JAR vs WAR

In der Java-Verpackung sind **JAR** (Java ARchive) und **WAR** (Web ARchive) beide Archivdateiformate, die auf der ZIP-Struktur basieren, aber unterschiedliche Zwecke erfüllen und unterschiedliche Merkmale haben. Im Folgenden wird eine detaillierte Erklärung ihrer Unterschiede gegeben:

---

1. Zweck

- **JAR**: Eine JAR-Datei ist ein allgemeines Archiv, das verwendet wird, um Java-Klassen, Bibliotheken oder eigenständige Anwendungen zu verpacken. Sie wird häufig verwendet, um wiederverwendbare Bibliotheken (z. B. Hilfsklassen) oder ausführbare Java-Programme zu verteilen, die auf der Java Virtual Machine (JVM) laufen können.
- **WAR**: Eine WAR-Datei ist speziell für Webanwendungen konzipiert. Sie verpackt alle Komponenten, die für eine Webanwendung benötigt werden, wie z. B. Servlets, JSPs (JavaServer Pages), HTML, CSS, JavaScript und Konfigurationsdateien, in eine einzige Einheit zur Bereitstellung auf einem Webserver oder Anwendungsserver (z. B. Apache Tomcat, JBoss).

---

2. Inhalte

- **JAR**: Enthält Java-Klassendateien (`.class`), eine `MANIFEST.MF`-Datei (im `META-INF`-Verzeichnis), die Metadaten bereitstellt (z. B. die Hauptklasse für ausführbare JARs), und optionale Ressourcen wie Konfigurationsdateien, Bilder oder Eigenschaftsdateien.
- **WAR**: Enthält web-spezifische Komponenten mit einer definierten Struktur:
  - **WEB-INF/**: Ein obligatorisches Verzeichnis, das enthält:
    - `web.xml` (Bereitstellungsdeskriptor zur Konfiguration von Servlets, Zuordnungen usw.),
    - `classes/` (kompilierte Java-Klassen),
    - `lib/` (JAR-Dateien, die als Abhängigkeiten von der Webanwendung verwendet werden).
  - Statische Ressourcen (z. B. HTML, CSS, JavaScript) befinden sich normalerweise im Stammverzeichnis oder in Unterverzeichnissen außerhalb von `WEB-INF`, obwohl JSPs auch innerhalb von `WEB-INF` platziert werden können, um den direkten Zugriff zu beschränken.

---

3. Struktur

- **JAR**: Hat eine flache Struktur, die hauptsächlich aus Klassendateien und Ressourcen besteht, wobei die Manifestdatei Metadaten angibt. Beispiel:

```
myapp.jar
  META-INF/
      MANIFEST.MF
  com/
      example/
          MyClass.class
  resources/
       config.properties
```

- **WAR**: Folgt einer hierarchischen Struktur, die für Webanwendungen angepasst ist. Beispiel:

```
mywebapp.war
  index.html
  css/
      style.css
  WEB-INF/
      web.xml
      classes/
          com/
              example/
                  MyServlet.class
      lib/
          dependency.jar
```

---

4. Bereitstellung und Nutzung

- **JAR**:
  - Wird in den Klassenpfad einer Anwendung aufgenommen, um Bibliotheken oder wiederverwendbaren Code bereitzustellen.
  - Wenn es ausführbar ist (mit einer `Main-Class` in der Manifestdatei angegeben), kann es direkt mit `java -jar myapp.jar` ausgeführt werden.
- **WAR**:
  - Wird auf einem Servlet-Container oder Anwendungsserver (z. B. Tomcat) bereitgestellt, der es entpackt und die Webanwendung unter einer bestimmten URL (Kontextpfad) zugänglich macht.
  - Kann nicht eigenständig ausgeführt werden; es verlässt sich auf den Server, um seinen Lebenszyklus zu verwalten.

---

5. Abhängigkeitsverwaltung

- **JAR**: Enthält keine anderen JAR-Dateien; Abhängigkeiten werden extern verwaltet (z. B. über den Klassenpfad oder Build-Tools wie Maven).
- **WAR**: Kann abhängige JAR-Dateien im `WEB-INF/lib/`-Verzeichnis bündeln, wodurch sichergestellt wird, dass die Webanwendung spezifische Versionen von Bibliotheken verwendet, die von anderen Anwendungen auf demselben Server isoliert sind.

---

6. Erstellung

Sowohl JAR- als auch WAR-Dateien können mit dem `jar`-Befehlszeilen-Tool aus dem JDK oder modernen Build-Tools wie Maven oder Gradle erstellt werden: - **JAR-Beispiel**: `jar cvf myapp.jar -C classes .` - **WAR-Beispiel**: `jar cvf mywebapp.war .` (vorausgesetzt, das Verzeichnis ist mit `WEB-INF` usw. strukturiert). - In **Maven** wird der Verpackungstyp in der `pom.xml` angegeben: - `<packaging>jar</packaging>` für JAR. - `<packaging>war</packaging>` für WAR.

Zusammenfassende Tabelle

Merkmal	JAR	WAR
Vollständiger Name	Java ARchive	Web ARchive
Zweck	Bibliotheken, eigenständige Apps	Webanwendungen
Inhalte	Klassen, Manifest, Ressourcen	Servlets, JSPs, HTML, WEB-INF, etc.
Struktur	Flach	Hierarchisch (WEB-INF, etc.)
Bereitstellung	Klassenpfad oder `java -jar`	Webserver (z. B. Tomcat)
Abhängigkeiten	Extern	Gebündelt in `WEB-INF/lib/`

Wichtige Erkenntnis

Im Wesentlichen ist eine **JAR** ein vielseitiges Archiv für allgemeinen Java-Code, während eine **WAR** ein spezialisiertes Archiv für Webanwendungen ist, das eine Struktur und Komponenten hat, die für die Bereitstellung auf einem Webserver angepasst sind. Das Verständnis ihrer unterschiedlichen Rollen stellt sicher, dass sie in Java-Entwicklungs- und Bereitstellungsszenarien korrekt verwendet werden.