**Sesión 4: “Preparando el modelo de datos”**

* Definición del modelo de datos
* Definición de las tablas y las entidades
* Definición de clases y relaciones

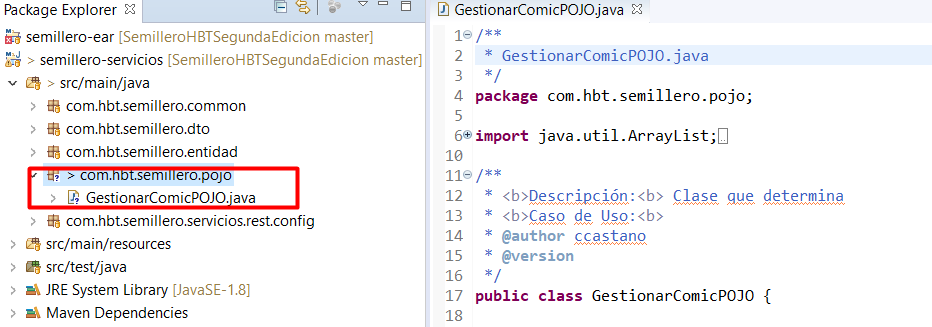
**Objetivos de la sesión:**

1. Realizar un POJO para poder manipular las estructuras de datos creados.
2. Realizando lógica de negocio

**Desarrollo de la sesión**

1. Realizar un POJO para poder manipular las estructuras de datos creados.

Un pojo es una clase java sencilla que no implementa ni extiende ninguna otra clase. En el paquete com.hbt.semillero.pojo crear la clase GestionarComicPOJO.java



Se requiere definir una estructura de datos para almacenar los comics en memoria para esto se requiere utilizar una lista que permite ir almacenando los comics en memoria, esto se hace de la siguiente manera:

private List<ComicDTO> listaComics = null; //Agregar los get y set de la lista

La anterior sentencia define el modificador de acceso private que determina que solo los métodos de la misma clase pueden usar la lista. También determina que es un java.util.List que va a contener solo objetos de tipo ComicDTO y hace su correspondiente inicialización.

Se debe crear el método public void crearComicDTO(){} con el objetivo de llenar uno a uno los datos relacionados con el comic

ComicDTO comic = new ComicDTO();

comic.setId("100");

comic.setNombre("EL EJÉRCITO DEL CAPITÁN AMÉRICA");

comic.setEditorial("Panini Comics ");

comic.setTematica("AVENTURA");

comic.setColeccion("BIBLIOTECA MARVEL ");

comic.setNumeroPaginas(128);

comic.setPrecio(new BigDecimal(50));

comic.setAutores("Phillippe Briones, Roger Stern ");

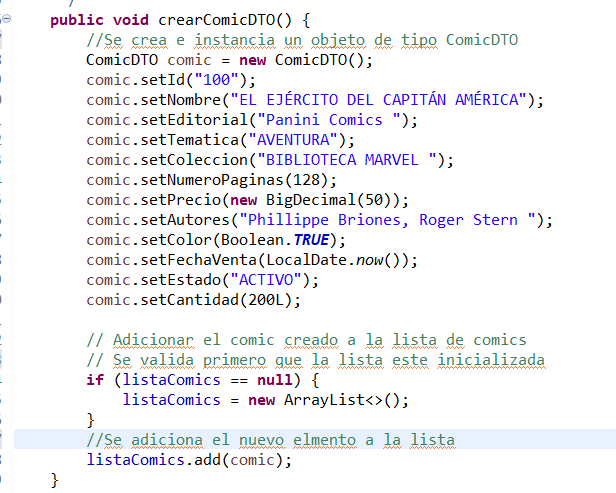
comic.setColor(Boolean.TRUE);

comic.setFechaVenta(LocalDate.now());

comic.setEstado("ACTIVO");

comic.setCantidad(200L);

Después adicionar el nuevo comic a la lista listaComics



¿Como se puede modificar el nombre a un comic ya almacenado en la listaComics?

Crear el método: public void modificarComicDTO(ComicDTO comicModificar) {}

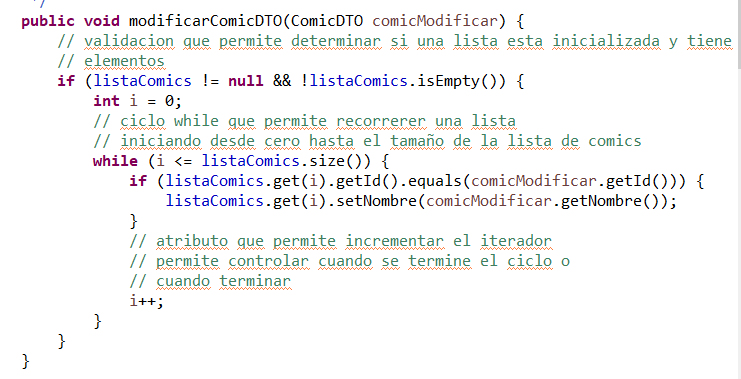
Validar que la lista tenga elementos

Utilizar un ciclo que permita recorrer los elementos de listaComics (while)

Validar que el elemento a modificar sea el elemento que se está iterando

Realizar la modificación

Salir del ciclo.



¿Como se puede eliminar un comic ya almacenado en la listaComics?

Crear el método: public void eliminarComicDTO(ComicDTO comicEliminar) {}

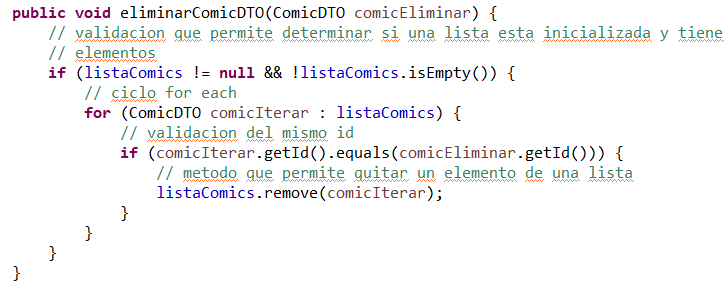
Validar que la lista tenga elementos

Utilizar un ciclo que permita recorrer los elementos de listaComics (for each)

Validar que el elemento a eliminar sea el elemento que se está iterando

Realizar la eliminación

Salir del ciclo.



¿Como se pueden ver toda la información un comic ya almacenado en la listaComics?

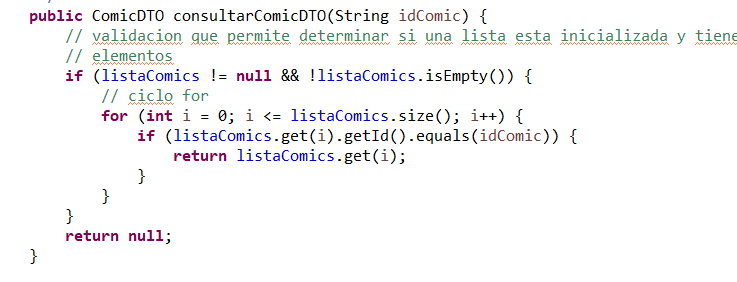
Crear el método: public ComicDTO consultarComicDTO(String idComic) {}

Validar que la lista tenga elementos

Utilizar un ciclo que permita recorrer los elementos de listaComics (for)

Validar que el elemento a consultar sea el elemento que tenga el mismo idComic que se ingresó por parámetros

Retornar el elemento



Salir del ciclo

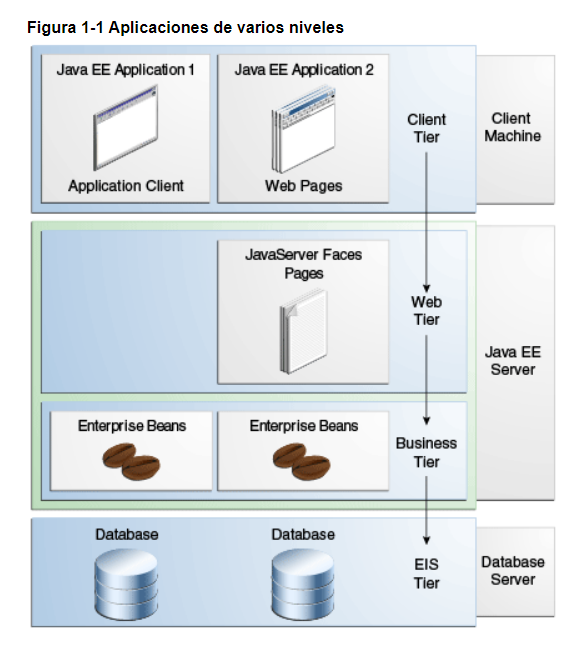
1. Primera aproximación a JEE (Java Platform, Enterprise Edition o Java EE ()

* Entendiendo los componentes de JEE (antes J2EE) que nos van a ayudar a crear lógica de negocio, como primera medida se debe entender lo siguiente:

El objetivo de la plataforma Java EE es proporcionar a los desarrolladores un potente conjunto de API (interfaz de programación de aplicaciones) al tiempo que acorta el tiempo de desarrollo, reduce la complejidad de la aplicación y mejora el rendimiento de la aplicación.

La plataforma Java EE utiliza un modelo de programación simplificado. Los descriptores de implementación XML son opcionales. En cambio, un desarrollador puede simplemente ingresar la información como una anotación directamente en un archivo fuente Java, y el servidor Java EE configurará el componente durante el despliegue y el tiempo de ejecución. Estas anotaciones generalmente se utilizan para incrustar en un programa datos que de otro modo se proporcionarían en un descriptor de implementación. Con anotaciones, coloca la información de especificación en su código al lado del elemento del programa afectado.

En la plataforma Java EE, la inyección de dependencia se puede aplicar a todos los recursos que un componente necesita, ocultando efectivamente la creación y búsqueda de recursos del código de la aplicación. La inyección de dependencia se puede utilizar en contenedores Enterprise JavaBeans (EJB), contenedores web y clientes de aplicaciones. La inyección de dependencia permite que el contenedor Java EE inserte automáticamente referencias a otros componentes o recursos necesarios, mediante anotaciones.



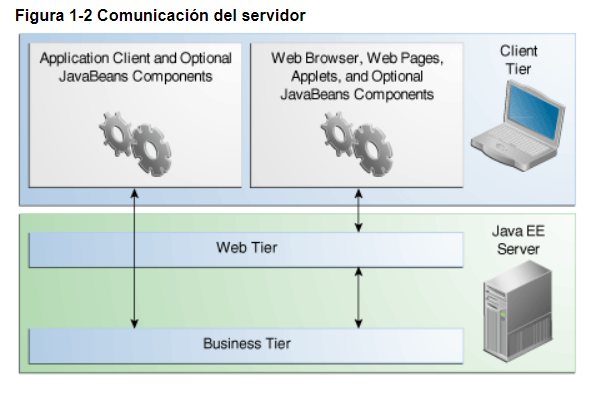
Componentes Java EE

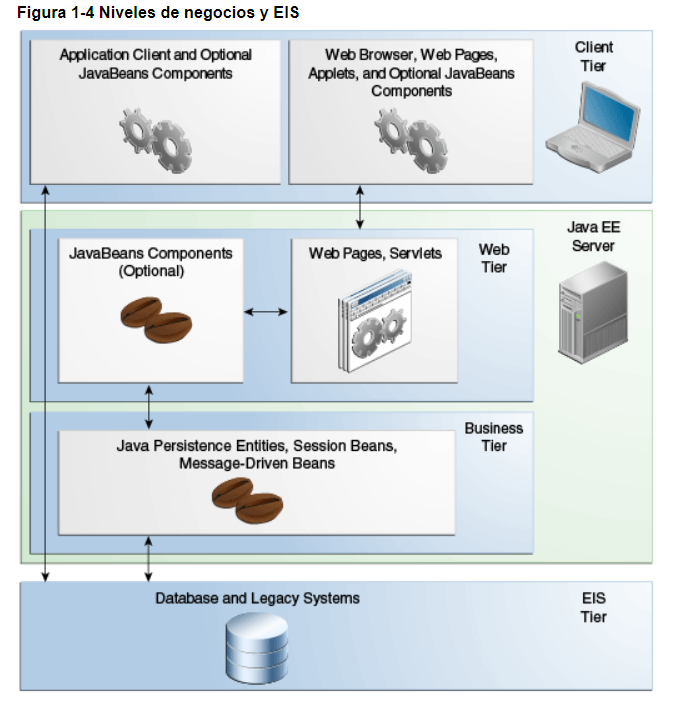
Las aplicaciones Java EE están formadas por componentes. Un componente Java EE es una unidad de software funcional autónoma que se ensambla en una aplicación Java EE con sus clases y archivos relacionados y que se comunica con otros componentes.

La especificación Java EE define los siguientes componentes Java EE:

* Los clientes de aplicaciones y los applets son componentes que se ejecutan en el cliente.
* Los componentes de tecnología Java Servlet, JavaServer Faces y JavaServer Pages (JSP) son componentes web que se ejecutan en el servidor.
* Los componentes EJB (beans empresariales) son componentes empresariales que se ejecutan en el servidor.

Los componentes Java EE están escritos en el lenguaje de programación Java y se compilan de la misma manera que cualquier programa en el lenguaje. Las diferencias entre los componentes Java EE y las clases Java "estándar" son que los componentes Java EE se ensamblan en una aplicación Java EE, se verifica que están bien formados y cumplen con la especificación Java EE, y se implementan en producción, donde son ejecutados y administrados por el servidor Java EE.





* Enfocarnos en componentes EJB

Un contenedor de aplicaciones JEE (JBoss, Glassfish, ..) puede controlar el ciclo de vida de una serie de clases java que lleven las anotaciones adecuadas. Entre estas clases, conocidas como Enterprise Java Beans o simplemente EJB, están los session beans, en concreto, Singleton, Statefull y Stateless.

Un Stateless bean es una clase java sin estado, es decir, no debería tener atributos, sólo métodos a los que se pueda llamar de forma independiente. Para que el contenedor de aplicaciones lo maneje, únicamente hay que ponerle la anotación @Stateless.

¿Qué quiere decir que lo manejará el contenedor de aplicaciones? En el caso de Stateless beann, el contenedor de aplicaciones creará varias instancias de estas clases (se encargará de hacer los new) y le pasará estas instancias a quien las necesite. Como son clases sin estado, el contenedor dará cualquiera de las instancias que tiene al que la necesite y no se preocupará de entregar siempre la misma al mismo.

Statefull bean

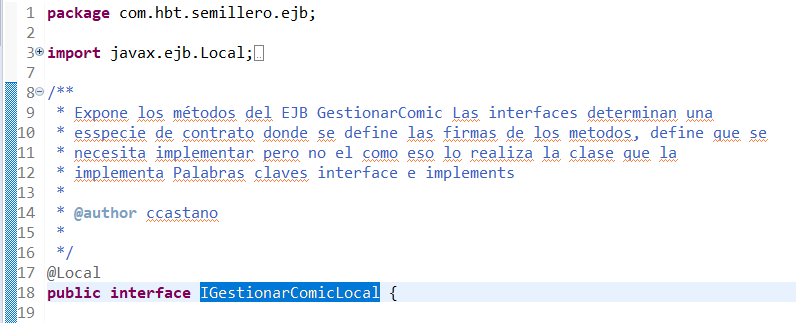
Un Statefull bean no es más que una clase en la que sí hay atributos cuyo valor es importante y debe conservarse entre llamadas a métodos. El contenedor de aplicaciones instanciará, al igual que con los Stateless bean, un número determinado de ellos y los reaprovechará. La diferencia es que el contenedor de aplicaciones se asegurará de dar siempre al mismo cliente/usuario la misma instancia. De esta forma, ese cliente/usuario puede guardar en él datos que sean particulares de él y que necesiten conservarse mientras está trabajando con la aplicación. Un ejemplo típico es el carrito de la compra de una web.

Singleton bean

Un Singleton bean es una clase Java que el contenedor de aplicaciones instanciará una única vez y será compartida por todo el mundo. Está pensada para guardar los datos que son comunes a todos los usuarios o clientes de nuestra aplicación.

* Realizando lógica de negocio

Crear una interface local de negocio, en el paquete “com.hbt.semillero.ejb” y crear la interface IGestionarComicLocal.java en donde se van a definir que métodos de negocio se requieren para resolver la gestión del comic



Solo se requieren las firmas de los métodos

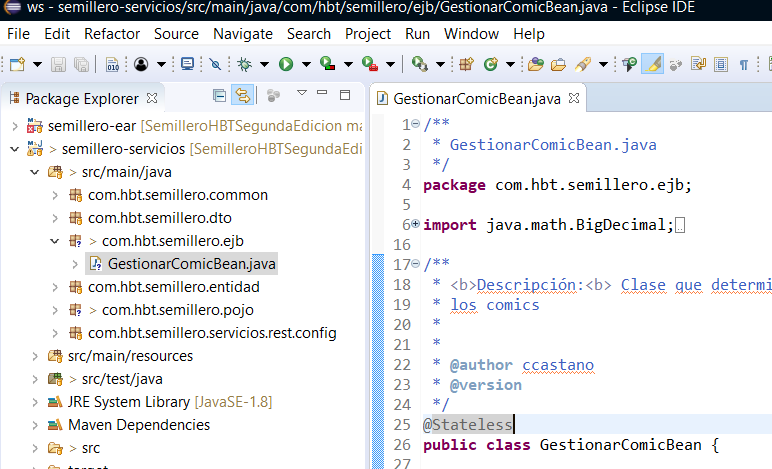
public void crearComic(ComicDTO comicNuevo);

public void modificarComic(Comic comicModificar);

public void eliminarComic(Comic comicEliminar);

public ComicDTO consultarComic(String idComic);

Crear un bean de negocio @Stateless en el mismo paquete GestionarComicBean.java que implemente la interface IGestionarComicLocal



Se requiere una forma de interactuar con la entidad Comic y que permita hacer el manejo de esta. Para ello se requiere crear un objeto javax.persistence.EntityManager

@PersistenceContext

private EntityManager em;

¿Como se puede crear un comic y almacenarlo en la Base de datos?

¿Como se puede modificar el nombre a un comic ya almacenado en la Base de datos?

¿Como se puede eliminar un comic ya almacenado en la Base de datos?

¿Como se puede obtener los comics ya almacenado en la Base de datos?

**Ejercicio sesión 3**

* Crear una clase de prueba llamada GestionarComicPOJOTest.java se deben crear dos comics más usados para guardarlos en la listaComics usando la información de ejemplo que se tiene en la tabla COMIC
* Realizar los casos de prueba para la clase GestionarComicPOJO.java para verificar que se agregan los 3 comic, se modifica el comic con id 2 y se elimina el comic con id 1.
* Lectura “Conceptos básicos de java.docx”

**Conceptos adicionales**

* Inyección de dependencias (en inglés Dependency Injection, DI) “es un [patrón de diseño](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o) orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos. Esos objetos cumplen contratos que necesitan nuestras clases para poder funcionar (de ahí el concepto de dependencia). Nuestras clases no crean los objetos que necesitan, sino que se los suministra otra clase 'contenedora' que inyectará la implementación deseada a nuestro contrato”.
* API interfaz de programación de aplicaciones “​ es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción”.
* JEE es una plataforma de programación —parte de la Plataforma Java— para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process (JCP), Java EE es también considerado informalmente como un estándar debido a que los proveedores deben de cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por JCP.

**Enlaces de interés**

* Documento “Rest vs WebServices.pdf”
* https://javaee.github.io/tutorial/toc.html