JAR vs WAR

En Java, le packaging, **JAR** (Java ARchive) et **WAR** (Web ARchive) sont tous deux des formats de fichiers d'archives basés sur la structure ZIP, mais ils servent des objectifs distincts et ont des caractéristiques différentes. Voici une explication détaillée de leurs différences :

---

1. Objectif

- **JAR** : Un fichier JAR est une archive à usage général utilisée pour empaqueter des classes Java, des bibliothèques ou des applications autonomes. Il est couramment employé pour distribuer des bibliothèques réutilisables (par exemple, des classes utilitaires) ou des programmes Java exécutables qui peuvent s'exécuter sur la machine virtuelle Java (JVM).
- **WAR** : Un fichier WAR est spécifiquement conçu pour les applications web. Il empaquette tous les composants nécessaires pour une application web, tels que les servlets, les JSP (JavaServer Pages), le HTML, le CSS, le JavaScript et les fichiers de configuration, en une seule unité pour le déploiement sur un serveur web ou un serveur d'applications (par exemple, Apache Tomcat, JBoss).

---

2. Contenu

- **JAR** : Contient des fichiers de classes Java (`.class`), un fichier `MANIFEST.MF` (sous le répertoire `META-INF`) qui fournit des métadonnées (par exemple, la classe principale pour les JAR exécutables), et des ressources optionnelles comme des fichiers de configuration, des images ou des fichiers de propriétés.
- **WAR** : Inclut des composants spécifiques au web avec une structure définie :
  - **WEB-INF/** : Un répertoire obligatoire contenant :
    - `web.xml` (descripteur de déploiement pour configurer les servlets, les mappages, etc.),
    - `classes/` (classes Java compilées),
    - `lib/` (fichiers JAR utilisés comme dépendances par l'application web).
  - Les ressources statiques (par exemple, HTML, CSS, JavaScript) résident généralement dans le répertoire racine ou dans des sous-répertoires en dehors de `WEB-INF`, bien que les JSP puissent être placées à l'intérieur de `WEB-INF` pour restreindre l'accès direct.

---

3. Structure

- **JAR** : A une structure plate, principalement constituée de fichiers de classes et de ressources, avec le fichier manifeste spécifiant les métadonnées. Exemple :

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

- **WAR** : Suit une structure hiérarchique adaptée aux applications web. Exemple :

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

---

4. Déploiement et Utilisation

- **JAR** :
  - Inclu dans le classpath d'une application pour fournir des bibliothèques ou du code réutilisable.
  - Si exécutable (avec une `Main-Class` spécifiée dans le manifeste), il peut être exécuté directement en utilisant `java -jar myapp.jar`.
- **WAR** :
  - Déployé sur un conteneur de servlets ou un serveur d'applications (par exemple, Tomcat), qui le décompresse et rend l'application web accessible à une URL spécifique (chemin de contexte).
  - Non exécutable de manière autonome ; il dépend du serveur pour gérer son cycle de vie.

---

5. Gestion des Dépendances

- **JAR** : Ne contient pas intrinsèquement d'autres JAR ; les dépendances sont gérées de manière externe (par exemple, via le classpath ou des outils de construction comme Maven).
- **WAR** : Peut regrouper des JAR dépendants à l'intérieur du répertoire `WEB-INF/lib/`, garantissant que l'application web utilise des versions spécifiques de bibliothèques, isolées des autres applications sur le même serveur.

---

6. Création

Les fichiers JAR et WAR peuvent être créés à l'aide de l'outil de ligne de commande `jar` du JDK ou des outils de construction modernes comme Maven ou Gradle : - **Exemple JAR** : `jar cvf myapp.jar -C classes .` - **Exemple WAR** : `jar cvf mywebapp.war .` (en supposant que le répertoire est structuré avec `WEB-INF`, etc.). - Dans **Maven**, le type de packaging est spécifié dans le `pom.xml` : - `<packaging>jar</packaging>` pour JAR. - `<packaging>war</packaging>` pour WAR.

Tableau Résumé

Caractéristique	JAR	WAR
Nom Complet	Java ARchive	Web ARchive
Objectif	Bibliothèques, applications autonomes	Applications web
Contenu	Classes, manifeste, ressources	Servlets, JSPs, HTML, WEB-INF, etc.
Structure	Plate	Hiérarchique (WEB-INF, etc.)
Déploiement	Classpath ou `java -jar`	Serveur web (par exemple, Tomcat)
Dépendances	Externe	Regroupées dans `WEB-INF/lib/`

Conclusion Clé

En résumé, un **JAR** est une archive polyvalente pour le code Java général, tandis qu'un **WAR** est une archive spécialisée pour les applications web, complète avec une structure et des composants adaptés au déploiement sur un serveur web. Comprendre leurs rôles distincts assure une utilisation appropriée dans les scénarios de développement et de déploiement Java.