

# Application Java Servlet sur WebSphere Liberty Profile

Voici un guide complet pour exécuter une application Java Servlet très simple sur WebSphere Liberty Profile (WLP). Cet exemple crée un Servlet de base qui renvoie "Hello World!" et le déploie sur WLP en utilisant le répertoire `dropins`. Nous supposons que vous avez déjà configuré WLP avec la fonctionnalité `servlet-4.0` activée, car cela fournit le support Servlet nécessaire. Ce guide évite l'utilisation de Maven et fournit tout le code et les étapes de déploiement nécessaires.

---

## Guide Étape par Étape

**1. Créer la Structure de l'Application Servlet** Créez manuellement une structure de répertoires pour votre application Servlet. Vous pouvez utiliser n'importe quel nom de dossier, mais pour cet exemple, nous l'appellerons `SimpleServletApp`.

- **Structure des Répertoires :**

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

**2. Écrire le Code du Servlet** Créez un fichier nommé `HelloServlet.java` dans `SimpleServletApp/src/com/example/` avec le contenu suivant :

```
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!");
    }
}

```

- **Explication :** Ce Servlet répond aux requêtes HTTP GET avec “Hello World!” en texte brut. Nous utilisons une simple méthode `doGet` sans annotations pour une compatibilité et une simplicité maximales.

**3. Créer le Descripteur de Déploiement `web.xml`** Créez un fichier nommé `web.xml` dans `SimpleServletApp/webapp/` avec le contenu suivant :

```

<?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_1_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>

```

- **Explication :** Le fichier `web.xml` mappe la classe `HelloServlet` au motif d’URL `/hello`. Cela est nécessaire puisque nous n’utilisons pas d’annotations comme `@WebServlet`.

**4. Compiler le Servlet** Compilez le fichier `HelloServlet.java` en un fichier `.class` en utilisant `javac`. Vous aurez besoin de la bibliothèque `javax.servlet-api` dans votre classpath, qui est fournie par WLP mais doit être disponible pendant la compilation.

- **Étapes :**

1. Localisez le fichier JAR de l’API Servlet dans votre installation WLP. Par exemple, si WLP est installé à `/opt/ibm/wlp`, le JAR est généralement à :

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

Le nom du fichier peut varier en fonction de votre version de WLP.

2. Exécutez la commande suivante à partir du répertoire `SimpleServletApp` :

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

3. Cela crée `HelloServlet.class` dans `SimpleServletApp/src/com/example/`.

**5. Emballer l'Application dans un Fichier WAR** Organisez les fichiers compilés et créez un fichier WAR manuellement.

- **Déplacer la Classe Compilée** : Créez un répertoire `WEB-INF/classes` et déplacez les fichiers de classe compilés :

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

- **Créer le Fichier WAR** : À partir du répertoire `SimpleServletApp`, utilisez la commande `jar` pour emballer le dossier `webapp` dans un fichier WAR :

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

Cela crée `myapp.war` dans le répertoire `SimpleServletApp`.

**6. Déployer le Fichier WAR sur WLP** Déployez le fichier WAR sur WLP en utilisant le répertoire `dropins` pour un déploiement automatique.

- **Localiser le Répertoire dropins** : Trouvez le répertoire `dropins` de votre serveur WLP. Si WLP est installé à `/opt/ibm/wlp` et que votre serveur s'appelle `myServer`, le chemin est :

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

- **Copier le Fichier WAR** : Déplacez le fichier WAR dans le répertoire `dropins` :

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

- **Démarrer le Serveur (si Non Démarré)** : Si WLP n'est pas en cours d'exécution, démarrez-le :

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

S'il est déjà en cours d'exécution, il détectera et déploiera automatiquement le fichier WAR.

- **Vérifier le Déploiement** : Vérifiez les journaux du serveur ou la console pour un message comme :

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

Les journaux se trouvent dans `/opt/ibm/wlp/usr/servers/myServer/logs/console.log`.

## 7. Accéder à l'Application

Testez le Servlet déployé dans un navigateur.

- **Ouvrir Votre Navigateur** : Accédez à :

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

- 9080 est le port HTTP par défaut de WLP.
- /myapp est la racine du contexte à partir du nom du fichier WAR.
- /hello est le motif d'URL défini dans `web.xml`.

- **Résultat Attendu** : Vous devriez voir :

Hello World!

affiché en texte brut.

---

### Notes

- **Pas de JSP Inclus** : La demande concernait des fichiers JSP, mais pour garder cela *très simple*, nous nous sommes concentrés sur une application Servlet uniquement de base. Les JSP nécessiteraient des fichiers et une complexité supplémentaires, qui peuvent être ajoutés si nécessaire.
  - **Racine du Contexte** : La racine du contexte (/myapp) provient du nom du fichier WAR. Renommez `myapp.war` pour le changer (par exemple, `app.war` pour /app).
  - **Numéro de Port** : WLP utilise par défaut 9080 pour HTTP. Vérifiez votre `server.xml` s'il est différent.
  - **Version de Java** : Utilisez une version de Java compatible avec WLP (par exemple, Java 8 ou 11).
  - **Dépannage** : Si cela ne fonctionne pas, vérifiez `/opt/ibm/wlp/usr/servers/myServer/logs/messages.log` pour les erreurs, assurez-vous que le fichier WAR est dans `dropins` et vérifiez que le serveur est en cours d'exécution.
- 

### Résumé

Ce guide montre comment : 1. Créer un Servlet simple (`HelloServlet.java`) et `web.xml`. 2. Compiler le Servlet sans Maven en utilisant `javac`. 3. L'emballer dans un fichier WAR (`myapp.war`) manuellement. 4. Le déployer dans le répertoire `dropins` de WLP. 5. Accéder à "Hello World!" à `http://localhost:9080/myapp/hello`.

Cela fournit une application Servlet minimale et directe fonctionnant sur WebSphere Liberty Profile sans outils de construction comme Maven.