Java Servlet Anwendung auf WebSphere Liberty Profil

Hier ist eine vollständige Anleitung zum Ausführen einer sehr einfachen Java-Servlet-Anwendung auf WebSphere Liberty Profile (WLP). Dieses Beispiel erstellt eine grundlegende Servlet, die "Hello World!"zurückgibt und sie auf WLP über das `dropins`-Verzeichnis bereitstellt. Wir gehen davon aus, dass Sie WLP bereits mit der `servlet-4.0`-Funktion eingerichtet haben, da diese die notwendige Servlet-Unterstützung bietet. Diese Anleitung verzichtet auf die Verwendung von Maven und bietet den gesamten erforderlichen Code und die Bereitstellungsschritte.

Schritt-für-Schritt-Anleitung

1. Erstellen der Servlet-Anwendungsstruktur Erstellen Sie eine Verzeichnisstruktur für Ihre Servlet-Anwendung manuell. Sie können jeden Ordnernamen verwenden, aber für dieses Beispiel nennen wir ihn `SimpleServletApp`.

- **Verzeichnisstruktur:**

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

2. Schreiben des Servlet-Codes Erstellen Sie eine Datei mit dem Namen `HelloServlet.java` in `SimpleServletApp/src/com/example/` mit folgendem Inhalt:

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

- **Erklärung:** Diese Servlet antwortet auf HTTP GET-Anfragen mit “Hello World!”im Klartext. Wir verwenden eine einfache `doGet`-Methode ohne Annotationen für maximale Kompatibilität und Einfachheit.

3. Erstellen der `web.xml`-Bereitstellungsbeschreibung Erstellen Sie eine Datei mit dem Namen `web.xml` in `SimpleServletApp/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
         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 ordnet die `HelloServlet`-Klasse dem URL-Muster `/hello` zu. Dies ist erforderlich, da wir keine Annotationen wie `@WebServlet` verwenden.

4. Kompilieren des Servlets Kompilieren Sie die `HelloServlet.java`-Datei in eine `.class`-Datei mit `javac`. Sie benötigen die `javax.servlet-api`-Bibliothek in Ihrem Klassenpfad, die von WLP bereitgestellt wird, aber während der Kompilierung verfügbar sein muss.

- **Schritte:**

  1. Legen Sie die Servlet-API-JAR in Ihrer WLP-Installation fest. Zum Beispiel, wenn WLP unter `/opt/ibm/wlp` installiert ist, befindet sich die JAR normalerweise unter:

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

     Der genaue Dateiname kann je nach WLP-Version variieren.

2. Führen Sie den folgenden Befehl aus dem `SimpleServletApp`-Verzeichnis aus:

   ```
   javac -cp "/opt/ibm/wlp/dev/api/spec/com.ibm.websphere.javaee.servlet.4.0_1.0.x.jar" src/com/example/
   ```

3. Dies erstellt `HelloServlet.class` in `SimpleServletApp/src/com/example/`.

5. Paketieren der Anwendung in eine WAR-Datei Ordnen Sie die kompilierten Dateien und erstellen Sie eine WAR-Datei manuell.

- **Verschieben der kompilierten Klasse:** Erstellen Sie ein `WEB-INF/classes`-Verzeichnis und verschieben Sie die kompilierten Klassen:

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

- **Erstellen der WAR-Datei:** Vom `SimpleServletApp`-Verzeichnis aus verwenden Sie den `jar`-Befehl, um das `webapp`-Verzeichnis in eine WAR-Datei zu paketieren:

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

  Dies erstellt `myapp.war` im `SimpleServletApp`-Verzeichnis.

6. Bereitstellen der WAR-Datei auf WLP Bereitstellen Sie die WAR-Datei auf WLP unter Verwendung des `dropins`-Verzeichnisses für die automatische Bereitstellung.

- **Lokalisieren des `dropins`-Verzeichnisses:** Finden Sie das `dropins`-Verzeichnis Ihres WLP-Servers. Wenn WLP unter `/opt/ibm/wlp` installiert ist und Ihr Server `myServer` heißt, lautet der Pfad:

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

- **Kopieren der WAR-Datei:** Verschieben Sie die WAR-Datei in das `dropins`-Verzeichnis:

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

- **Starten des Servers (falls nicht bereits gestartet):** Wenn WLP nicht läuft, starten Sie es:

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

  Wenn es bereits läuft, wird es die WAR-Datei automatisch erkennen und bereitstellen.

- **Überprüfen der Bereitstellung:** Überprüfen Sie die Serverprotokolle oder die Konsole auf eine Nachricht wie:

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

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

7. Zugriff auf die Anwendung Testen Sie die bereitgestellte Servlet in einem Browser.

- **Öffnen Sie Ihren Browser:** Navigieren Sie zu:

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

  - `9080` ist der Standard-HTTP-Port von WLP.
  - `/myapp` ist der Kontextstamm aus dem Namen der WAR-Datei.
  - `/hello` ist das in `web.xml` definierte URL-Muster.

- **Erwartetes Ergebnis:** Sie sollten sehen:

  ```
  Hello World!
  ```

  als Klartext angezeigt.

Hinweise

- **Keine JSP enthalten:** Die Anfrage fragte nach JSP-Dateien, aber um dies *sehr einfach* zu halten, habe ich mich auf eine grundlegende Servlet-Only-Anwendung konzentriert. JSPs würden zusätzliche Dateien und Komplexität erfordern, die hinzugefügt werden können, falls erforderlich.
- **Kontextstamm:** Der Kontextstamm (`/myapp`) stammt aus dem Namen der WAR-Datei. Benennen Sie `myapp.war` um, um ihn zu ändern (z. B. `app.war` für `/app`).
- **Portnummer:** WLP verwendet standardmäßig `9080` für HTTP. Überprüfen Sie Ihre `server.xml`, falls dies anders ist.
- **Java-Version:** Verwenden Sie eine Java-Version, die mit WLP kompatibel ist (z. B. Java 8 oder 11).
- **Fehlerbehebung:** Wenn es nicht funktioniert, überprüfen Sie `/opt/ibm/wlp/usr/servers/myServer/logs/message` auf Fehler, stellen Sie sicher, dass die WAR-Datei sich im `dropins`-Verzeichnis befindet, und überprüfen Sie, ob der Server läuft.

Zusammenfassung

Diese Anleitung zeigt, wie Sie: 1. Ein einfaches Servlet (`HelloServlet.java`) und `web.xml` erstellen. 2. Das Servlet ohne Maven mit `javac` kompilieren. 3. Es manuell in eine WAR-Datei (`myapp.war`) paketieren. 4. Es im `dropins`-Verzeichnis von WLP bereitstellen. 5. "Hello World!"unter `http://localhost:9080/myapp/hello` aufrufen.

Dies bietet eine einfache, minimale Servlet-Anwendung, die auf WebSphere Liberty Profile ohne Build-Tools wie Maven läuft.