**Sesión 1: “Configurando el taller”**

* Presentación del semillero
* Configuración de ambiente (se debe tener los prerrequisitos instalados)
* Justificación de cada herramienta a utilizar
* Introducción programación orientada a objetos
* Pruebas unitarias y seguimiento de código
* Introducción a la tecnología java
* Presentación del problema a resolver y establecer las historias de usuario a resolver

**Objetivos de la sesión:**

1. Validar configuración de ambiente
2. Entender cada herramienta y su rol dentro del desarrollo de una aplicación web empresarial
3. Primeros pasos con Pruebas unitarias

Desarrollo de la sesión

**1.Validar configuración de ambiente**

Seguir la guía de instalación paso a paso y hacer la validación de que el servidor de aplicaciones está arrancando correctamente

**2.Entender cada herramienta y su rol dentro del desarrollo de una aplicación**

- Maven: “Herramienta open-source, que se creó en 2001 con el objetivo de simplificar los procesos de build (compilar y generar ejecutables a partir del código fuente). Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.”

-JDK: “Java Development Kit (Herramientas de Desarrollo de Java). Sirve para construir programas usando el lenguaje de programación Java. Trae herramientas útiles como el compilador (javac), el desensamblador de binarios (javap), debugger, entre otras herramientas. Incluso, el JDK provee herramientas de evaluación de rendimiento de aplicaciones, como son VisualVM y Mission Control. Todo esto y más herramientas. Una instalación de JDK ya contiene un JRE dentro de las carpetas.”

-JRE: “Java Runtime Environment (Entorno de Ejecución de Java). Contiene a la JVM y otras herramientas que permiten la ejecución de las aplicaciones Java. JRE no posee compiladores ni herramientas para desarrollar las aplicaciones Java, solo posee las herramientas para ejecutarlas.”

-Wildfly: “Es un servidor de aplicaciones Java EE de código abierto implementado en Java puro, más concretamente la especificación Java EE. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java.”

- Eclipse: “Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.”

- Visual Studio Code: “Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.”

Oracle Database es un sistema de gestión de base de datos de tipo objeto-relaciona, Oracle, a partir de la versión 10g Release 2, cuenta con 7 ediciones y la que se va a utilizar en el semillero es Express Edition (XE) q es una edición gratuita.

-Nodejs: Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

-npm Node Package Manager o simplemente npm es un gestor de paquetes, donde se puede tener cualquier librería disponible con solo una línea de código, npm nos ayudará a administrar módulos, distribuir paquetes y agregar dependencias de una manera sencilla.

-angular-cli: es la herramienta de línea de comandos estándar para crear, depurar y publicar aplicaciones Angular.

-Github: plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

Comandos importantes:

mvn clean

mvn install

**3. Primeros pasos con Pruebas unitarias**

¿Que son las pruebas unitarias?

Es un método para conservar el estado del código, garantizar la cobertura del código y detectar errores y fallos antes de que lo hagan los clientes. Se deben ejecutar las pruebas unitarias con frecuencia para asegurarse de que el código funciona correctamente.

¿Qué se necesita para ejecutar una prueba unitaria?

Es requerido tener una metodología para poder ejecutar las pruebas de manera automática, existen diversos frameworks para efectos de semillero se va a usar tesntg.

¿Como empezar?

* Agregar la dependencia de testng en el proyecto ambiente\semillero-hbt\semillero-padre\semillero-servicios, para agregar una dependencia se necesita modificar el archivo pom.xml

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

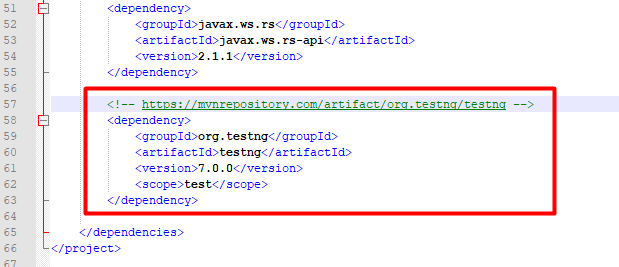
<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

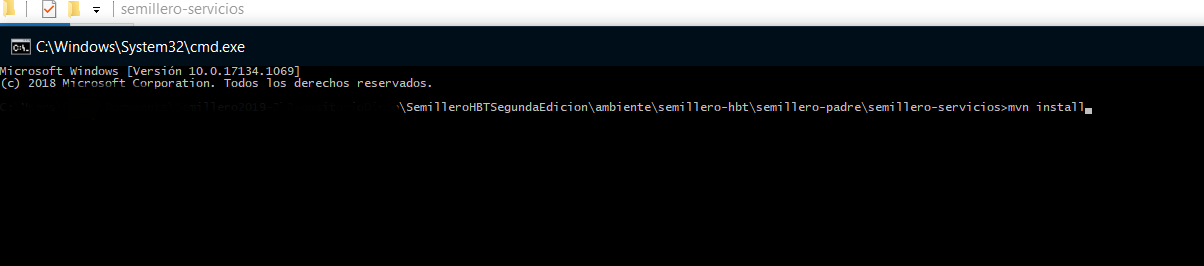
<version>7.0.0</version>

<scope>test</scope>

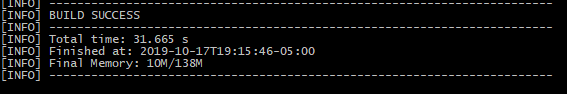
</dependency>



* Compilar el proyecto semillero-servicios para que la nueva dependencia sea actualizada, para hacer esto se debe abrir una consola de comandos en la ruta del proyecto a compilar en este caso es ambiente\semillero-hbt\semillero-padre\semillero-servicios y ejecutar el comando: mvn install



* Validar el estado de la ejecución del comando, esto se hace mirando la salida de la consola y ver si sale el “BUILD SUCCESS”

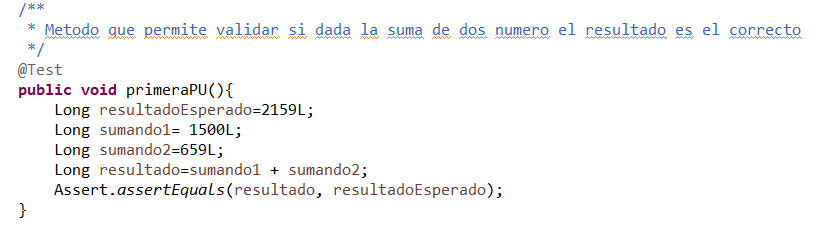


¿Qué artefactos se necesitan crear par ejecución de la prueba?

* Para las pruebas se requiere una clase que este configurada con una serie de anotaciones las cuales permite darle mas significado a la clase o los métodos para que sean procesados con el framework de testng.

¿Cómo empezar a hacer una clase de prueba?

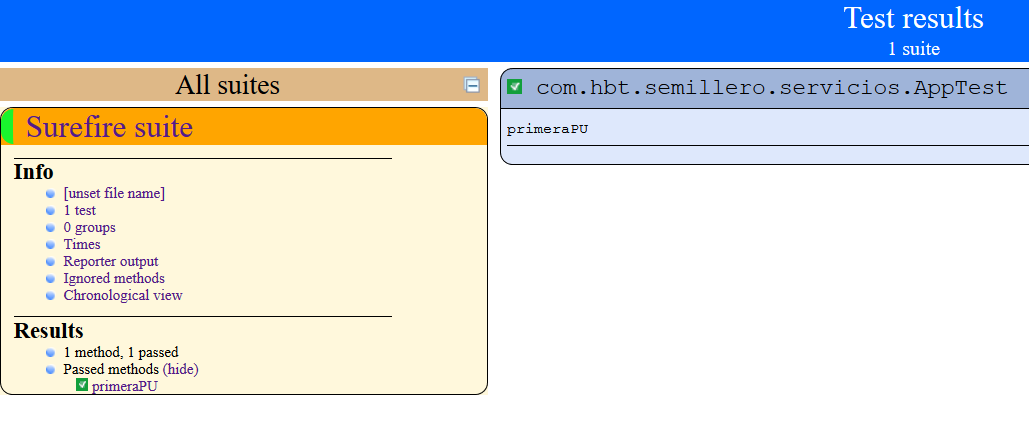
* Localizar el paquete en el cual se va a guardar la clase test, en este caso será en la ruta /semillero-servicios/src/test/java/com/hbt/semillero/servicios
* Crear la clase, para el ejercicio se va a modificar la clase AppTest.java, esta clase debe tener los imports correspondientes a testng osino no se podría ejecutar
* Adicionar los métodos de prueba, crear el método public void primeraPU(){…}
* Realizar las validaciones necesarias



* Compilar la clase y ejecutar la prueba unitaria, ubicarse por la consola de comandos (cmd) en la ruta del proyecto /semillero-servicios y ejecutar el comando

“mvn test -Dmaven.test.skip=false”, por defecto las pruebas unitarias están desactivadas para su ejecución automática es por esto que se debe agregar el -Dmaven.test.skip=false

* Ver los resultados de la prueba



¿Qué vamos a probar?

* En el ejercicio del semillero se va a necesitar realizar un CRUD lo que es el proceso completo de manipulación de determinada información en este caso un comic el cual va a ser representado en una estructura de datos llamada ComicDTO, como primer paso del desarrollo se deben crear las firmas de los métodos de prueba.

Ejercicio sesión 1

Dado el siguiente método en la clase AppTest.java realizar todos los casos de prueba requeridos realizando un Método encargado de validar que se invierte la cadena correctamente:

private String invertirCadena(String cadena) {

String cadenaInvertida = "";

for (int x = cadena.length() - 1; x >= 0; x--) {

cadenaInvertida = cadenaInvertida + cadena.charAt(x);

}

return cadenaInvertida;

}

¿Y que pasa si se envía la cadena "Yo soy"?, ¿Qué se debe tener en cuenta para validar que la cadena enviada es la misma que la cadena retornada por el método invertirCadena()?

**Conceptos adicionales**

* Un framework: “entorno de trabajo​ o marco de trabajo2​ es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.”
* “Anotación Java es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución o de compilación.”
* “Un Paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa y que por lo general tiene una funcionalidad y elementos comunes, definiendo la ubicación de dichas clases en un directorio de estructura jerárquica.”
* Pom.xml Project Object Model, es un fichero XML, que es la “unidad” principal de un proyecto Maven. Contiene información acerca del proyecto, fuentes, test, dependencias, plugins, versión y otros datos de administración para Maven.

**Enlaces de interés**

* Sitio oficial testng https://testng.org/doc/