推薦修復

根據提供的信息，最有可能的問題是應用程式上下文初始化期間的問題或與 WLP 的微妙配置不匹配。以下是如何修復它的方法：

1. **從日誌開始**：
   - 添加 `@PostConstruct` 日誌（第 2 步）並啟用 `debug=true`（第 3 步）。
   - 重新部署並檢查 `console.log` 和 `messages.log` 中的錯誤或啟動卡住的位置。
2. **驗證配置**：
   - 確認 WAR 文件設置和依賴項（第 4 步）。
   - 本地測試（第 5 步）以排除應用程式級別的問題。
3. **迭代和簡化**：
   - 如果需要，測試最小項目（第 7 步）以隔離問題。

通過遵循這些步驟，您應該能夠找到問題的原因——無論是依賴項衝突、慢啟動組件還是 WLP 配置問題——並相應地解決它。如果您仍然遇到困難，請分享日誌中的具體錯誤訊息以獲取進一步的幫助！