WAR (file format)

In software engineering, a **WAR** file (**W**eb **A**pplication **R**esource[1] or **W**eb application **AR**chive[2]) is a file used to distribute a collection of JAR-files, JavaServer Pages, Java Servlets, Java classes, XML files, tag libraries, static web pages (HTML and related files) and other resources that together constitute a web application.

Content and structure

Web ARchive	
Filename extension	`.war`
Internet media type	applicat archive
Magic number	`PK\x03\x` (standard Z file)
Developed by	Sun Micro
Container for	JSP, Java Servle
Extended from	JAR

A WAR file may be digitally signed in the same way as a JAR file in order to allow others to determine where the source code came from.

There are special files and directories within a WAR file:

- The /WEB-INF directory in the WAR file contains a file named web.xml which defines the structure of the web application. If the web application is only serving JSP files, the web.xml file is not strictly necessary. If the web application uses servlets, then the servlet container uses web.xml to ascertain to which servlet a URL request will be routed. The web.xml file is also used to define context variables which can be referenced within the servlets and it is used to define environmental dependencies which the deployer is expected to set up. An example of this is a dependency on a mail session used to send email. The servlet container is responsible for providing this service.

Advantages of WAR files

- Easy testing and deployment of web applications
- Easy identification of the version of the deployed application
- All Java EE containers support WAR files
- MVC structure supports WAR files.

Assuming production environments do not promote a fix without sufficient testing prior to deployment, a WAR file has a distinct advantage when properties files are used to identify environment specific variables. For example, an LDAP server in a testing environment may be something like `ldaps://testauth.example.com:636`. The LDAP server in a production environment is `ldaps://auth.example.com:636`. An external properties file would define the link with some thing like:

```
LINKED_PAGE=ldaps://testauth.example.com:636
```

The source code reads the property file to determine the target LDAP server. In this way, developers can be certain that the WAR file tested and verified is exactly the same as that which is being promoted to production.

Disadvantages of WAR files

Some consider web deployment using WAR files to be disadvantageous when minor changes to source code are required for dynamic environments. Each change to source code must be repackaged and deployed in development.[3] This does not require stopping the web server if configured for runtime deployment.[4]

Example

The following sample *web.xml* file demonstrates the declaration and association of a servlet:

```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun
Microsystems, Inc.//DTD Web
Application 2.2//EN"
```

```
"http://java.sun.com/j2ee/d
tds/web-app_2_2.dtd">

<web-app>
    <servlet>
        <servlet-
name>HelloServlet</servlet-
name>
        <servlet-
class>mypackage.HelloServle
t</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-
name>HelloServlet</servlet-
name>
        <url-
pattern>/HelloServlet</url-
pattern>
    </servlet-mapping>

    <resource-ref>
        <description>
            Resource
reference to a factory for
javax.mail.Session
            instances that
may be used for sending
electronic mail messages,
            preconfigured
to connect to the
```

```
appropriate SMTP server.
        </description>
        <res-ref-
name>mail/Session</res-ref-
name>
        <res-
type>javax.mail.Session</re
s-type>
        <res-
auth>Container</res-auth>
    </resource-ref>
</web-app>
```

The `/WEB-INF/classes` directory is on the ClassLoader's classpath. (The classpath consists of a list of locations from which `.class` files can be loaded and executed by the JVM.) The `/WEB-INF/classes` directory contains the classes associated with the web application itself.

Any JAR files placed in the `/WEB-INF/lib` directory will also be placed on the ClassLoader's classpath.

See also

- EAR (file format)
- JAR (file format)

References

1. *Crossley, Allistair. "Apache Tomcat 8 (8.0.44) - Tomcat Web Application Deployment" (https://tomcat.apache.org/tomcat-8.0-doc/deployer-howto.html) . Apache Software Foundation. Retrieved 2017-06-27.*
2. *Hunter, Jason (1999-10-15). "What's New in Java Servlet API 2.2?" (https://www.infoworld.com/article/2076518/what-s-new-in-java-servlet-api-2-2-.html) . JavaWorld. Retrieved 2020-11-08.*
3. *"Web Application Lifecycle" (https://docs.oracle.com/javaee/6/tutorial/doc/bnadu.html) . The Java EE 6 Tutorial. Oracle.*

4. *"Deploying on a running Tomcat server" (https://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html#Deploying_on_a_running_Tomcat_server) . Apache Software Foundation.*

External links

- Packaging Web Archives (https://eclipse-ee4j.github.io/jakartaee-tutorial/packaging003.html#BCGHAHGD) Archived (https://web.archive.org/web/20201106084953/https://eclipse-ee4j.github.io/jakartaee-tutorial/packaging003.html#BCGHAHGD) 2020-11-06 at the Wayback Machine (The Jakarta EE 8 Tutorial)

- JSR 154: JavaTM Servlet 2.4 Specification (https://jcp.org/en/jsr/detail?id=154)