AN INTRODUCTION TO BACKEND FOR BEGINNERS

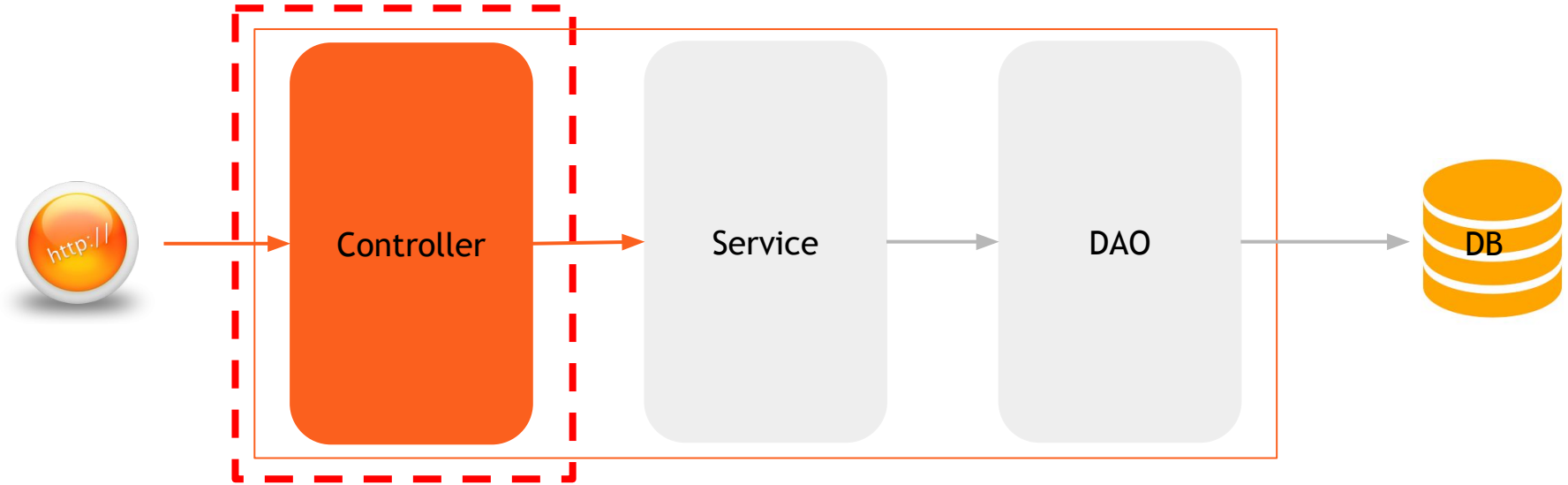
MODULO 1 – BACKEND OVERVIEW AND TOOLS

Matteo Colombo Speroni
Armando Esposito

Course Goal

*Learn how to **create** a simple application in **JAVA** environment to expose **REST** service for a Frontend application:*

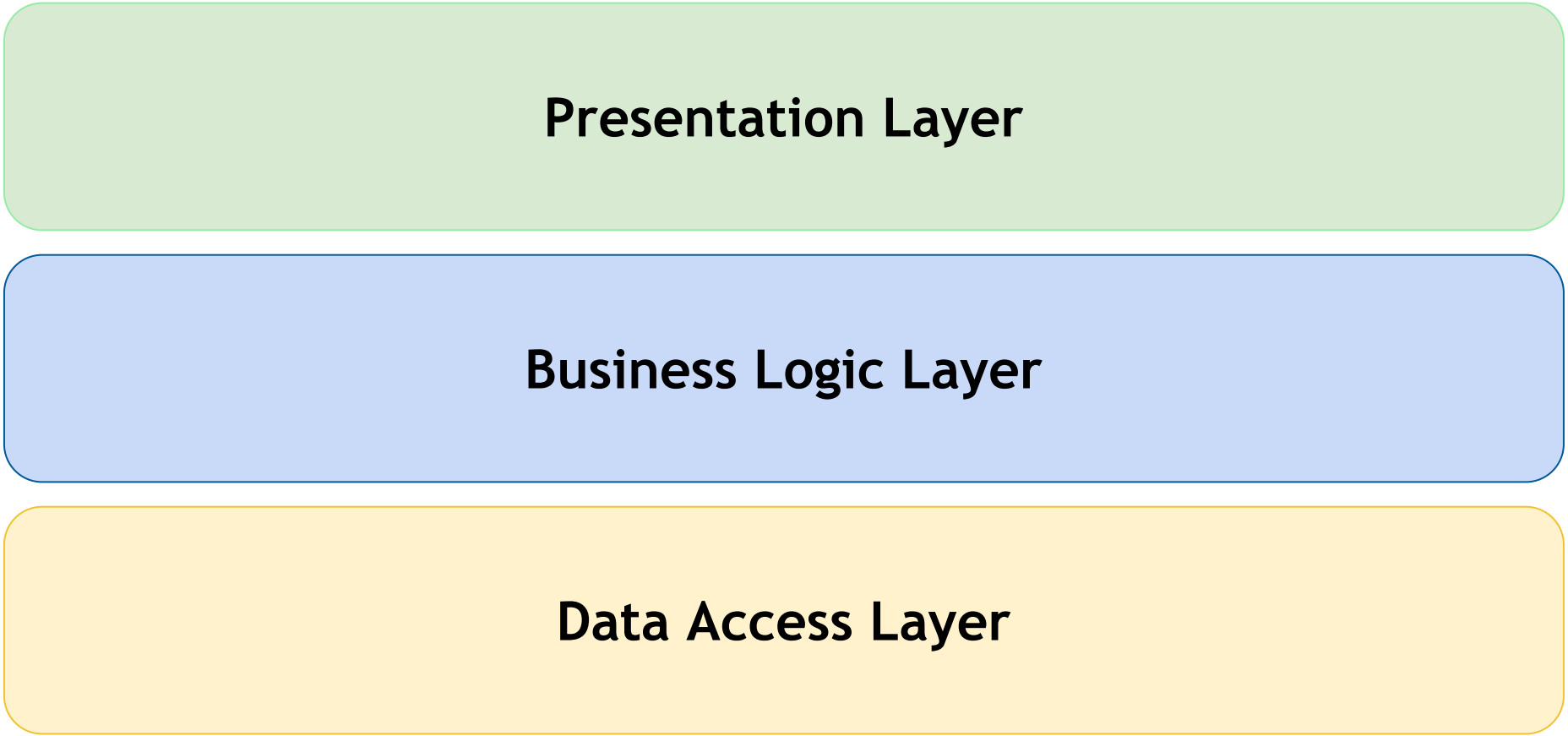
- High level overview of enterprise architecture
- Java Language basics
- Java Web basics
- Springs basics



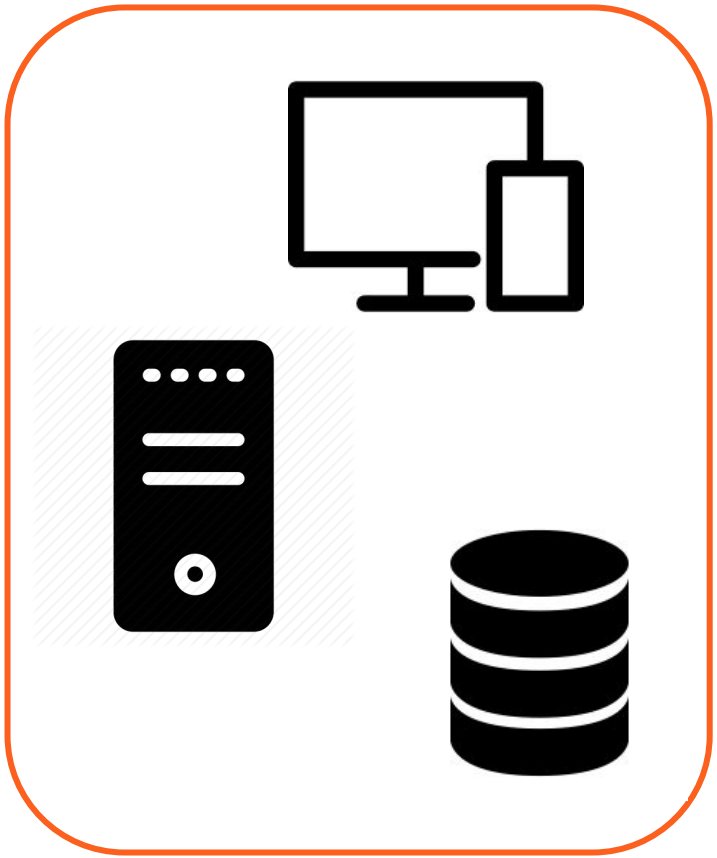
Agenda

DAY 1	EE Architecture and Tools
DAY 2	Java SE
DAY 3	Java EE
DAY 4	Spring Framework
DAY 5	Service and Persistence Layers
DAY 6	Q&A, Follow-Up and Exercises

Application layers organization



Architecture evolution - 1 Tier



Presentation, Business, Data Access layers in a single software package.

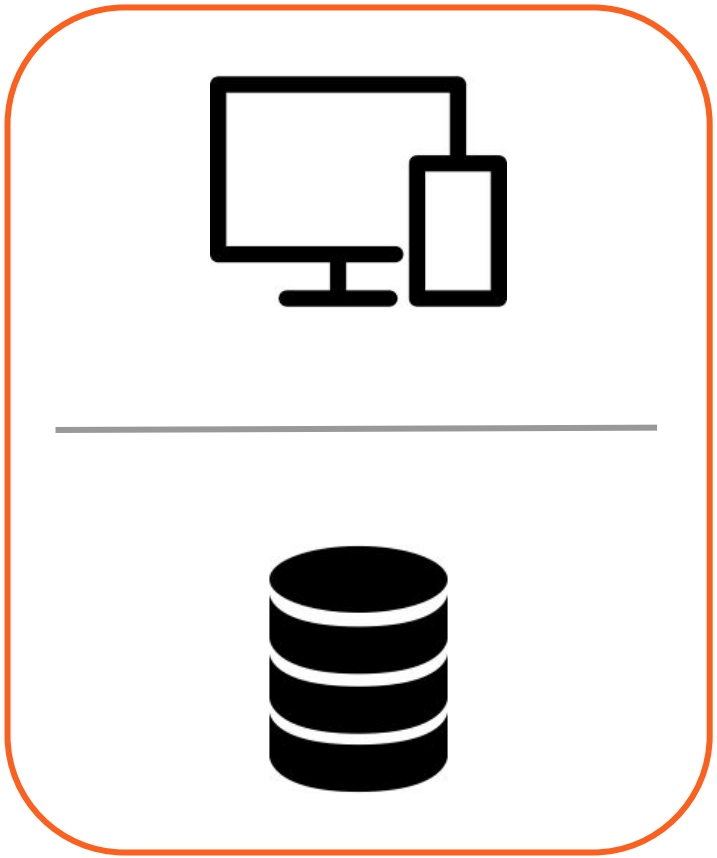
PROS:

- Simple and easier
- Direct access to everything
- Equivalent to a standalone application

CONS:

- Scalability
- Data are not shared

Architecture evolution - 2 Tier



Client system handles both Presentation and Application layers and Server system handles Database layer.

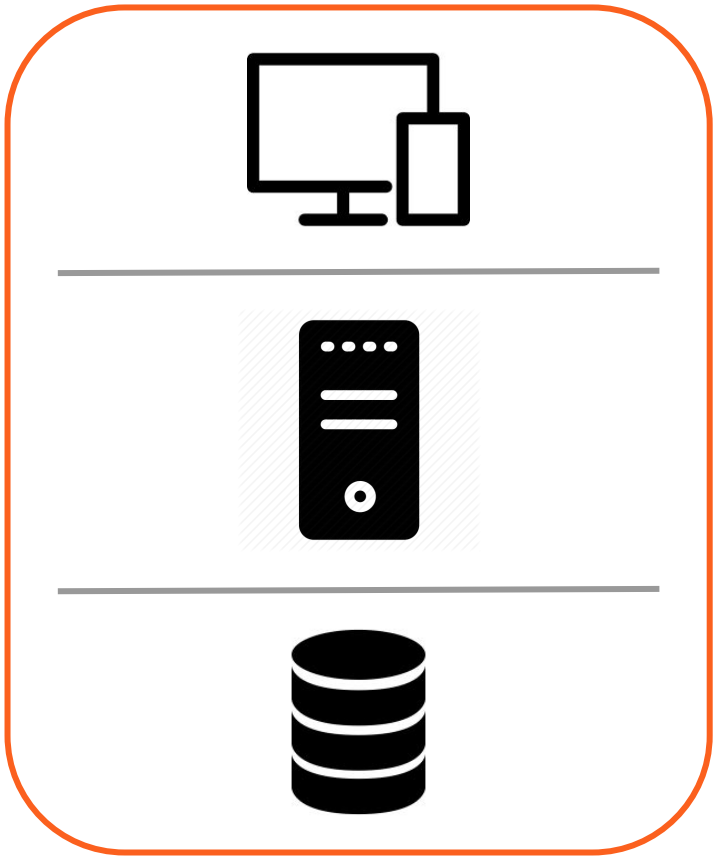
PROS:

- Basic Web model
- Improved scalability
- Data sharing

CONS:

- No decoupling between client and business logic

Architecture evolution - 3 Tier



Client system handles Presentation layer, Application server handles Application layer and Server system handles Database layer.

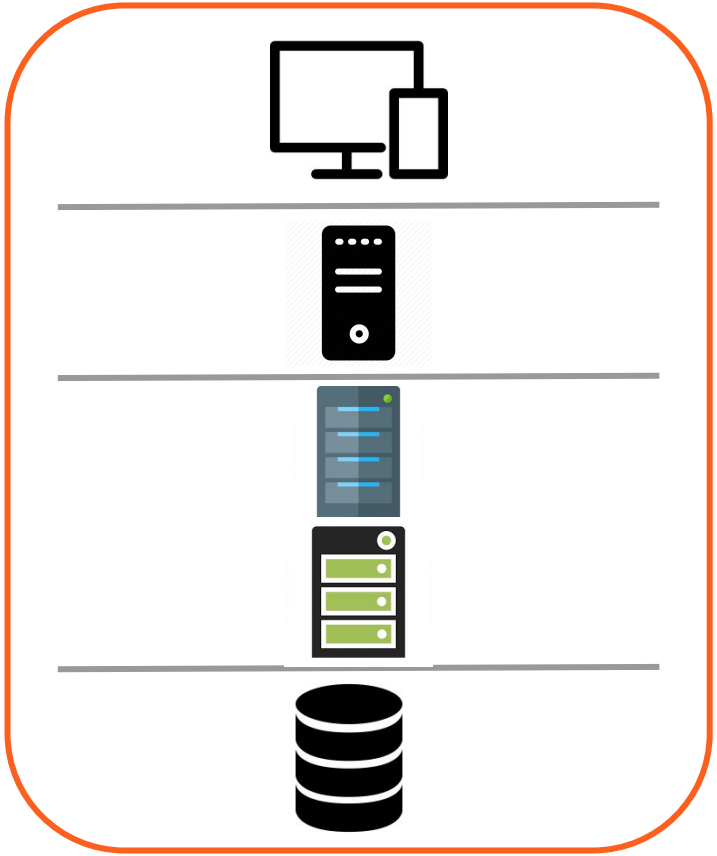
PROS:

- Scalability
- Reusability
- Data integrity
- Security

CONS:

- Complex structure, difficulty to set up and maintain

Architecture evolution - N Tier



Similar to three tier architecture but number of application servers are increased and represented in individual tiers in order to distributed the business logic so that the logic will be distributed.

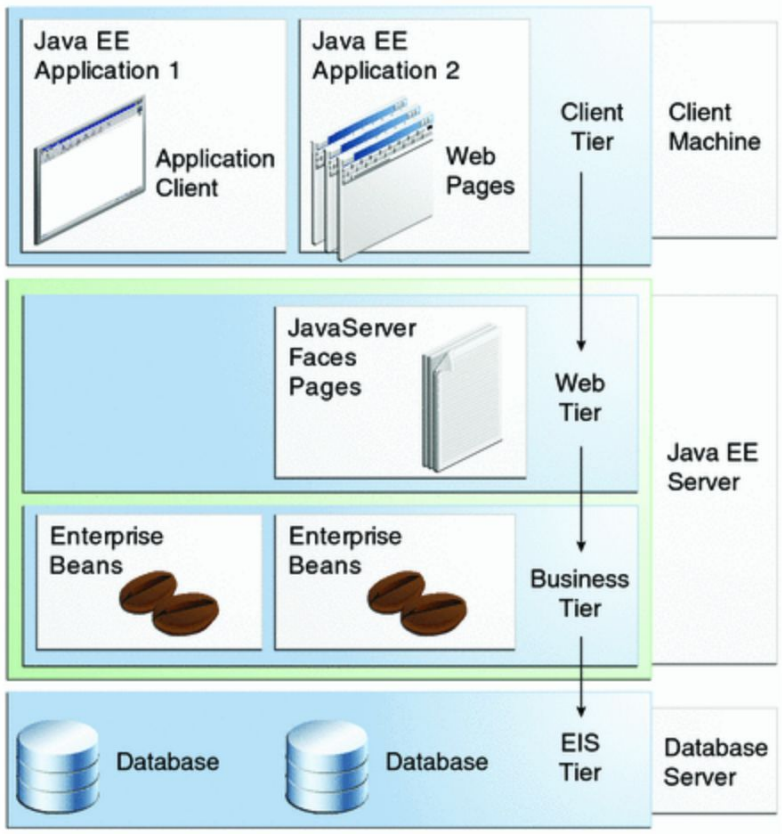
PROS:

- Single Responsibility
- Maintainability and Scalability
- Easy to change a single component

CONS:

- Very complex architecture model

JAVA EE Architecture Specifications



Java EE applications are made up of components. **A Java EE component is a self-contained functional software unit** and that communicates with other components:

Application clients and applets are components that run on the client (**Browser**).

Java Servlet, JavaServer Faces, and JavaServer Pages (**JSP**) technology components are web components that **run on the server**.

Enterprise JavaBeans (EJB) components (enterprise beans) are **business components** that run on the server.

JAVA EE - Container

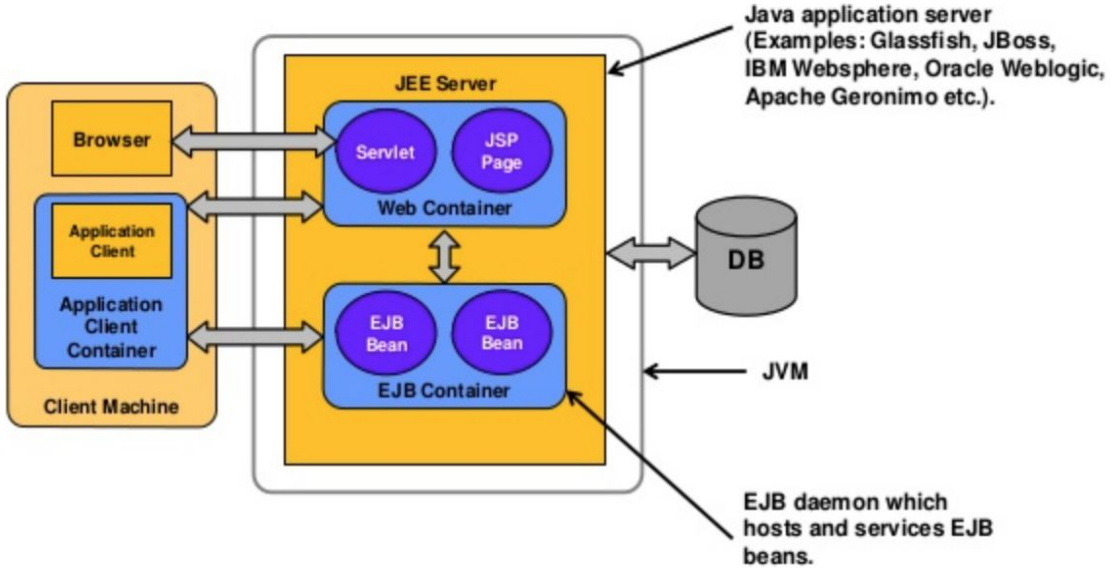
Interface between component and low level functionalities. Must respect JEE specifications.

Manage all runtime aspects:

- security
- lyfecycle
- transactions
- concurrency
- resources

EJB containers are provided only by JEE servers and manages execution of EJB providing

Web containers *manages execution of Servlets and JSPs, handling HTTP requests/Response*



Necessary Tools

What is Tomcat?

HTTP server

Modular

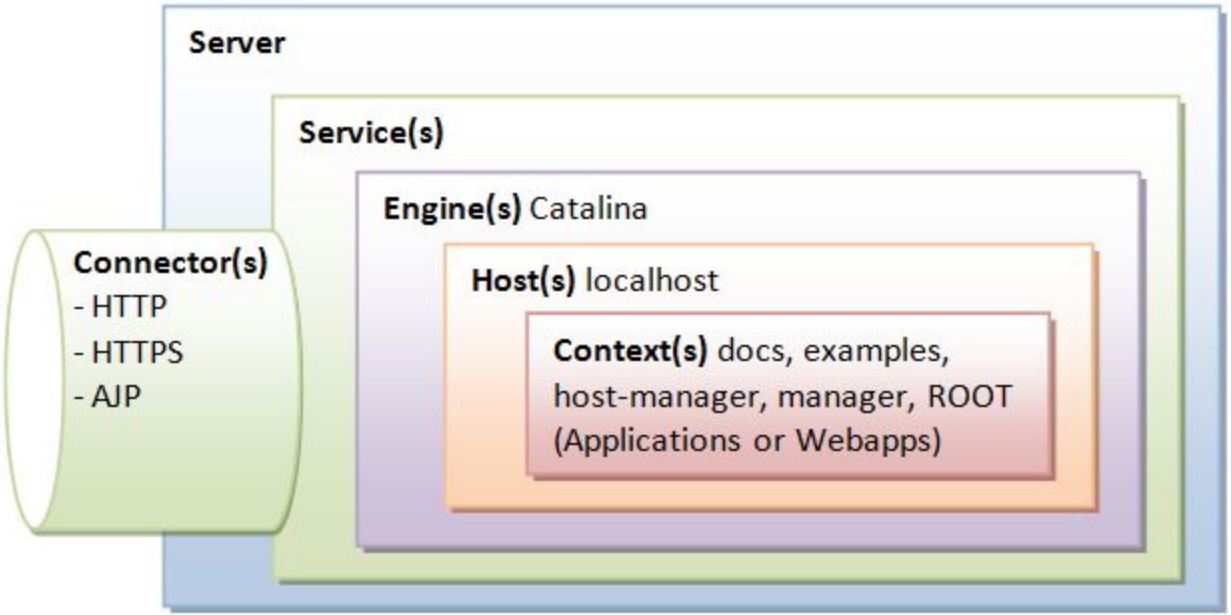
Servlet container

Open Source

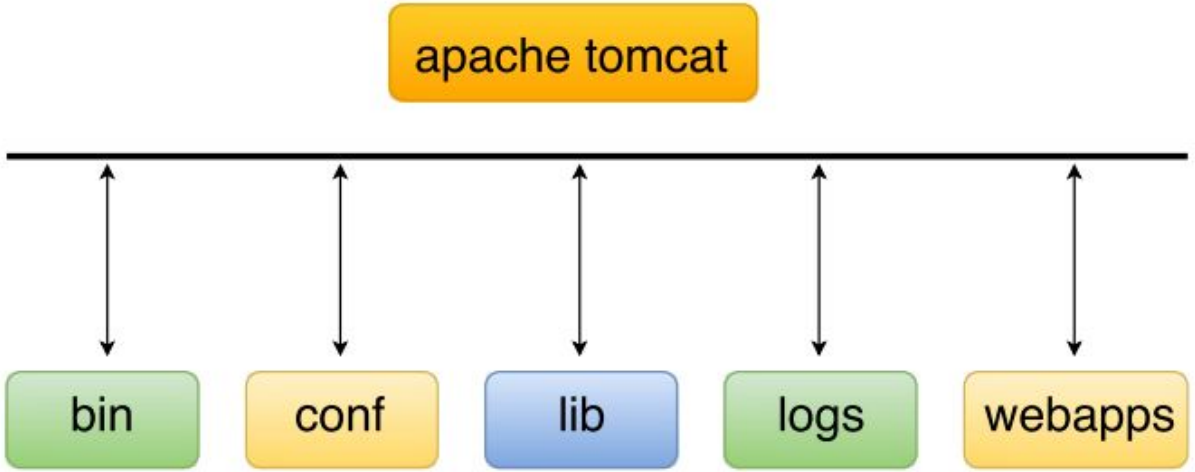
Hierarchical

Stable (since 1999)

Tomcat Container



Tomcat Structure



BIN - directory contains tomcat server and server **scripts** (start, stop, …)

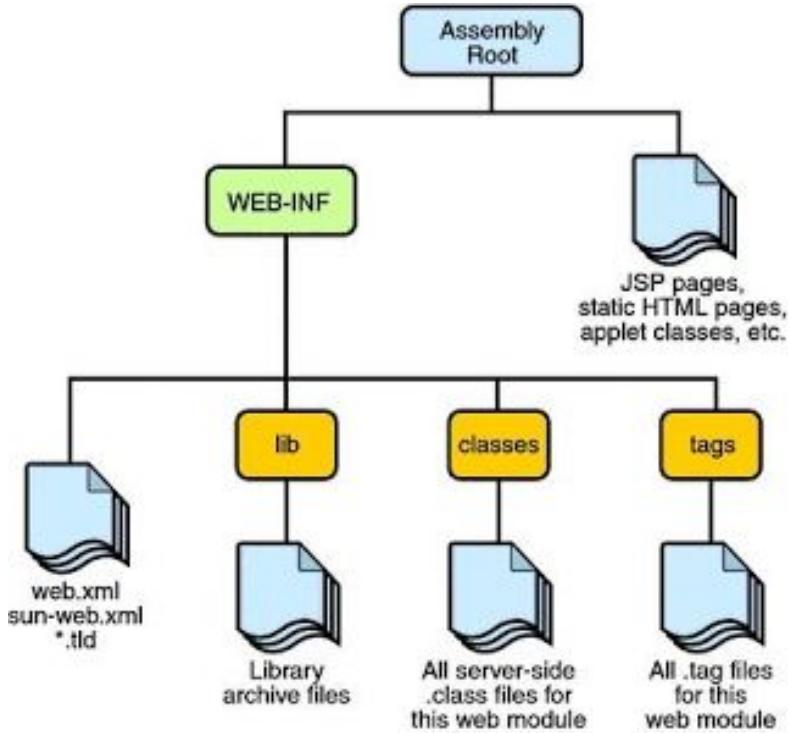
CONF - directory contains tomcat **configuration** files (server.xml, context.xml, …)

LIB - directory contains **library** files which used to run tomcat server

LOGS - directory contains **log files** all about your tomcat server

WEBAPPS - deployment folder where put distribution, which is same as htdocs in httpd server

Tomcat - Application distribution



Application Root - root directory of application and the default context. All **JSP, HTML and static** files/folders are stored here.

WEB-INF - contains all **application resources**, in particular **deployment descriptor**. The folder is **not public** and files can't be served directly.

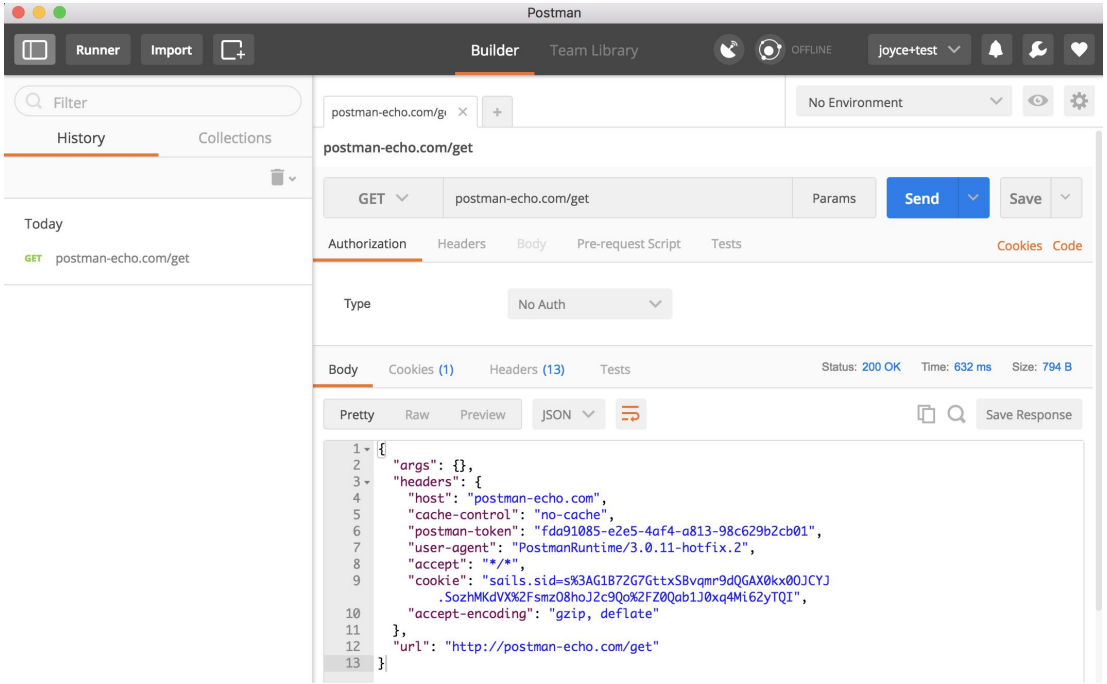
Classes - contains all **compiled sources**.

Lib - contains **dependency** libraries.

Web.xml - The **deployment descriptor** is an XML file and forms the **heart** of the Web Application.

What is Postman?

Web **browsers** tools are good **for web apps**, when working on an **HTTP API**, you need something more specific…



Postman Components

Requests

Configure url, parameters, headers, …
Replay requests
Authentication support (Baic, OAuth, …)

www.getpostman.com/docs/requests

Collections

Organize requests using folders
Synchronize across multiple devices
Export collections in JSON

www.getpostman.com/docs/collections

Environments

Information shared across multiple collections
Synchronize through multiple devices
Export environments in JSON

www.getpostman.com/docs/environments

Tests

Make assertions on API responses
Integrated libraries (Lodash, SugarJS, CryptoJS, …)
Use data files to set up multiple iterations

www.getpostman.com/docs/writing_tests

What is Maven?

Project management tool

Project Object Model (POM)

Plugin based

Common lifecycle

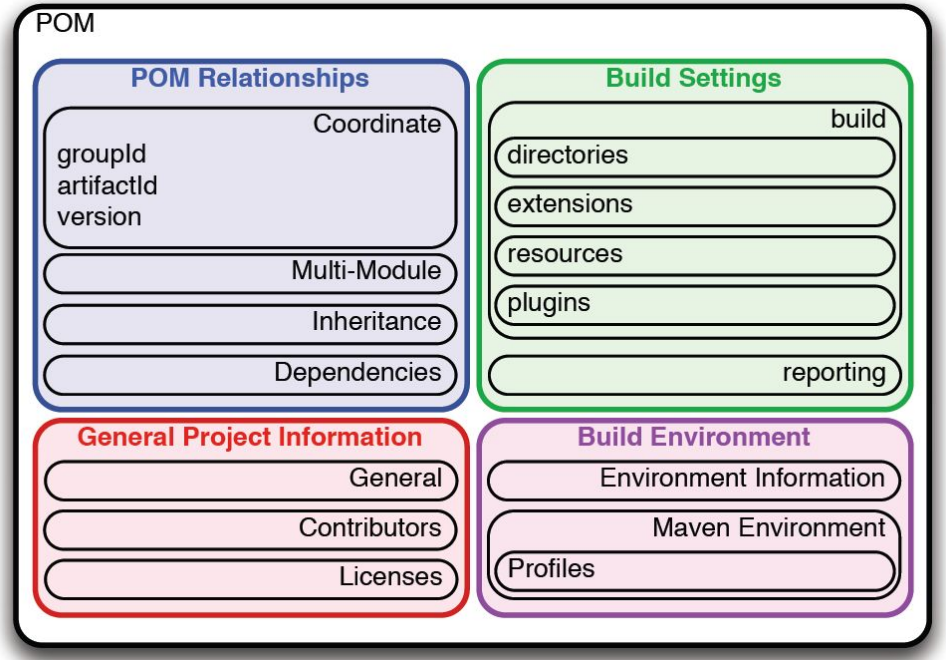
Convention over configuration

Dependencies management

Project Object Model

- **Analogous to** Makefile (C), build.xml (Ant), **package.json (NPM)**
- **Versioned** <major>.<minor>.incremental>-<qualifier>

  (*eg 1.0.0-SNAPSHOT*)
- **Packaging** (pom,jar,war,ear, …)
- **Inheritance**
- **Multi-module**
- **Dependencies**
- **Profiles**
- **Properties**

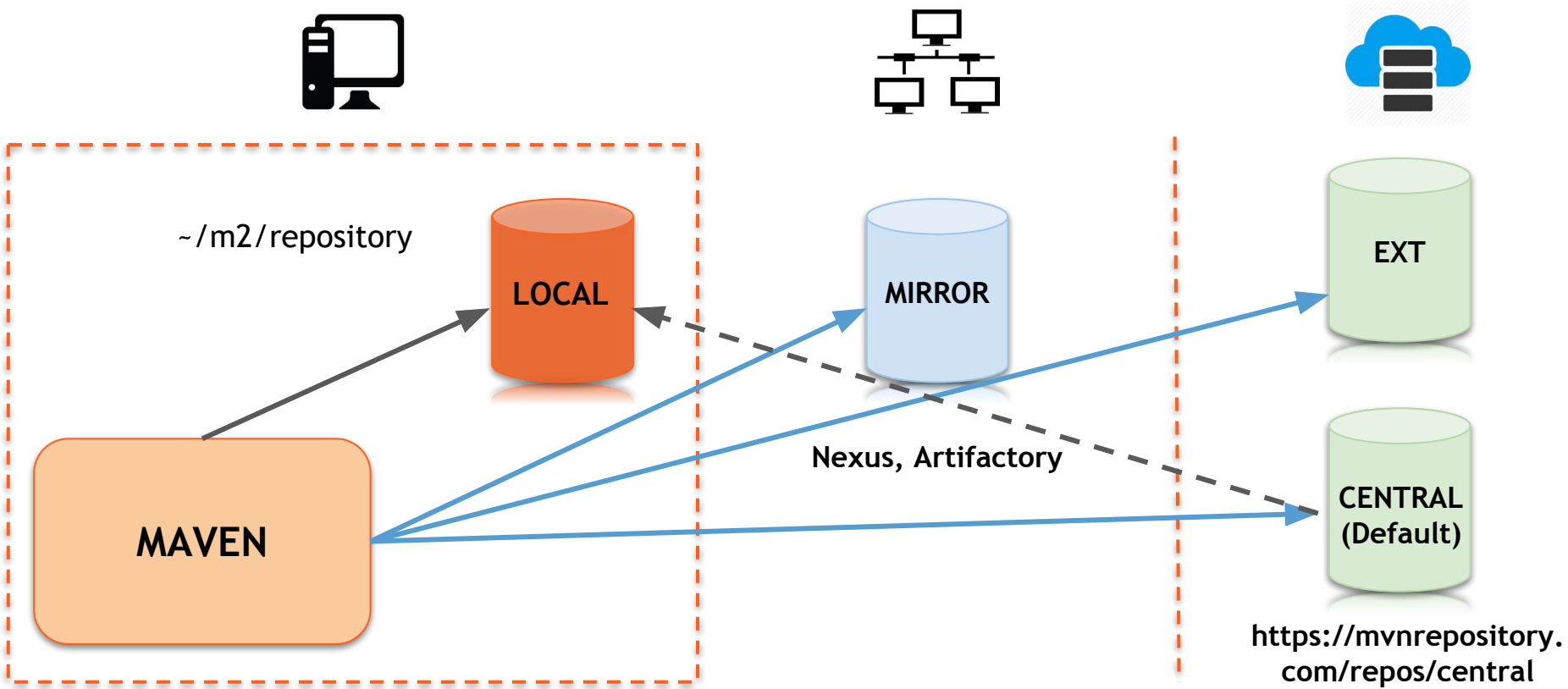


Keywords

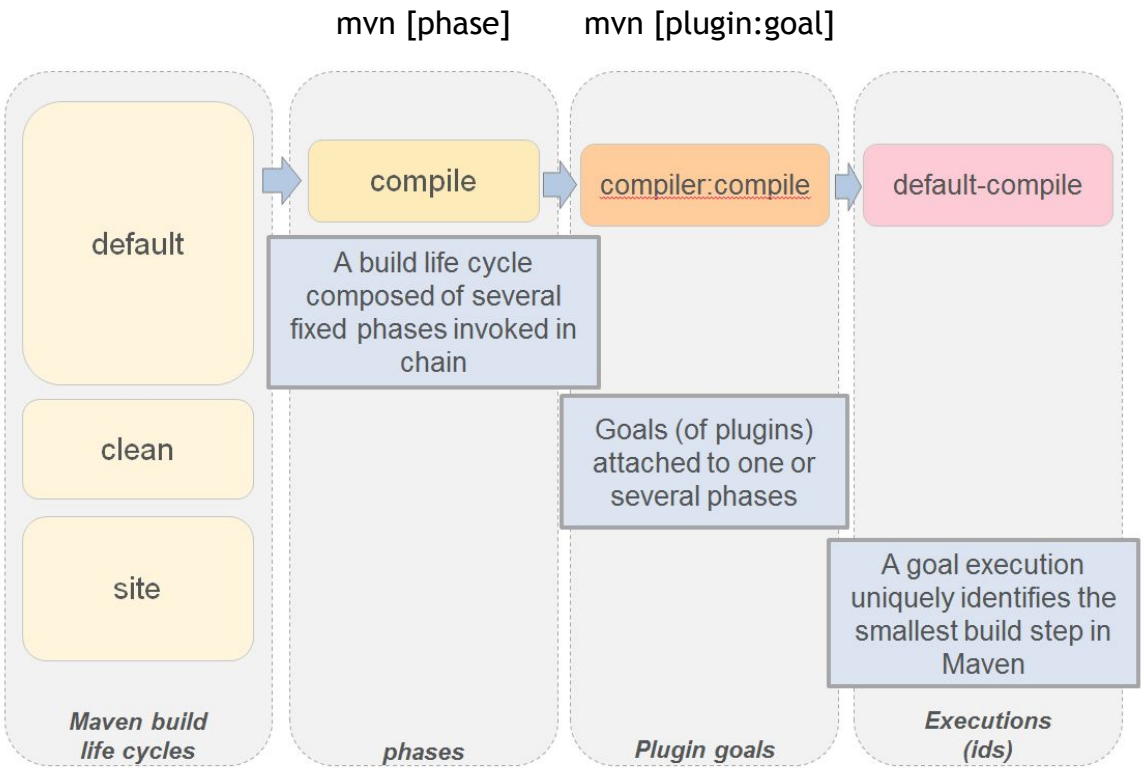
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- **modelVersion**: reference version of pom structure
- **groupId**: ID group (package)
- **artifactId**: ID of the artifact
- **version**: version of the artifact
- **packaging**: type of archive to be exported
- **dependency**: dependency definition to be contained inside **dependencies**
- **scope**:
  - compile
  - provided
  - runtime
  - test

Repository


~/m2/repository
LOCAL
MIRROR
EXT
MAVEN
Nexus, Artifactory
CENTRAL
(Default)
https://mvnrepository.
com/repos/central

Goals and Plugins and Phases

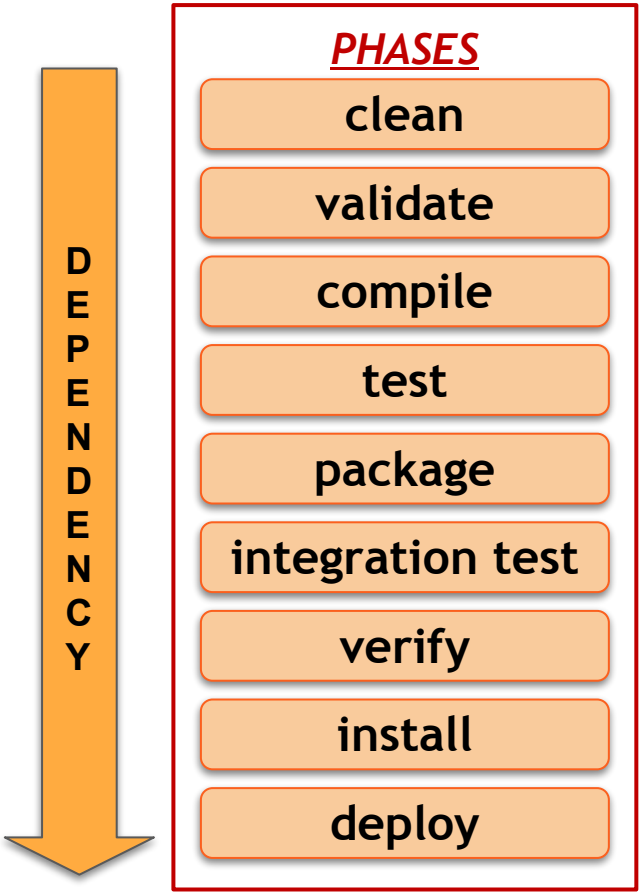


Phase: set of plugin - goal pairs.

Plugin: collection of goals reusable in all project. Maven provides build in plugins.

Goal: single task belonging to project or reusable

Build lifecycle



Clean target folder

Validate all necessary information for the project

Compile source code

Test compiled source code (run unit tests)

Package compiled sources and other resources

Deploy package for integration tests

Verify package against criteria

Install package into local repository

Deploy package into remote repository

Quickstart archetype

mvn archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -D**groupId**=com.mycompany.app \
  -D**artifactId**=my-app

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Useful Links

Architectures:
Develop n-Tier applications
Presentation Domain Data Layering - Martin Fowler
The evolution and future of software architecture - Mark Richards
JEE Tutorial - Oracle

Tomcat:
Tomcat 8 Official Docs

Maven:
Maven reference book - Sonatype
Maven Official Docs

Github project course repository and Lessons documentation:
https://github.com/mcolombosperoni/an-introduction-to-backend-for-beginners