Tomcat

is a web server and servlet container that is used to serve **Java web applications**. When you start Tomcat, it runs as a Java process and **listens** for incoming HTTP requests on a specified port (usually port 8080).
When a request comes in, Tomcat **processes** the request and **sends** a response back to the client. A Java web application typically consists of Java classes, HTML pages, CSS files, JavaScript files, and other resources. These files are **packaged into a single file called a WAR (Web ARchive) file**.
The WAR file is essentially a **compressed** archive that contains all the necessary files and folders needed to run the web application.

Deploy to Tomcat: Theroy

When you deploy a WAR file to Tomcat, Tomcat **extracts** the contents of the WAR file into a directory on the server's file system. By default, this directory is located at `$CATALINA_HOME/webapps/<war-file-name>/`, where $CATALINA_HOME is the installation directory of Tomcat and is the name of the WAR file (without the .war extension).

For example, if you deploy a web application called "myapp.war" to Tomcat, Tomcat will extract the contents of the WAR file to the directory $CATALINA_HOME/webapps/myapp/.

Tomcat then **creates a context** for the web application based on the contents of the WEB-INF/web.xml file. The context contains information about the web application, such as the URL mappings, servlets, filters, and other configuration information.

Deploy: Non Theory

1. **Build** your Java web application using a build tool such as Maven or Gradle. This will generate a WAR file in a directory such as "target".
2. **Copy** the WAR file to the "webapps" directory of your Tomcat installation, which is typically located at $CATALINA_HOME/webapps/. where $CATALINA_HOME is the installation directory of Tomcat.
   Tomcat will automatically deploy the web application by extracting its contents to a new directory under "webapps". The name of the new directory will be the same as the name of the WAR file (without the .war extension).

You can then access your web application by using the context path of the new directory in the URL.

Example

if you have a WAR file called "myapp.war", you can deploy it to Tomcat by copying it to the "webapps" directory. After the deployment, Tomcat will ex-

tract the contents of the WAR file to a new directory called “myapp” under “webapps”. You can then access your application by using the context path of “/myapp” in the URL.

Customize

If you want to customize the deployment settings of your web application, you can create a context XML file and place it in the “conf/Catalina/localhost” directory of your Tomcat installation. The context XML file should have the same name as the directory where your web application is deployed (e.g., “myapp.xml” for a web application deployed to “webapps/myapp”).

In the context XML file, you can define various settings for your web application, such as the context path, the database connection settings, and the security settings. Here is an example of a simple context XML file:

```
<Context path="/myapp">
    <Resource name="jdbc/mydb" auth="Container" type="javax.sql.DataSource"
              username="myuser" password="mypassword"
              driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://localhost/mydb"/>
</Context>
```

In this example, the context path of the web application is set to “/myapp”, and a database connection pool is defined with the JNDI name “jdbc/mydb”. You can place your own context XML file in the “conf/Catalina/localhost” directory to customize the deployment settings of your web application.

In Action

When a client sends a request for a URL that maps to the web application, Tomcat uses the context to determine how to handle the request. It might call a servlet to generate a response, or it might serve up a static file such as an HTML page or an image file.

In summary

Tomcat is a web server and servlet container that is used to serve Java web applications. * It extracts the contents of a WAR file into a directory on the server’s file system * creates a context for the web application based on the contents of the WEB-INF/web.xml file * uses the context to handle incoming HTTP requests.

CATALINA_BASE and CATALINA_HOME variables

If you are running **multiple** instances of Tomcat on a single host you should set + CATALINA_BASE: to be equal to the …/tomcat_instance1 or …/tomcat_instance2 directory as appropriate for each instance +

CATALINA_HOME: to the common Tomcat installation whose files will be shared between the two instances. The CATALINA_HOME environment variable should be set to the location of the root directory of the "binary" distribution of Tomcat.

The Tomcat startup scripts have some logic to set this variable automatically if it is absent, based on the location of the startup script in *nix and on the current directory in Windows. That logic might not work in all circumstances, so setting the variable explicitly is recommended.

The CATALINA_BASE environment variable specifies location of the root directory of the "active configuration" of Tomcat. It is optional. It defaults to be equal to CATALINA_HOME.

Using distinct values for the CATALINA_HOME and CATALINA_BASE variables is recommended to simplify further upgrades and maintenance. It is documented in the "Multiple Tomcat Instances" section below.

Folders Content

When running with a separate CATALINA_HOME and CATALINA_BASE, the files and directories are split as following:

In CATALINA_BASE:

bin - Only: setenv.sh (*nix) or setenv.bat (Windows), tomcat-juli.jar conf - Server configuration files (including server.xml) lib - Libraries and classes, as explained below logs - Log and output files webapps - Automatically loaded web applications work - Temporary working directories for web applications temp - Directory used by the JVM for temporary files> In CATALINA_HOME:

bin - Startup and shutdown scripts lib - Libraries and classes, as explained below endorsed - Libraries that override standard "Endorsed Standards". By default it's absent.