部署 Spring Boot 到 WebSphere Liberty Profile

這是一個在 WebSphere Liberty Profile (WLP) 上運行簡單 Spring Boot 應用程序的完整指南。本示例創建了一個具有基本 REST 端點的 Spring Boot 應用程序，該端點返回“Hello World!”，並將其部署到 WLP 使用 `dropins` 目錄。我們假設您已經設置了 WLP 並啟用了 `javaee-8.0` 功能，因為這提供了必要的 Servlet 支持。

逐步指南

1. 創建 Spring Boot 應用程序 從創建一個新的 Spring Boot 專案開始，使用 Spring Initializr（或您喜歡的方法）。我們將使用 Maven、Java 和 Spring Web 依賴項來構建一個簡單的 Web 應用程序。

- **生成專案：**
  - 訪問 Spring Initializr。
  - 配置以下內容：
    - **專案：** Maven
    - **語言：** Java
    - **Spring Boot 版本：** 2.7.x（或最新穩定版本）
    - **組：** `com.example`
    - **工件：** `demo`
    - **依賴項：** Spring Web
  - 點擊“生成”下載專案 ZIP，然後解壓縮並在 IDE 中打開。
- **添加簡單的 REST 控制器：**在 `src/main/java/com/example/demo` 中創建一個名為 `HelloController.java` 的文件，內容如下：

```
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!";
    }
}
```

這將在根路徑（`/`）創建一個 REST 端點，返回“Hello World!”作為純文本。

2. 配置應用程序以進行 WAR 部署 默認情況下，Spring Boot 將應用程序打包為包含嵌入式伺服器（例如 Tomcat）的 JAR 文件。要在 WLP 上部署，我們需要將其打包為 WAR 文件並配置其與 WLP 的 Servlet 容器一起工作。

- **修改主應用程序類：**編輯 `src/main/java/com/example/demo/DemoApplication.java` 以擴展 `SpringBootServletInitial` 這允許應用程序在外部 Servlet 容器（如 WLP）中運行：

```
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);
    }
}
```

- **更新 `pom.xml` 以進行 WAR 打包：**打開 `pom.xml` 並進行以下更改：
  - 通過在頂部添加以下行（`<modelVersion>` 以下）將打包設置為 WAR：

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

  - 將嵌入式 Tomcat 依賴項標記為 `provided`，以便它不包含在 WAR 中（WLP 提供自己的 Servlet 容器）。修改 `spring-boot-starter-web` 依賴項（其中包括 Tomcat）如下：

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

    在其下方添加：

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

```
        <scope>provided</scope>
    </dependency>
```

您的 `pom.xml` 依賴項部分現在應該看起來像這樣：

```
<dependencies>
  <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>
  <!-- 其他依賴項如 spring-boot-starter-test 可能保持不變 -->
</dependencies>
```

3. 打包 WAR 文件 使用 Maven 將應用程序編譯並打包為 WAR 文件。

- **運行構建命令：**從專案根目錄（`pom.xml` 所在的位置）執行：

  ```
  mvn clean package
  ```

  這將在 `target` 目錄中生成 WAR 文件，例如 `target/demo-0.0.1-SNAPSHOT.war`。

- **重命名 WAR 文件（可選）：**為了更簡潔的 URL，將 WAR 文件重命名為 `myapp.war`：

  ```
  mv target/demo-0.0.1-SNAPSHOT.war target/myapp.war
  ```

  這將上下文根簡化為 `/myapp` 而不是 `/demo-0.0.1-SNAPSHOT`。

4. 在 WLP 上部署 WAR 文件 使用 `dropins` 目錄將 WAR 文件部署到 WLP，這將啟用自動部署。

- **定位 `dropins` 目錄：**找到 WLP 伺服器的 `dropins` 目錄。如果 WLP 安裝在 `/opt/ibm/wlp` 並您的伺服器名為 `myServer`，則路徑為：

  ```
  /opt/ibm/wlp/usr/servers/myServer/dropins
  ```

- **複製 WAR 文件：**將 WAR 文件移動到 `dropins` 目錄：

  ```
  cp target/myapp.war /opt/ibm/wlp/usr/servers/myServer/dropins/
  ```

- **啟動伺服器（如果未運行）：**如果 WLP 未運行，請啟動它：

  ```
  /opt/ibm/wlp/bin/server start myServer
  ```

如果它已運行，它將自動檢測並部署 WAR 文件。

- **驗證部署：**在伺服器日誌或控制台中查找類似以下的消息：

  ```
  [AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/myapp/
  ```

  - 日誌在 `/opt/ibm/wlp/usr/servers/myServer/logs/console.log`（後台模式）或顯示在終端（前台模式與 `./server run myServer`）。

5. 訪問應用程序 在瀏覽器中測試部署的 Spring Boot 應用程序。

- **打開瀏覽器：**導航到：

  ```
  http://localhost:9080/myapp/
  ```

  - `9080` 是 WLP 的默認 HTTP 端口。
  - `/myapp` 是來自 WAR 文件名的上下文根。
  - `/` 符合控制器中的 `@GetMapping("/")`。

- **預期結果：**您應該看到：

  ```
  Hello World!
  ```

  作為純文本顯示。

注意事項

- **上下文根：**上下文根（`/myapp`）來自 WAR 文件名。根據需要通過重命名 WAR 文件進行調整。
- **端口號：** WLP 默認使用 `9080` 作為 HTTP 端口。如果您的伺服器使用不同的端口，請相應更新 URL。
- **Java 版本：**確保 WLP 和您的構建環境使用兼容的 Java 版本（例如，Java 8 或 11 適用於 Spring Boot 2.7.x）。
- **WLP 功能：** `javaee-8.0` 功能包括 Servlet 4.0 支持，這與 Spring Boot 2.7.x 的要求（Servlet 3.1+）兼容。
- **故障排除：**
  - 如果頁面未加載，請檢查 `/opt/ibm/wlp/usr/servers/myServer/logs/messages.log` 中的錯誤。
  - 驗證 WAR 文件在 `dropins` 中並伺服器正在運行。

摘要

本指南演示了如何：1. 創建一個具有 REST 端點的 Spring Boot 應用程序。2. 通過擴展 `SpringBootServletInitializer` 並將 Tomcat 設置為 `provided` 來配置 WAR 部署。3. 將 WAR 文件構建並部署到 WLP 的 `dropins` 目錄。4. 在 `http://localhost:9080/myapp/` 訪問“Hello World!”輸出。

這種方法利用了 WLP 的自動部署功能和 Spring Boot 的靈活性，提供了一個簡單但功能強大的 Web 應用程序在 WebSphere Liberty Profile 上。