Laboratorio de programación avanzada

ModeloVistaControlador

El modelo vista-controlador separa el código sus distintas responsabilidades mediante capas. Cada capa se encarga de hacer una tarea muy concreta, lo que ofrece distintos beneficios. Se usa principalmente en sistemas donde se requiere el uso de interfaces de usuario.

Su fundamento es la separación de código en tres capas diferentes en lo que se llama Modelos, Vistas y Controladores.

Modelos: es la capa donde se trabaja con los datos.

Vistas: contiene el código de la aplicación que va a producir la visualización de las interfaces de usuario.

Controladores: contiene el código necesario para responder a las acciones que se solicitan en la aplicación. Es una capa que sirve de enlace entre las vistas y los modelos.

Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java. 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 sus elementos. Viene con objetivos predefinidos para realizar ciertas tareas, como la compilación y su empaquetado, simplificando así la build del código.

Maven está listo para ser usar en red.

Tomcat

Es un software desarrollado con Java que sirve como servidor web con soporte de serverlets y JSPs.

JavaServlet

Es una clase de Java utilizada para ampliar las capacidades de un servidor. Es un software que permite que un servidor web maneje contenido web dinámico basado en Java utilizando el protocolo HTTP o XML.

JDBC

JBDC (JavaDataBaseConnector) significa conectividad de bases de datos de Java, que es un API estándar de Java utilizada para la base de datos independiente entre el lenguaje de programación de Java y la base de datos. Nos permite, mediante ciertas librerías, conectarnos a la base de datos de forma nativa.

ORM

Pretende ser un puente entre los objetos y la base de datos. Dice qué objeto va a una tabla y que campo de un objeto va a la base de datos.

```java
Customer customer = new Customer();

customer.setId(1L);
customer.setName("Oscar Blancarte");
customer.setStatus(Status.ACTIVE);

manager.persist(customer);
```

Al usar un ORM no tenemos que pasar por toda la sintaxis de SQL. Al hacer el persist de customer, este objeto se mete en la base de datos.

Con las etiquetas le decimos a JPA que la clase va a ser una entidad, entonces será usada en la base de datos. Convertimos una clase normal en una entidad.

```java
@Entity
@Table(name = "customers")
public class Customer {

  @Id
  @Column(name="id")
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(name="name", length = 500, nullable = false )
  private String name;

  @Enumerated(EnumType.STRING)
  @Column(name = "status", length = 10, nullable = false)
  private Status status;
}
```

JPA

Java Persistence API o JPA es el standard de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos. Estandariza como los ORM deben trabajar con Java para la persistencia de objetos.

JPA es solo una especificación, por lo que existen diversas especificaciones. Hibernate, EclipseLink son implementaciones de JPA.

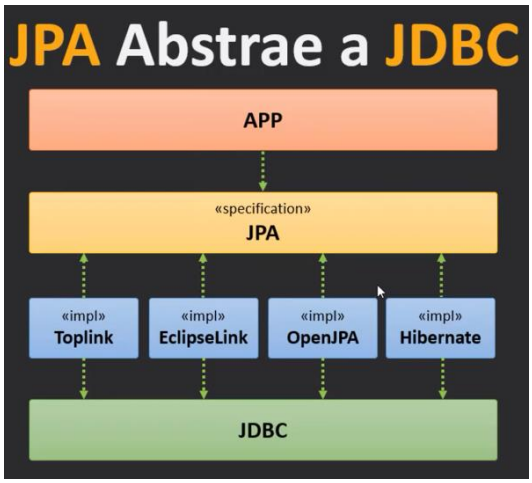
Persistence.xml: es un fichero que se encarga de conectarnos a la base de datos y define el conjunto de entidades que vamos a gestionar.

EntityManagerFactory: con los conceptos del persistence.xml se genera un objeto EntityManagerFactory que se encargará de gestionar todas estas entidades para la base de datos.

EntityManager: una vez dispuesto el ENF, este será capaz de construir un objeto de tipo EntityManager que gestiona un conjunto de entidades o objetos.

PersistenceContext: hace referencia a los objetos que han sido manipulados por el EntityManager y se encuentran bajo su control.

JPA Y JDBC



JPA abstrae JDBC

La aplicación solomente conoce JPA (anotaciones, clases, interfaces).

JPA abstrae las implementaciones, independiemente de la que usemos.

Los implementadores internamente utilizan JDBC para lograr la conexión a la base de datos.

Ventajas de JPA:

-Nos permite desarrollar mucho más rápido (trabaja en la parte de arriba)

-Trabajamos con Entidades (no se usan las bases de datos)

-Elimina muchos errores en tiempo de ejecución

-Mas legible y entendible (al trabajar con clases)

Desventajas de JPA:

-No es tan potente al ejecutar querys nativos

-Puede tener menor rendimiento

Ventajas de JDBC

-Tiene mayor rendimiento (al estar por debajo es más preciso al realizar las operaciones)

-Mayor versatilidad al usar los comandos

Desventajas:

-Requiere mas dedicación a la hora de desarrollarlo

-Se producen mas errores (el compilador no puede saber si los querys están bien)

POM

Project Object Model es un archivo XML que contiene la información del proyecto y detalles de configuración usado por Maven para construir el proyecto. Define cosas como:

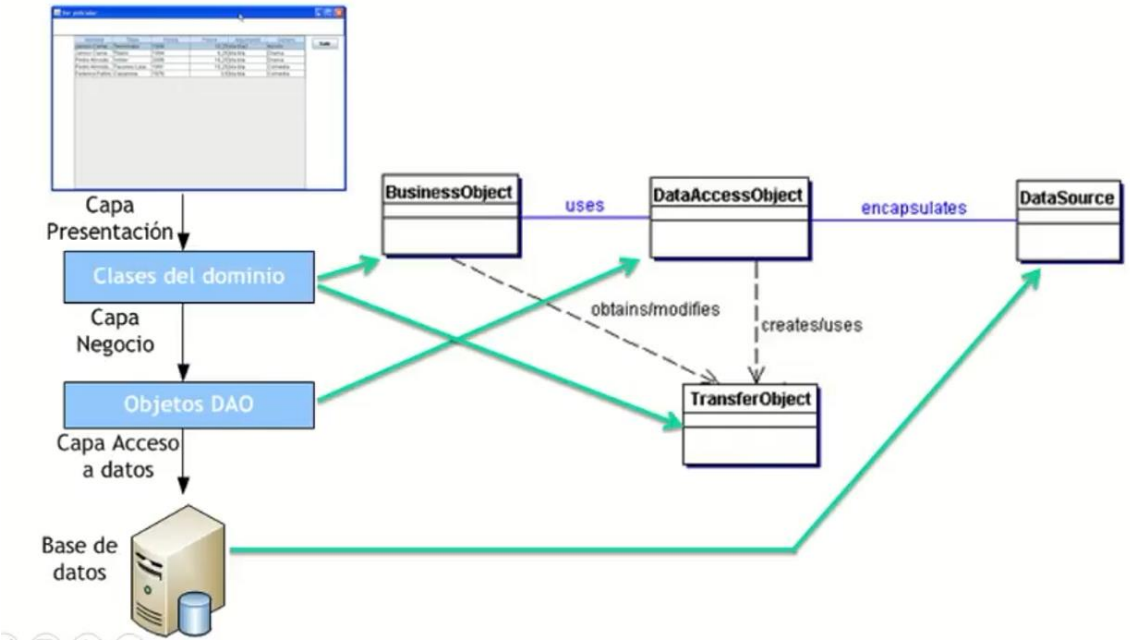
- La descripción de un proyecto
- Nombre, versión, tipos, localización de código y dependencias
- Plug-ins
- Perfiles

Patrón DAO

El patrón DAO (DataAccessObject) es un patrón que nos proporciona un objeto de acceso a datos para mantener la persistencia de datos dentro de una aplicación de software. Lo que busca es que la capa de persistencia sea independiente de la capa de modelo. De esta forma cuando codificamos los modelos no nos preocupamos de que forma vamos a aplicar la persistencia en ellos.

Qué es el patron DAO:

Es un componente software que actúa como una pasarela (como un puente de implementación) entre los datos que se van a almacenar en la capa de persistencia y la capa de lógica de un sistema. Suele ofrecer métodos para añadir, actualizar, buscar y borrar elementos.



BusinessObject: pertenece a la clase del dominio y es aquel objeto de negocio sobre el que nos interesa almacenar dentro de la capa de persistencia.

Objeto DAO: encapsula el acceso a la base datos. Utilizara un objeto de transferencia como contenedor para mover los datos.

Objeto de negocio: delega todas las operaciones de lectura y escritura en el objeto de acceso a datos.

Objeto de transferencia: nos permite acceder a la base de datos.