

# Java-Servlet-Anwendung mit Maven

Hier ist eine vollständige Anleitung zur Erstellung und Bereitstellung einer einfachen Java-Servlet-Anwendung auf WebSphere Liberty Profile (WLP) unter Verwendung von Maven zum Erstellen des Projekts und des Liberty-Maven-Plugins zur Bereitstellung. Dieses Beispiel erstellt eine grundlegende Servlet, die "Hello World!" zurückgibt, wenn darauf zugegriffen wird, baut sie in eine WAR-Datei mit Maven und stellt sie auf WLP bereit. Wir gehen davon aus, dass WLP bereits mit dem `servlet-4.0`-Feature aktiviert ist, da dies die notwendige Servlet-Unterstützung bietet.

---

## Schritt-für-Schritt-Anleitung

### 1. Erstellen der Maven-Projektstruktur

Beginnen Sie mit dem Einrichten einer Standard-Maven-Webanwendungsstruktur. Sie können dies manuell erstellen oder Maven verwenden, um sie zu generieren.

- **Verzeichnisstruktur:**

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

- **Optional mit Maven generieren:** Führen Sie diesen Befehl aus, um die Struktur zu erstellen, und passen Sie sie dann nach Bedarf an:

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

Dies erstellt eine grundlegende Webanwendungsstruktur, die Sie in den nächsten Schritten anpassen werden.

## 2. Schreiben des Servlet-Codes

Erstellen Sie eine Datei mit dem Namen `HelloServlet.java` in `src/main/java/com/example/` mit folgendem Inhalt:

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

- **Erklärung:** Dieses Servlet antwortet auf HTTP GET-Anfragen mit "Hello World!" im Klartext. Es verwendet eine einfache `doGet`-Methode und verzichtet auf Annotationen für die Kompatibilität mit der expliziten `web.xml`-Konfiguration.

## 3. Erstellen des `web.xml`-Bereitstellungsdeskriptors

Erstellen Sie eine Datei mit dem Namen `web.xml` in `src/main/webapp/WEB-INF/` mit folgendem Inhalt:

```
<?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.xsd"
          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>
```

- **Erklärung:** Die web.xml-Datei definiert die HelloServlet-Klasse und ordnet sie dem URL-Muster /hello zu. Dies ist notwendig, da wir keine @WebServlet-Annotationen verwenden.

#### 4. Konfigurieren des Maven pom.xml

Erstellen oder aktualisieren Sie pom.xml im Verzeichnis SimpleServletApp/ mit folgendem Inhalt:

```

<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 (von WLP bereitgestellt) -->
    <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 zum Erstellen der WAR-Datei -->
      <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 für die Bereitstellung -->
<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>

```

- **Erklärung:**

- **Koordinaten:** Definiert das Projekt mit groupId, artifactId und version. Die packaging ist auf war für eine Webanwendung gesetzt.
- **Eigenschaften:** Setzt Java 8 als Quell- und Zielversion.
- **Abhängigkeiten:** Enthält die Servlet-API mit provided-Bereich, da sie von WLP zur Laufzeit bereitgestellt wird.
- **Maven WAR Plugin:** Konfiguriert den Namen der WAR-Datei auf myapp.war mit <finalName>.
- **Liberty Maven Plugin:** Konfiguriert die Bereitstellung auf einen Liberty-Server unter /opt/ibm/wlp, Servername myServer, Bereitstellung im Verzeichnis dropins.

## 5. Bauen des Projekts

Bauen Sie die WAR-Datei aus dem Verzeichnis SimpleServletApp/ mit Maven:

```
mvn clean package
```

- **Ergebnis:** Dies kompiliert das Servlet, packt es mit web.xml in target/myapp.war und bereitet es für die Bereitstellung vor.

## 6. Bereitstellen und Ausführen auf WebSphere Liberty

Stellen Sie sicher, dass Ihr Liberty-Server (`myServer`) mit dem `servlet-4.0`-Feature aktiviert ist. Überprüfen Sie Ihre `server.xml` auf:

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

Bereitstellen und Ausführen der Anwendung mit dem Liberty-Maven-Plugin:

```
mvn liberty:run
```

- **Was passiert:**

- Startet den Liberty-Server im Vordergrund (falls er nicht bereits läuft).
- Bereitstellt `myapp.war` automatisch im Verzeichnis `dropins`.
- Lässt den Server laufen, bis er gestoppt wird.

- **Überprüfen der Bereitstellung:** Suchen Sie nach einer Logmeldung wie:

```
[AUDIT    ] CWWKT0016I: Webanwendung verfügbar (default_host): http://localhost:9080/myapp/
```

Protokolle befinden sich normalerweise in `/opt/ibm/wlp/usr/servers/myServer/logs/console.log`.

## 7. Zugriff auf die Anwendung

Öffnen Sie einen Browser und navigieren Sie zu:

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

- **Erwartete Ausgabe:**

Hello World!

- **URL-Zusammenfassung:**

- 9080: Standard-HTTP-Port für WLP.
- `/myapp`: Kontextwurzel aus dem Namen der WAR-Datei (`myapp.war`).
- `/hello`: URL-Muster aus `web.xml`.

## 8. Stoppen des Servers

Da `mvn liberty:run` den Server im Vordergrund ausführt, stoppen Sie ihn, indem Sie `Ctrl+C` in der Konsole drücken.

## Hinweise

- **Voraussetzungen:**

- Maven muss auf Ihrem System installiert und konfiguriert sein.
- Liberty muss unter /opt/ibm/wlp installiert sein und die Serverinstanz myServer muss existieren.  
Passen Sie installDirectory und serverName in pom.xml an, wenn Ihre Einrichtung abweicht (z.B. /usr/local/wlp oder defaultServer).
- Das servlet-4.0-Feature muss in server.xml aktiviert sein.

- **Alternative Bereitstellung:**

- Zum separaten Bauen und Bereitstellen:

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

Starten Sie den Server manuell, falls erforderlich:

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

- **Portkonfiguration:** Wenn Ihr Liberty-Server einen anderen HTTP-Port verwendet, überprüfen Sie server.xml auf <httpEndpoint> und passen Sie die URL entsprechend an.

- **Kontextwurzel:** Benennen Sie <finalName> in pom.xml um (z.B. <finalName>app</finalName>), um die Kontextwurzel zu /app zu ändern.

- **Fehlerbehebung:**

- Wenn die Bereitstellung fehlschlägt, überprüfen Sie installDirectory und serverName in pom.xml.
  - Überprüfen Sie die Protokolle unter /opt/ibm/wlp/usr/servers/myServer/logs/messages.log auf Fehler.
  - Stellen Sie sicher, dass Java 8 oder höher verwendet wird, was den maven.compiler-Einstellungen entspricht.
- 

## Zusammenfassung

Diese Anleitung zeigt, wie Sie: 1. Ein Maven-Projekt mit einer einfachen Servlet (HelloServlet.java) und web.xml einrichten. 2. pom.xml mit der Servlet-API, dem Maven WAR Plugin und dem Liberty Maven Plugin konfigurieren. 3. Die Anwendung in myapp.war mit mvn clean package bauen. 4. Sie auf WLP mit mvn liberty:run bereitstellen und ausführen. 5. "Hello World!" unter <http://localhost:9080/myapp/hello> aufrufen.

Dies bietet einen gestrafften, Maven-basierten Ansatz zur Entwicklung und Bereitstellung einer Servlet-Anwendung auf WebSphere Liberty Profile.