WebSphere Liberty - Hello World Application

Here's a simple example to run a "Hello World"application on your WebSphere Liberty server, assuming you've already set it up with the `javaee-8.0` feature enabled as per previous instructions. This example uses a basic JSP (JavaServer Pages) file and deploys it using the `dropins` directory for simplicity.

Step-by-Step Guide

1. Create the Application Directory and File You'll create a small web application by placing a JSP file in the `dropins` directory of your Liberty server. The `dropins` directory allows Liberty to automatically detect and deploy applications.

- **Locate the `dropins` directory**: Navigate to the `dropins` directory within your server's folder. If your Liberty installation is located at `/opt/ibm/wlp` and your server is named `myServer`, the path is:

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

  Replace `/opt/ibm/wlp` with your actual Liberty installation directory and `myServer` with your server's name.

- **Create an exploded WAR directory**: Inside the `dropins` directory, create a directory named `myApp.war`. The `.war` extension tells Liberty to treat this as a web application. Use this command:

  ```
  mkdir -p /opt/ibm/wlp/usr/servers/myServer/dropins/myApp.war
  ```

- **Create the `index.jsp` file**: Inside `myApp.war`, create a file named `index.jsp` with the following content to display "Hello World!":

  ```
  <html>
  <body>
  <h2>Hello World!</h2>
  </body>
  </html>
  ```

  You can create it directly with a command like:

  ```
  echo '<html><body><h2>Hello World!</h2></body></html>' > /opt/ibm/wlp/usr/servers/myServer/dropins/myApp.
  ```

  Alternatively, use a text editor to create `index.jsp` and save it in that location.

2. Start the Server (if Not Already Running) If your server isn't running, you need to start it so it can deploy and serve the application.

- **Navigate to the `bin` directory**: Go to the `bin` directory in your Liberty installation:

```
cd /opt/ibm/wlp/bin
```

- **Start the server**: Run the server in foreground mode to see the output directly:

  ```
  ./server run myServer
  ```

  Alternatively, start it in the background:

  ```
  ./server start myServer
  ```

  If the server is already running, skip this step—Liberty will detect the new application automatically.

3. Verify Application Deployment Liberty will deploy the `myApp.war` application automatically when it detects it in the `dropins` directory.

- **Check the console output**: If you started the server in foreground mode, watch for a message like:

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

  This confirms the application is deployed and available.

- **Check logs (if running in background)**: If the server is running in the background, look in the log file at:

  ```
  /opt/ibm/wlp/usr/servers/myServer/logs/console.log
  ```

  for the same deployment message.

4. Access the Application Now that the application is deployed, you can view it in a web browser.

- **Open your browser**: Navigate to:

  ```
  http://localhost:9080/myApp/
  ```

  The context root `/myApp` comes from the `myApp.war` directory name, and `index.jsp` is served by default as a welcome file.

- **Expected result**: You should see a page displaying:

  ```
  Hello World!
  ```

  in large text.

Notes

- **Port number**: This example uses port `9080`, the default HTTP port for Liberty. If you've configured a different port, adjust the URL accordingly (e.g., `http://localhost:YOUR_PORT/myApp/`).
- **Permissions**: Ensure you have write permissions to the `dropins` directory.
- **JSP support**: The `javaee-8.0` feature includes JSP support, so no additional configuration is needed.
- **Troubleshooting**: If the page doesn't load, check the server logs at `/opt/ibm/wlp/usr/servers/myServer/logs/me` for errors, or ensure the server is running and the files are in the correct location.

This simple example demonstrates how to deploy and run a "Hello World"application on your WebSphere Liberty server using the `dropins` directory for automatic deployment. Enjoy your first Liberty application!

Here's an explanation of how a "Hello World"application works when deployed on WebSphere Liberty:

1. The Dropins Directory

WebSphere Liberty provides a convenient feature for deploying applications through its `dropins` directory. This is a special folder where you can simply place your application—either as a WAR (Web Application Archive) file or an exploded WAR directory—and Liberty will automatically detect and deploy it. Unlike traditional deployment methods, you don't need to manually configure the application in the `server.xml` file. When Liberty starts or notices a change in the `dropins` folder (like adding a new application), it kicks off the deployment process automatically.

2. Using an Exploded WAR Directory

In this example, the application is deployed as an exploded WAR directory named `myApp.war` instead of a single packaged WAR file. An exploded WAR is just a folder containing all the contents of a web application (like HTML, JSP files, and other resources) in an unzipped form. Liberty treats this directory exactly like it would a WAR file, deploying it as a fully functional web application. This method is especially useful during development because you can edit the files directly (e.g., tweak the HTML or JSP) without needing to repackage everything into a WAR file.

3. The JSP File

The heart of this "Hello World"application is a file called `index.jsp`, a JavaServer Page (JSP). This file contains basic HTML to display "Hello World!"on the screen and could include Java code if needed (though in this case, it's kept simple). When you access the application, Liberty dynamically compiles the JSP into a servlet—a small Java program that generates the webpage—and serves it to your browser.

4. Enabling Java EE Features

To make all this work, Liberty relies on specific features being enabled in its configuration file, `server.xml`. Here, the `javaee-8.0` feature is activated, which provides support for technologies like JSPs, servlets, and other components of the Java Enterprise Edition (EE) 8 platform. This feature ensures Liberty has the necessary libraries and settings loaded to run the application smoothly.

5. Automatic Deployment Process

Once you place the `myApp.war` directory into the `dropins` folder and start Liberty (or if it's already running), the server automatically detects and deploys the application. You'll see log messages in Liberty's output indicating that the application has started and is available at a specific URL. This hands-off deployment process makes it quick and easy to get an application up and running.

6. Accessing the Application: Context Root

The URL where you can access the application is determined by its **context root**, which is simply the name of the WAR file or directory. In this case, since the directory is named `myApp.war`, the context root becomes `/myApp`. By default, Liberty runs on port `9080`, so the full URL to access the application is `http://localhost:9080/myApp/`.

7. Serving the Welcome File

When you visit `http://localhost:9080/myApp/` in your browser, Liberty needs to figure out what to display. It looks for a **welcome file**—a default page to serve when no specific file is requested. In this case, `index.jsp` is located in the root of the `myApp.war` directory and is recognized as a welcome file by default. Liberty compiles and serves `index.jsp`, which then displays "Hello World!"on your screen.

Summary

This "Hello World"application showcases how WebSphere Liberty simplifies web application deployment. By leveraging the `dropins` directory for automatic deployment, supporting exploded WARs for easy editing, and using the `javaee-8.0` feature for Java EE compatibility, Liberty lets you quickly deploy and test a simple web app without complex setup. The result is a lightweight, developer-friendly way to see "Hello World!"at `http://localhost:9080/myApp/`, perfect for learning or prototyping.