JEE Basics

Udemy Course by Tim Buchalka

Table of Contents

[1. The Theory of Java EE 3](#_Toc89036697)

[1.1. What is it? 3](#_Toc89036698)

[1.2. What is an Application Server? 3](#_Toc89036699)

[1.3. What is a JSR? 3](#_Toc89036700)

[1.4. Reference Implementation 3](#_Toc89036701)

[1.5. Jakarta EE 4](#_Toc89036702)

[1.6. Java EE and Spring Framework 4](#_Toc89036703)

[2. Software to install 4](#_Toc89036704)

[2.1. Helloworld 4](#_Toc89036705)

[3. Todo App 4](#_Toc89036706)

[4. What is JEE so far? 6](#_Toc89036707)

# The Theory of Java EE

# What is it?

Collection of abstracs specifications that offer a solution for solving common challenges in the Developer's world. An annotation driven platform (Framework, like Java Spring).

* Abstract Specs
* Commonly Faced Challenges - Persistence, web services, transactions, security, lose coupling, etc.
* Focus is on abstract (Developers can focus on solving the challenge at hand, instead on building usual libraries, that are already in the JEE set)

# What is an Application Server?

* Concrete implementation - application server
* Abstract from detailed implementation
* javax.\* package

Examples of Java EE App Servers

* Payara server (Glassfish) - <https://payara.fish>
* IBM OpenLiberty - <https://openliberty.io>
* JBOSS Wildfly - <http://www.wildfly.org/>

# What is a JSR?

Java Specification Request

* Formal request to Java Community Process (JCP) (a body that standardizes APIs on the Java technology platform.
* (for) New proposal, enhancement to existing APIs eg CDI 1 -> CDI 2 (to the java platform)
* Used to group APIs into silos

Examples:

* <https://jcp.org/en/home/index>
* <https://jcp.org/en/jsr/platform>
* Link for the JEE Requests: <https://jcp.org/en/jsr/platform?listBy=3&listByType=platform>

# Reference Implementation

* Concrete realization of the abstract JSR
* Eg JAX-RS reference implementation - Jersez
* Java EE itself is a JSR!
* Java EE 8 is JSR 366 - RI Glassfish 5

Application server redefinition: Collection of the various reference implementations for the JSRs.

# Jakarta EE

(Bord when Oracle open up the development of JEE to the developers community)

* Java EE going forward
* Hosted by Eclipse Foundation
* https://jakarta.ee

# Java EE and Spring Framework

* Java EE influenced by Spring Framework
* Spring boot influenced by Java EE
* Both good platforms - use what works best and solves your problem
* No flame wars - quite pointless

Spring Framework (more simplified) both frameworks influence each other.

# Software to install

* Apache maven: <https://www.baeldung.com/install-maven-on-windows-linux-mac>
* Insomnia: <https://docs.insomnia.rest/insomnia/install>
* apache netbeans: <https://netbeans.apache.org/>
* git: <https://git-scm.com/downloads>
* payara: <https://www.payara.fish/>

# Helloworld

Dynamic Web App (Netbean Project hello-javaee8)

Deploy a web app using the payara.jar from the command line:

java -jar payara.jar --deploy /Users/lluis.carrasco.martinez-cic@ibm.com/NetBeansProjects/hello-javaee8/target/hello-javaee8.war --port 8080

See the app in the browser:

http://localhost:8080/hello-javaee8/resources/ping

# Todo App

Open NetBeans

* New Project
  + Maven Project from archetype
  + javaee8-essentials-archetype
  + Name the project "hello-todo"
  + Group Id: academy.learnprogramming
  + Package: academy.learnprogramming
  + Click on "Finish".
* In the created Project, in the "Project Files/pom.xml" file are listed the dependencies.
* To import the java library of a class in Netbeans press "Alt + enter" keys
* To automatically generate the Getters and Setters, go to the Menu:
  + Source
  + Insert Code
  + Getters and Setters
* The persistence.xml information is located in "Other Sources/src/main/resources/META-INF/"

Text

Description automatically generated

* Set tomcat: <https://www.dariawan.com/tutorials/java/how-to-add-apache-tomcat-server-in-netbeans-10/>
* Open a terminal in the project folder (or a command line) and prompt: mvn package (Maven will create our Web Archive of the application).
  + a <project-name>.war file will be created
* Using payara, like in the example before, we will deploy the application:
  + From the terminal in the folder where payara resides, prompt the next line to deploy the application: java -jar payara.jar --deploy [/Users/lluis.carrasco.martinez-cic@ibm.com/NetBeansProjects/hello-todo/target/hello-todo.war](mailto:/Users/lluis.carrasco.martinez-cic@ibm.com/NetBeansProjects/hello-todo/target/hello-todo.war) --port 8080
* Open the Insomnia Client to test.
* Prompt the following line in the folder where our payara.jar file resides to create an UberJar bundle: java -jar payara.jar --deploy [/Users/lluis.carrasco.martinez-cic@ibm.com/NetBeansProjects/hello-todo/target/hello-todo.war](mailto:/Users/lluis.carrasco.martinez-cic@ibm.com/NetBeansProjects/hello-todo/target/hello-todo.war) --port 8088 --outputUberJar helloTodo.jar
* Prompt to run the application: java -jar helloTodo.jar
* Another way to package the application is by using the concept of **maven profiles**. He is using the **payara-micro** package, however, he does not explains how to get it into the project, neither he passed the **hello-todo** maven configuration file. (Prompt: mvn package payara-micro:start)

# What is JEE so far?

Has 3 APIs that we should master:

* JPA (Java Persistence API, responsible for
  + Storing information in the relational DB
  + Retrieving information from the relational DB
* CDI (Context and Dependency Injection)
  + Dependency Injection API on the JEE platform
  + To create highly coupled applications
  + Act as a bridge between the JPA and the JAX-RS
* Jax-RS
  + To expose the resources over the HTTP protocol
  + Standardized REST resources for the end points
  + Expose the resources of the Web App to the world (in the internet)