WAR file i.e. Web Application Archive is a JAR file used to distribute a collection of JSP, Servlets, Java classes, XML files, tag libraries, static web pages (HTML and related files) and other resources that together constitute a web application.

**Structure of a WAR File - Servlet Deployment Environment**

*MyWebapp* is the project name and it is used to access the web application when it is deployed. It called as *Context Path* of the web application. The context path refers to everything in the URL after the server and port number.

A context path is used to access a web application. In our example, the project *MyWebapp* can be accessed via:

http://localhost:8080/MyWebapp

Context Path is also used to resolve the location of a resource.

**Public Resources:**

Everything outside WEB-INF and META-INF directories are public resources and can be accessed using context path via appropriate URL. In our example, *contacts.html* and *index.jsp* along with *images* directory are public resources. They can be accessed via:

http://localhost:8080/MyWebapp/contacts.html

HTML, JSP, CSS and images form the public resources of a web application. The placement of public resources is arbitrary as far as the specifications of Web applications are concerned. Application developer can logically separate these files by putting them inside optional *html* and *jsp* directories.

*index.jsp* is the default welcome page for the Web application. The *welcome page* is the Web page served up when you access the Web application

http://localhost:8080/MyWebapp/

If this Web page is not present, then, by default, *index.html* and *index.htm* are looked for and served. These welcome pages are subject to configuration and can be modified

**Private Resources:**

All the contents of WEB-INF and META-INF directories fall into the category of application's private resources, and cannot be accessed directly by client applications.

1. **WEB-INF:**

All the files in WEB-INF are protected against being requested by the web-container, i.e. they are invisible to the outside world. WEB-INF directory contains a deployment descriptor (web.xml) and 3 sub-directories:

* + classes
  + lib
  + tags

The WEB-INF directory itself is not supposed to be on the classpath. However, /WEB-INF/classes is on the classpath, and so are the JAR files in /WEB-INF/lib

Say if we have a directory called as foo inside WEB-INF and it contains a file called as bar.properties, then bar.properties will not be present in classpath.

The 'classes' Directory:

The *classes* directory contains servlet and utility classes, including JavaBeans. It may also contain a number of resource files such as key/value message lists, which contain error messages and user prompts for the application, and application-specific configuration information. Resource files may be used for application externalization / internationalization.

Each class is stored within a directory hierarchy that matches its fully qualified name (FQN). Therefore, a class DatabaseServlet with package structure com.wrox.db.DatabaseServlet will be stored in the classes/com/wrox/db directory structure. Because servlets are merely Java classes that implement a specified interface, they are stored in the *classes* directory, too.

The 'lib' Directory:

This directory contains packaged Java libraries ( .jar files) that the application requires and that are bundled with the application. JAR files that are placed here are available only to the Web application. If the libraries are to be accessed across Web applications, they should be placed under <TOMCAT\_HOME>/lib

The 'tags' Directory:

An optional tags directory within the WEB-INF directory contains configuration files for tag libraries. A *tag library* is a group of Java classes that define the functionality of dynamic markup tags.

The 'web.xml' File:

The Deployment Descriptor - web.xml is used by Java web applications as a deployment descriptor file to determine how URLs map to servlets, which URLs require authentication, and other information like filters, listeners, initialization parameters, container managed security constraints, resources and welcome pages.

**The contents of the WEB-INF directory are visible to servlet code using the *getResource()* and *getResourceAsStream()* method calls on the *ServletContext*, and may be exposed using the *RequestDispatcher* calls.**

This means that WEB-INF resources are accessible to the resource loader of your Web-Application and not directly visible for the public. This is why a lot of projects put their resources like JSP files, JARs / libraries and their own class files or property files or any other sensitive information in the WEB-INF folder. Otherwise they would be accessible by using a simple static URL

The web.xml 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. web.xml 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.

1. **META-INF:**

A Web application may have an optional META-INF directory that contains deployment information for tools that create war files and resources that applications may rely on. Therefore, a Servlet container will refuse to show the contents of the META-INF directory to a client. The META-INF directory can contain two configuration files:

* + The manifest file (MANIFEST.MF) and
  + context file (context.xml)

MANIFEST.MF:

The MANIFEST.MF file is an optional configuration file for a Web application. It contains a list of JAR files on which an application relies. The container can then use this to check for all the required libraries that are to be made available for the Web application. This provides the developer a facility to initiate a deploy time check by the container which ensures that the container can find all the classes your application depends on. This ensures that in case you missed a JAR, you don't have to wait till your application blows at runtime to realize that it's missing.

context.xml:

The context file (context.xml) contains the configuration for the Web applications’ Context. The context represents the web application.