Java Servlet 應用程式於 WebSphere Liberty Profile

這是一個在 WebSphere Liberty Profile (WLP) 上運行非常簡單的 Java Servlet 應用程式的完整指南。這個範例創建了一個基本的 Servlet，返回“Hello World!”，並使用 `dropins` 目錄將其部署到 WLP。我們假設您已經啟用了 `servlet-4.0` 功能的 WLP，因為這提供了必要的 Servlet 支持。這個指南避免使用 Maven，並提供了所有所需的代碼和部署步驟。

分步指南

1. 創建 Servlet 應用程式結構　手動創建 Servlet 應用程式的目錄結構。您可以使用任何文件夾名稱，但這個範例中我們將其稱為 `SimpleServletApp`。

- **目錄結構:**

```
SimpleServletApp/
  src/
      com/
          example/
              HelloServlet.java
  webapp/
      WEB-INF/
          web.xml
```

2. 編寫 Servlet 代碼　在 `SimpleServletApp/src/com/example/` 中創建一個名為 `HelloServlet.java` 的文件，內容如下：

```java
package com.example;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class HelloServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().write("Hello World!");
```

```
    }
}
```

- **說明：**這個 Servlet 會對 HTTP GET 請求以純文本形式返回“Hello World!”。我們使用了一個簡單的 `doGet` 方法，沒有使用註解，以實現最大的兼容性和簡單性。

3. 創建 `web.xml` 部署描述符 在 `SimpleServletApp/webapp/WEB-INF/` 中創建一個名為 `web.xml` 的文件，內容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0
         version="4.0">
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>com.example.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

- **說明：** `web.xml` 文件將 `HelloServlet` 類映射到 `/hello` URL 模式。這是必需的，因為我們沒有使用像 `@WebServlet` 這樣的註解。

4. 編譯 Servlet 使用 `javac` 編譯 `HelloServlet.java` 文件為 `.class` 文件。您需要在類路徑中包含 `javax.servlet-api` 庫，這是由 WLP 提供的，但在編譯時必須可用。

- **步驟:**
  1. 找到 WLP 安裝中的 Servlet API JAR。例如，如果 WLP 安裝在 `/opt/ibm/wlp`，JAR 通常在：

     `/opt/ibm/wlp/dev/api/spec/com.ibm.websphere.javaee.servlet.4.0_1.0.x.jar`

     具體的文件名可能會根據您的 WLP 版本而有所不同。
  2. 從 `SimpleServletApp` 目錄運行以下命令：

     ```
     javac -cp "/opt/ibm/wlp/dev/api/spec/com.ibm.websphere.javaee.servlet.4.0_1.0.x.jar" src/com/example/
     ```
  3. 這將在 `SimpleServletApp/src/com/example/` 中創建 `HelloServlet.class`。

5. 將應用程式打包為 WAR 文件　手動組織編譯後的文件並創建 WAR 文件。

- **移動編譯後的類文件：**創建 `WEB-INF/classes` 目錄並移動編譯後的類文件：

```
mkdir -p webapp/WEB-INF/classes/com/example
mv src/com/example/HelloServlet.class webapp/WEB-INF/classes/com/example/
```

- **創建 WAR 文件：**從 `SimpleServletApp` 目錄使用 `jar` 命令將 `webapp` 文件夾打包為 WAR 文件：

```
cd webapp
jar -cvf ../myapp.war .
cd ..
```

這將在 `SimpleServletApp` 目錄中創建 `myapp.war`。

6. 在 WLP 上部署 WAR 文件　使用 `dropins` 目錄將 WAR 文件部署到 WLP 以進行自動部署。

- **找到 `dropins` 目錄：**找到您的 WLP 伺服器的 `dropins` 目錄。如果 WLP 安裝在 `/opt/ibm/wlp` 並且您的伺服器名稱為 `myServer`，路徑是：

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

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

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

- **啟動伺服器（如果未運行）：**如果 WLP 沒有運行，請啟動它：

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

如果它已經運行，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`。

7. 訪問應用程式　在瀏覽器中測試部署的 Servlet。

- **打開您的瀏覽器：**導航到：

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

  - `9080` 是 WLP 的默認 HTTP 端口。
  - `/myapp` 是來自 WAR 文件名的上下文根。

- `/hello` 是在 `web.xml` 中定義的 URL 模式。

- **預期結果：**您應該會看到：

  ```
  Hello World!
  ```

  以純文本顯示。

注意事項

- **沒有包含 JSP：**查詢要求 JSP 文件，但為了保持這個 非常簡單，我們專注於基本的 Servlet-only 應用程式。如果需要，可以添加 JSP。
- **上下文根：**上下文根 (`/myapp`) 來自 WAR 文件名。將 `myapp.war` 重命名以更改它（例如，`app.war` 以獲取 `/app`）。
- **端口號：** WLP 默認使用 `9080` 作為 HTTP 端口。如果不同，請檢查您的 `server.xml`。
- **Java 版本：**使用與 WLP 兼容的 Java 版本（例如，Java 8 或 11）。
- **故障排除：**如果不工作，請檢查 `/opt/ibm/wlp/usr/servers/myServer/logs/messages.log` 中的錯誤，確保 WAR 文件在 `dropins` 中，並驗證伺服器正在運行。

摘要

這個指南展示了如何：1. 創建簡單的 Servlet (`HelloServlet.java`) 和 `web.xml`。2. 使用 `javac` 編譯 Servlet，而不使用 Maven。3. 手動將其打包為 WAR 文件 (`myapp.war`)。4. 將其部署到 WLP 的 `dropins` 目錄。5. 訪問“Hello World!”於 `http://localhost:9080/myapp/hello`。

這提供了一個簡單的 Servlet 應用程式，在 WebSphere Liberty Profile 上運行，而不使用像 Maven 這樣的構建工具。