# 02.01 - JAR vs WAR - Comprehensive Notes

## 1. Introduction

When developing Java applications, we often package them into **JAR (Java ARchive)** or **WAR (Web Application ARchive)** files for deployment. These formats serve different purposes:

| Format | Usage |
|---|---|
| **JAR (Java Archive)** | Used for standalone applications and libraries |
| **WAR (Web Application Archive)** | Used for deploying web applications on a servlet container |

---

## 2. Understanding JAR (Java ARchive)

### What is a JAR file?

A **JAR file** is a **compressed package of compiled Java classes, metadata, and resources**. It allows Java programs to be bundled and distributed as a single file.

### When to Use JAR?

✅ **Standalone Java applications**
✅ **Java libraries** (like logging libraries, utility functions)
✅ **Spring Boot microservices**
✅ **CLI-based applications**

## JAR File Structure

```
/META-INF
  ├── MANIFEST.MF  (Contains metadata like Main-Class)
/com/example (package structure)
  ├── MyClass.class
  ├── Utils.class
/resources
  ├── config.properties
  ├── image.png
```

## Running a JAR File

1. If the JAR has a `Main-Class` defined in `MANIFEST.MF`, run:

   ```
   java -jar myapp.jar
   ```

2. If it's just a library JAR (no main method), it is used as a dependency in other projects.

## Creating a JAR File with Gradle

Gradle is a popular build tool used to create JAR files. Here's how to configure it:

### 1️⃣ Apply the Java Plugin (build.gradle)

For **Gradle (Groovy DSL)**:

```
plugins {
    id 'java'
}
```

```
jar {
    manifest {
        attributes(
            'Main-Class': 'com.example.MainApp'
        )
    }
}
```

For **Gradle (Kotlin DSL)**:

```
plugins {
    java
}

tasks.jar {
    manifest {
        attributes["Main-Class"] = "com.example.MainApp"
    }
}
```

## 2️⃣ Build the JAR File

Run the command:

```
gradle jar
```

This creates a JAR file inside `build/libs/`.

## 3️⃣ Run the JAR

```
java -jar build/libs/myapp.jar
```

---

# 3. Understanding WAR (Web Application ARchive)

## What is a WAR file?

A **WAR file** is a package for Java web applications. It contains Java classes, JSPs, HTML, CSS, JavaScript, and configuration files required to run a web application.

## When to Use WAR?

✅ **Servlet-based web applications**
✅ **Traditional Java EE applications**
✅ **Spring MVC-based applications**
✅ **Enterprise-level applications needing external deployment**

## WAR File Structure

```
/META-INF
  ├── MANIFEST.MF
/WEB-INF
  ├── web.xml (Servlet configuration)
  ├── classes/ (Compiled Java classes)
  ├── lib/ (JAR dependencies)
  ├── views/ (JSP files)
/index.html
```

```
/style.css
/script.js
```

## Deploying a WAR File

WAR files are deployed to **Java EE servers like Tomcat, JBoss, WebLogic, and WildFly**.

1. Place the WAR file in the `webapps/` directory of Tomcat.
2. Start the Tomcat server.
3. Access the application at `http://localhost:8080/myapp`.

## Creating a WAR File with Gradle

### 1️⃣ Apply the War Plugin

For **Gradle (Groovy DSL)**:

```
plugins {
    id 'war'
}

dependencies {
    implementation 'javax.servlet:javax.servlet-api:4.0.1'
}

war {
    archiveFileName = "myapp.war"
}
```

For **Gradle (Kotlin DSL)**:

```
plugins {
    war
}

dependencies {
    implementation("javax.servlet:javax.servlet-api:4.0.1")
}

tasks.war {
    archiveFileName.set("myapp.war")
}
```

### 2️⃣ Build the WAR File

```
gradle war
```

The WAR file will be created in `build/libs/`.

### 3️⃣ Deploy to Tomcat

Copy the `myapp.war` file into Tomcat's `webapps/` folder and restart Tomcat.

---

## 4. Key Differences: JAR vs WAR

| Feature | JAR (Java ARchive) | WAR (Web Application ARchive) |
|---|---|---|
| **Purpose** | Packages standalone applications or libraries | Packages web applications |
| **Deployment** | Runs using `java -jar` | Requires deployment on a servlet container |

| Feature | JAR (Java ARchive) | WAR (Web Application ARchive) |
|---|---|---|
| **Contains** | Java classes, resources, metadata | Java classes, resources + JSP, HTML, CSS, JS |
| **Entry Point** | `Main-Class` in `MANIFEST.MF` | `web.xml` or annotations for servlets |
| **Execution** | Runs independently | Runs inside a web server like Tomcat |
| **Build Tool Support** | Gradle, Maven | Gradle, Maven |

---

## 5. JAR vs WAR in Spring Boot

- **Spring Boot favors JAR** over WAR because it embeds Tomcat/Jetty inside the JAR.
- Spring Boot applications are built using:

```
gradle bootJar
```

- To build a WAR instead, set:

```
plugins {
    id 'war'
}
```

---

# Frequently Asked Interview Questions & Answers

## Basic Questions

## 1. What is the difference between a JAR and a WAR file?

✅ **Answer:**
A **JAR (Java ARchive)** is used for **packaging standalone Java applications and libraries**, whereas a **WAR (Web Application ARchive)** is used for **packaging web applications** that need to be deployed in a servlet container.

| Feature | JAR | WAR |
|---|---|---|
| **Usage** | Standalone Java apps, libraries | Web applications |
| **Contains** | Java classes, resources | Java classes + JSP, HTML, CSS, JS |
| **Execution** | `java -jar app.jar` | Deploy on a servlet container (Tomcat, Jetty) |
| **Entry Point** | `Main-Class` in `MANIFEST.MF` | `web.xml` or annotations |
| **Server Requirement** | No external server required | Requires a web server (Tomcat, WildFly, etc.) |

---

## 2. When should you use a JAR file instead of a WAR file?

✅ **Answer:**
Use a **JAR file** if:

- You are building a **standalone Java application**.
- You are packaging a **Java library** (e.g., a logging framework).
- You are developing a **Spring Boot microservice** (Spring Boot prefers JARs over WARs).

Use a **WAR file** if:

- You are developing a **traditional Java EE web application** that runs on a **servlet container** (e.g., Tomcat).
- Your application includes **JSP files, Servlets, or JSF pages**.

## 3. How do you create a JAR file using Gradle?

✅ **Answer:**
In **Gradle (Groovy DSL)**:

```
plugins {
    id 'java'
}

jar {
    manifest {
        attributes(
            'Main-Class': 'com.example.MainApp'
        )
    }
}
```

Run:

```
gradle jar
```

The JAR will be located in `build/libs/`.

---

## 4. How do you deploy a WAR file in Tomcat?

✅ **Answer:**

1. Build the WAR file using **Gradle**:

```
gradle war
```

The WAR file will be generated in `build/libs/`.

2. Copy the `myapp.war` file to the `webapps/` directory of Tomcat:

```
cp build/libs/myapp.war /path/to/tomcat/webapps/
```

3. Start Tomcat:

```
sh /path/to/tomcat/bin/startup.sh
```

4. Access the application at:

```
http://localhost:8080/myapp
```

---

# Advanced Questions

## 5. How does Spring Boot change the way JARs and WARs are used?

✅ **Answer:**
Spring Boot **prefers JAR over WAR** because:

- It **embeds Tomcat/Jetty** inside the JAR, making deployment easier.
- You can run it like a normal application:

```
java -jar myapp.jar
```

- **WAR files require a servlet container** (e.g., Tomcat), while Spring Boot's JARs **include the web server** inside them.

However, if you **still need a WAR**, Spring Boot supports it by changing the `build.gradle`:

```
plugins {
    id 'war'
}

dependencies {
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
}
```

---

## 6. Why are JARs preferred in microservices architecture?

✅ **Answer:**
JARs are preferred in **microservices** because:

- **Self-contained**: Each service includes all dependencies and an embedded server.
- **Easier to deploy**: No need for an external servlet container.
- **Better suited for containerization**: JARs can be directly used in **Docker/Kubernetes**.

Example Dockerfile for a Spring Boot JAR:

```
FROM openjdk:17
COPY myapp.jar /app.jar
CMD ["java", "-jar", "/app.jar"]
```

## 7. How do embedded servers (like Tomcat) impact JAR vs WAR choices?

✅ **Answer:**

- **Traditional WAR deployments** require an **external Tomcat** server.
- **Spring Boot JARs** embed Tomcat/Jetty, eliminating the need for an external server.
- This **simplifies deployment** because you don't need to configure the server separately.

---

## 8. Can a WAR file contain a JAR file inside it?

✅ **Answer:**
Yes! A WAR file **can contain JAR files** inside the `WEB-INF/lib/` folder.
Example structure:

```
/META-INF
/WEB-INF
  ├── web.xml
  ├── classes/ (compiled Java classes)
  ├── lib/ (JAR dependencies)
```

If your web app uses **Spring, Hibernate, or any external library**, their JARs are placed inside `WEB-INF/lib/`.

---

# Bonus: Practical Use Cases

| Scenario | JAR | WAR |
|---|---|---|
| Microservices (Spring Boot) | ✅ Yes | ❌ No |
| Traditional Java EE web apps | ❌ No | ✅ Yes |
| CLI-based applications | ✅ Yes | ❌ No |
| Requires Tomcat to be installed separately | ❌ No | ✅ Yes |

---

## Final Thoughts

1. **Use JARs** for standalone applications and modern **Spring Boot microservices**.
2. **Use WARs** if your application requires **traditional servlet containers**.
3. **Gradle makes it easy** to build both JAR and WAR files.

---