Java Servlet 应用程序与 Maven

这是一个完整的指南，介绍如何使用 Maven 构建项目和 Liberty Maven 插件部署简单的 Java Servlet 应用程序到 WebSphere Liberty Profile（WLP）。这个示例创建了一个基本的 Servlet，当访问时返回“Hello World!”，并将其打包成 WAR 文件，然后部署到 WLP。我们假设你已经启用了 `servlet-4.0` 功能的 WLP，因为它提供了必要的 Servlet 支持。

分步指南

1. 创建 Maven 项目结构

首先，设置一个标准的 Maven Web 应用程序结构。你可以手动创建它，或者使用 Maven 生成它。

- **目录结构：**

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

- **可选：使用 Maven 生成：**运行以下命令创建结构，然后根据需要进行调整：

```
mvn archetype:generate -DgroupId=com.example -DartifactId=simple-servlet-app -DarchetypeArtifactId=maven-
```

这将创建一个基本的 Web 应用程序结构，你将在接下来的步骤中进行修改。

2. 编写 Servlet 代码

在 `src/main/java/com/example/`中创建一个名为 `HelloServlet.java` 的文件，内容如下：

```
package com.example;

import javax.servlet.http.HttpServlet;
```

```java
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!");
    }
}
```

- **说明：**这个 Servlet 对 HTTP GET 请求响应“Hello World!”，使用简单的 `doGet` 方法，避免使用注解以兼容显式的 `web.xml` 配置。

3. 创建 `web.xml` 部署描述符

在 `src/main/webapp/WEB-INF/`中创建一个名为 `web.xml` 的文件，内容如下：

```xml
<?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_(
         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>
```

- **说明：**`web.xml` 文件定义了 `HelloServlet` 类，并将其映射到`/hello` URL 模式。这在我们不使用 `@WebServlet` 注解时是必要的。

4. 配置 Maven `pom.xml`

在 `SimpleServletApp/`目录中创建或更新 `pom.xml`，内容如下：

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>simple-servlet-app</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- Servlet API (provided by WLP) -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Maven WAR Plugin to build the WAR file -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
                <configuration>
                    <finalName>myapp</finalName>
                </configuration>
            </plugin>
            <!-- Liberty Maven Plugin for deployment -->
            <plugin>
                <groupId>io.openliberty.tools</groupId>
                <artifactId>liberty-maven-plugin</artifactId>
```

```
                <version>3.3.4</version>
                <configuration>
                    <installDirectory>/opt/ibm/wlp</installDirectory>
                    <serverName>myServer</serverName>
                    <appsDirectory>dropins</appsDirectory>
                    <looseApplication>false</looseApplication>
                    <stripVersion>true</stripVersion>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

- **说明：**
  - **坐标：**定义了项目的 `groupId`、`artifactId` 和 `version`。`packaging` 设置为 `war` 以表示 Web 应用程序。
  - **属性：**将 Java 8 设置为源和目标版本。
  - **依赖项：**包括 Servlet API，范围为 `provided`，因为它在运行时由 WLP 提供。
  - **Maven WAR 插件：**使用 `<finalName>` 将 WAR 文件名称配置为 `myapp.war`。
  - **Liberty Maven 插件：**配置将应用程序部署到`/opt/ibm/wlp` 的 Liberty 服务器，服务器名称为 `myServer`，部署到 `dropins` 目录。

5. 构建项目

从 `SimpleServletApp/`目录中使用 Maven 构建 WAR 文件：

```
mvn clean package
```

- **结果：**这将编译 Servlet，将其与 `web.xml` 打包成 `target/myapp.war`，并为部署做好准备。

6. 在 WebSphere Liberty 上部署和运行

确保你的 Liberty 服务器（`myServer`）已启用 `servlet-4.0` 功能。检查你的 `server.xml`，确保包含以下内容：

```
<featureManager>
    <feature>servlet-4.0</feature>
</featureManager>
```

使用 Liberty Maven 插件部署和运行应用程序：

```
mvn liberty:run
```

- **发生的事情：**
  - 如果服务器尚未运行，则在前台启动 Liberty 服务器。
  - 自动将 `myapp.war` 部署到 `dropins` 目录。
  - 保持服务器运行，直到停止。
- **验证部署：**查找类似以下的日志消息：

  ```
  [AUDIT   ] CWWKT0016I: Web应用程序可用（default_host）：http://localhost:9080/myapp/
  ```

  日志通常在`/opt/ibm/wlp/usr/servers/myServer/logs/console.log` 中。

7. 访问应用程序

在浏览器中导航到：

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

- **预期输出：**

  ```
  Hello World!
  ```

- **URL 分解：**
  - `9080`：WLP 的默认 HTTP 端口。
  - `/myapp`：从 WAR 文件名称（`myapp.war`）获得的上下文根。
  - `/hello`：从 `web.xml` 获得的 URL 模式。

8. 停止服务器

由于 `mvn liberty:run` 在前台运行服务器，因此通过在终端中按 `Ctrl+C` 停止它。

注意事项

- **先决条件：**
  - 必须在系统上安装并配置 Maven。
  - 必须在`/opt/ibm/wlp` 安装 Liberty，并且必须存在服务器实例 `myServer`。如果你的设置不同（例如`/usr/local/wlp` 或 `defaultServer`），请调整 `pom.xml` 中的 `installDirectory` 和 `serverName`。
  - 必须在 `server.xml` 中启用 `servlet-4.0` 功能。
- **替代部署：**

- 单独构建和部署：

```
mvn clean package
mvn liberty:deploy
```

如果需要，手动启动服务器：

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

- **端口配置：**如果你的 Liberty 服务器使用不同的 HTTP 端口，请检查 `server.xml` 中的 `<httpEndpoint>` 并相应地调整 URL。
- **上下文根：**重命名 `pom.xml` 中的 `<finalName>`（例如 `<finalName>app</finalName>`）以将上下文根更改为`/app`。
- **故障排除：**
  - 如果部署失败，请验证 `pom.xml` 中的 `installDirectory` 和 `serverName`。
  - 检查`/opt/ibm/wlp/usr/servers/myServer/logs/messages.log` 中的错误。
  - 确保使用 Java 8 或更高版本，与 `maven.compiler` 设置匹配。

总结

本指南演示了如何：1. 使用 Maven 设置一个简单的 Servlet（`HelloServlet.java`）和 `web.xml` 的项目。2. 配置 `pom.xml`，包括 Servlet API、Maven WAR 插件和 Liberty Maven 插件。3. 使用 `mvn clean package` 将应用程序打包成 `myapp.war`。4. 使用 `mvn liberty:run` 在 WLP 上部署和运行它。5. 在 `http://localhost:9080/myapp/hello` 访问“Hello World!”。

这提供了一种基于 Maven 的简化方法，用于在 WebSphere Liberty Profile 上开发和部署 Servlet 应用程序。