Sistemas Distribuidos e Internet

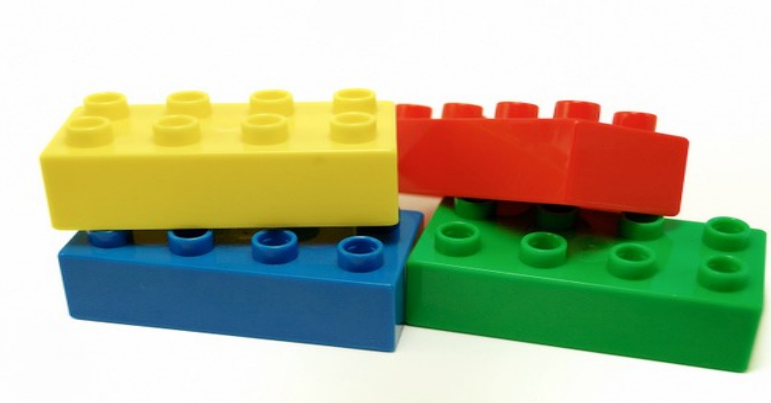
Tema 3

Introducción a Patrones para la Web

Índice

- Introducción
- MVC
- Capas
- Fachada
- Factoría
- DAO y DTO

Introducción



Df. de Patrón y tipos

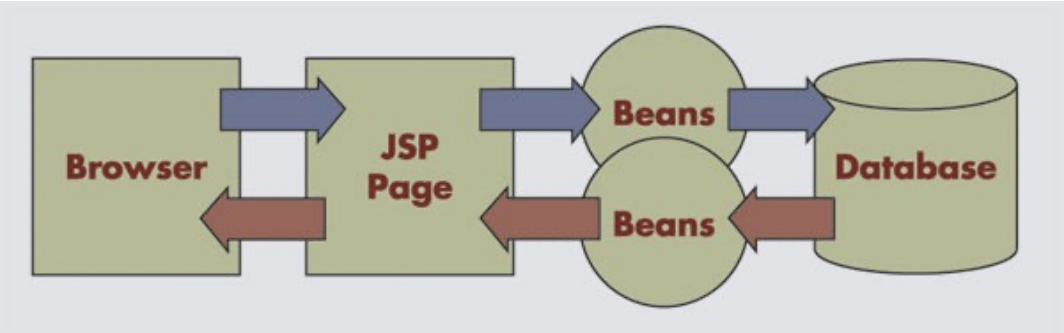
- Df. Un Patrón es la repetición de las mejores prácticas de lo que funciona en cualquier dominio
- Tipos de patrones:
  - **Arquitectónicos**: Relacionados con el diseño a gran escala y de granularidad gruesa. Ejemplo: El patrón Capas.
  - **Diseño**: Relacionados con el diseño de objetos y *frameworks* de pequeña y mediana escala. Ejemplo: El patrón *Fachada.*
  - **Estilos**: Soluciones de diseño de bajo nivel orientadas a la implementación o al lenguaje. Ejemplo: El patrón *Singleton.*

Patrones arquitectónicos y de diseño

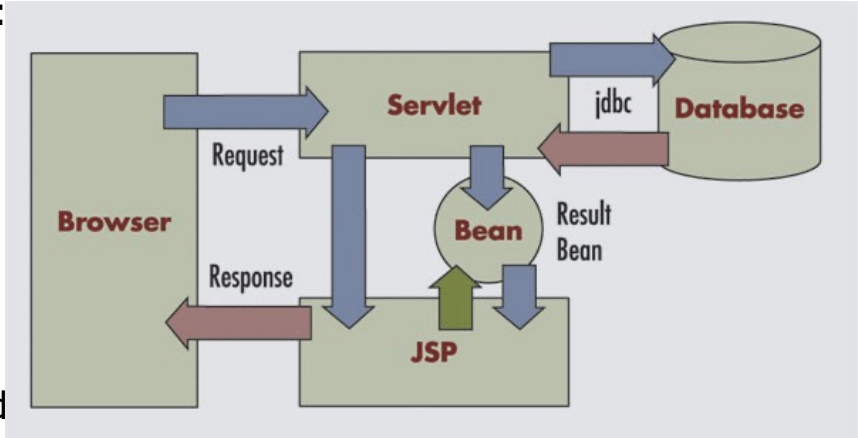
- En Pattern of Enterprise Application Architecture [Fowler03] los Patrones arquitectónicos se clasifican en:
  - Domain Logic Pattern.
  - Mapping to Relational Databases.
  - Web Presentation Patterns: MVC, Page Controller..
  - Session State Patterns
- Patrones de diseño. Un patrón de arquitectura puede contener múltiples patrones de diseño [GOF94]. Por ejemplo en una arquitectura MVC se suelen emplear los siguientes patrones de diseño:
  - Creacionales (Factory, Protoype, …)
  - Estructurales (Facade, Adapter, …)
  - Comportamiento (Command, Interpreter, …)

Modelos de desarrollo de aplicaciones Web en JEE (Servlets y JSPs)

- Dos arquitecturas en desuso:
  - Model-1.5: JSPs para presentación y control y JavaBeans para la lógica



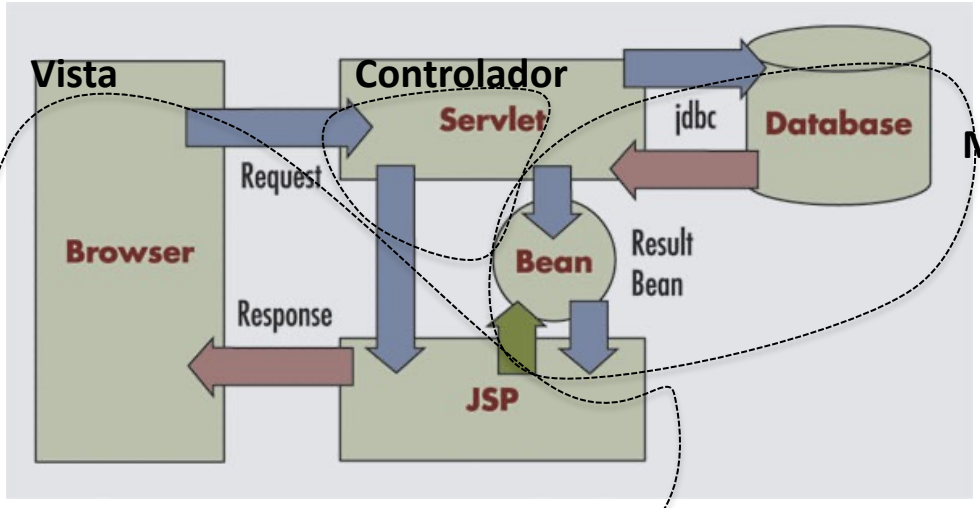
  - Model-2: Model-View-Controller = JavaBeans-JSPs-Servlets
    - MVC es tan común que se han desarrollado varias infraestructuras en torno a este patrón de diseño:
      - Apache Struts
      - Java Server Faces
      - Spring



MVC

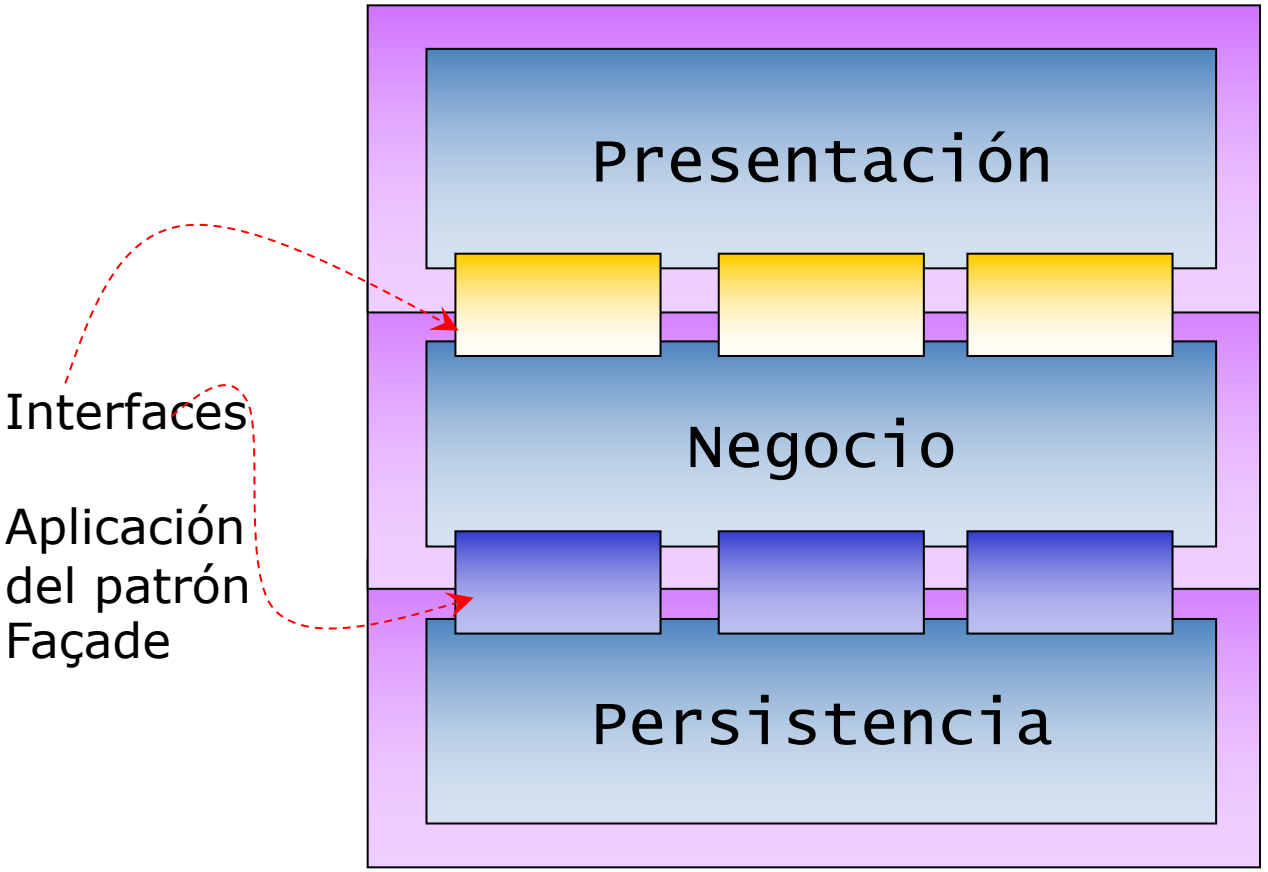
Patrón MVC

- Desarrollado por *Trygve Reenskau* para la plataforma SmallTalk a finales de los 70s.
- Evolución del modelo 1.5 (sin controlador)
- Roles en el patrón Arquitectónico MVC.
  - Controlador: Navegación/Servlet
  - Modelo (Negocio y Datos): Servlet/Beans
  - Presentación: JSPs



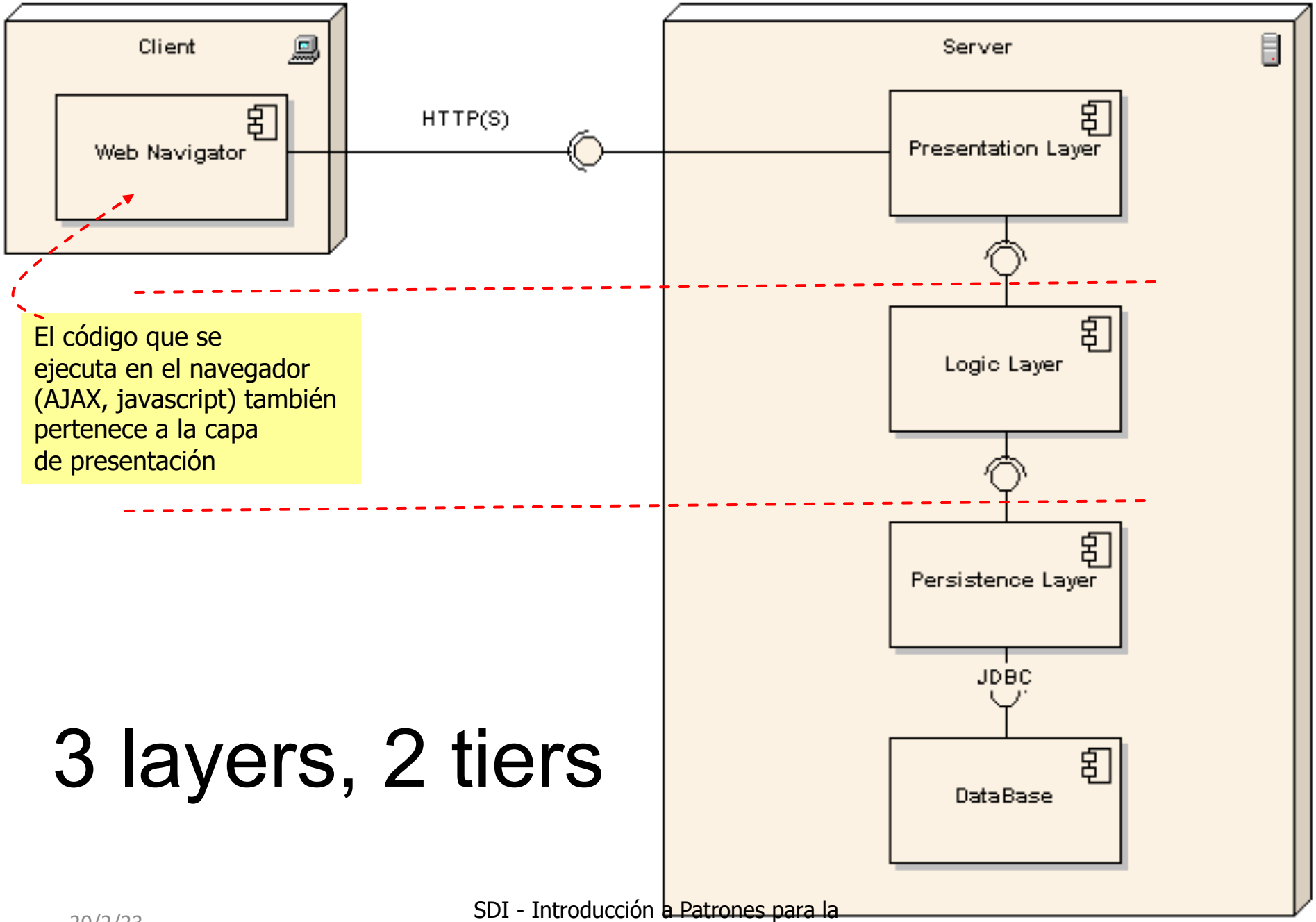
Patrón N-Capas

Patrón Capas (Modelo de Brown n-capas)



Layers y Tiers

- Layer: capa arquitectónica de la aplicación software
  - Presentación, lógica, persistencia
- Tier: capa física de la arquitectura de despliege del hardware
  - Máquinas: Servidor web, servidor de aplicaciones, servidor de base de datos
- Las "layers" se despliegan sobre las "tiers"



3 layers, 2 tiers

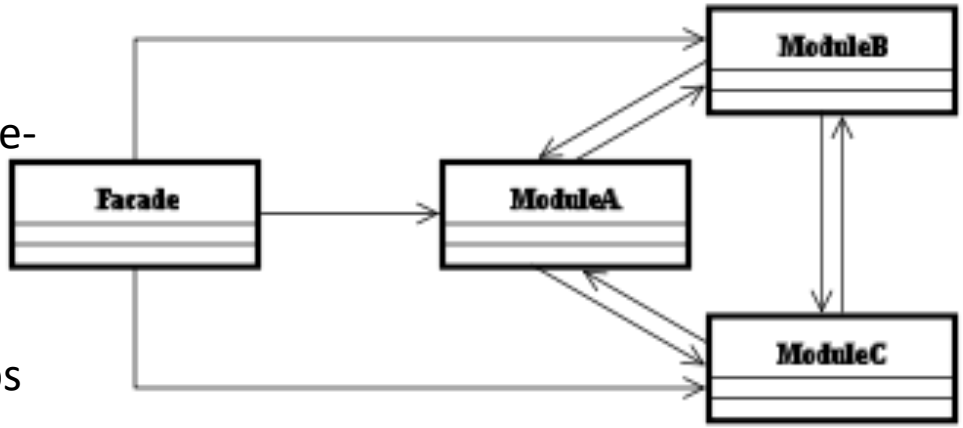
Arquitectura en capas: patrones

Presentación	Negocio	Persistencia
MVC	Fachada Factoría	DAO DTO Factoría Active Record

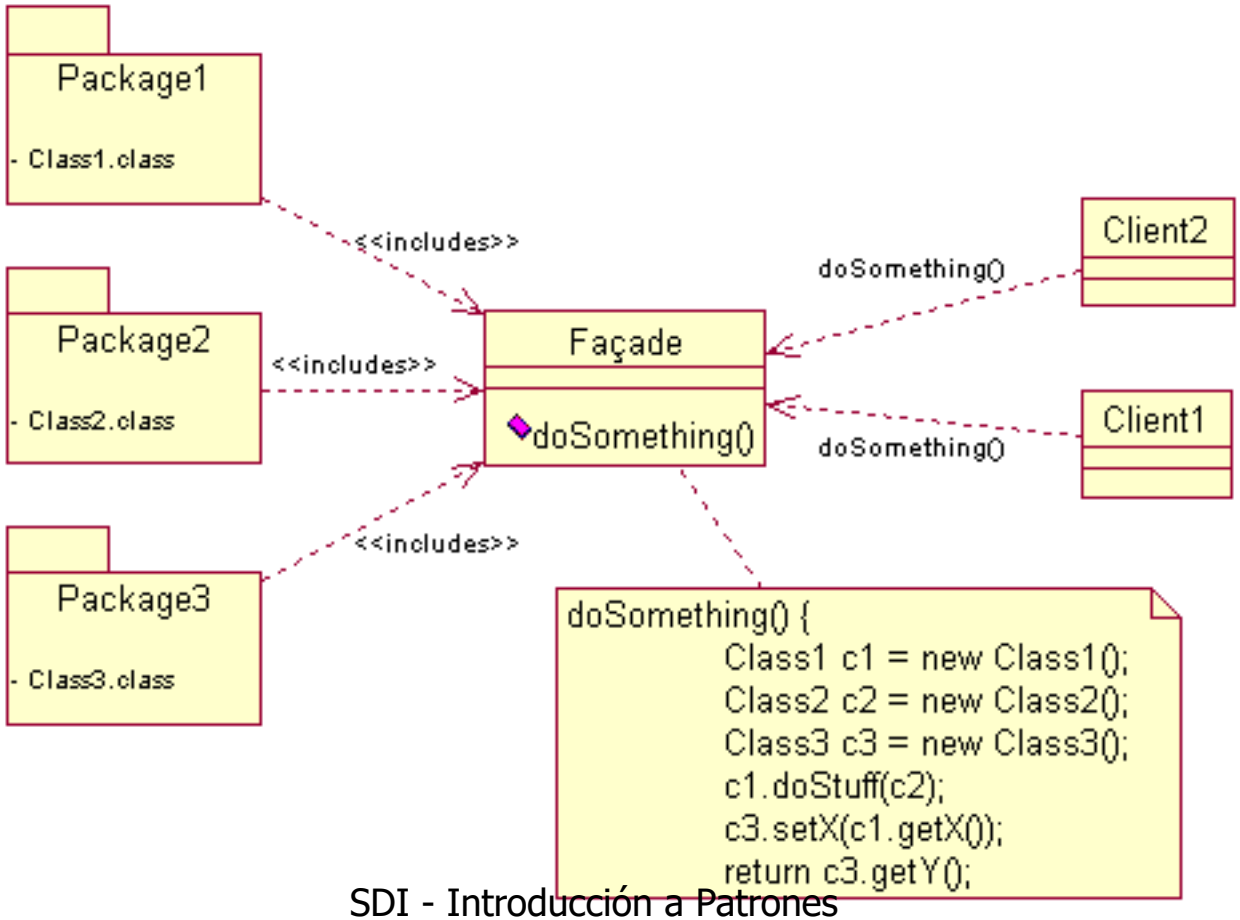
Patrón Fachada

- Df. Interfaz único y simplificado de los servicios más generales de un subsistema
- Cuando se usa:
  - Se busca un interfaz simple para un subsistema complejo
  - Hay muchas dependencias entre clientes y clases que implementan una abstracción
  - Se desea obtener una división en capas de nuestros subsistemas

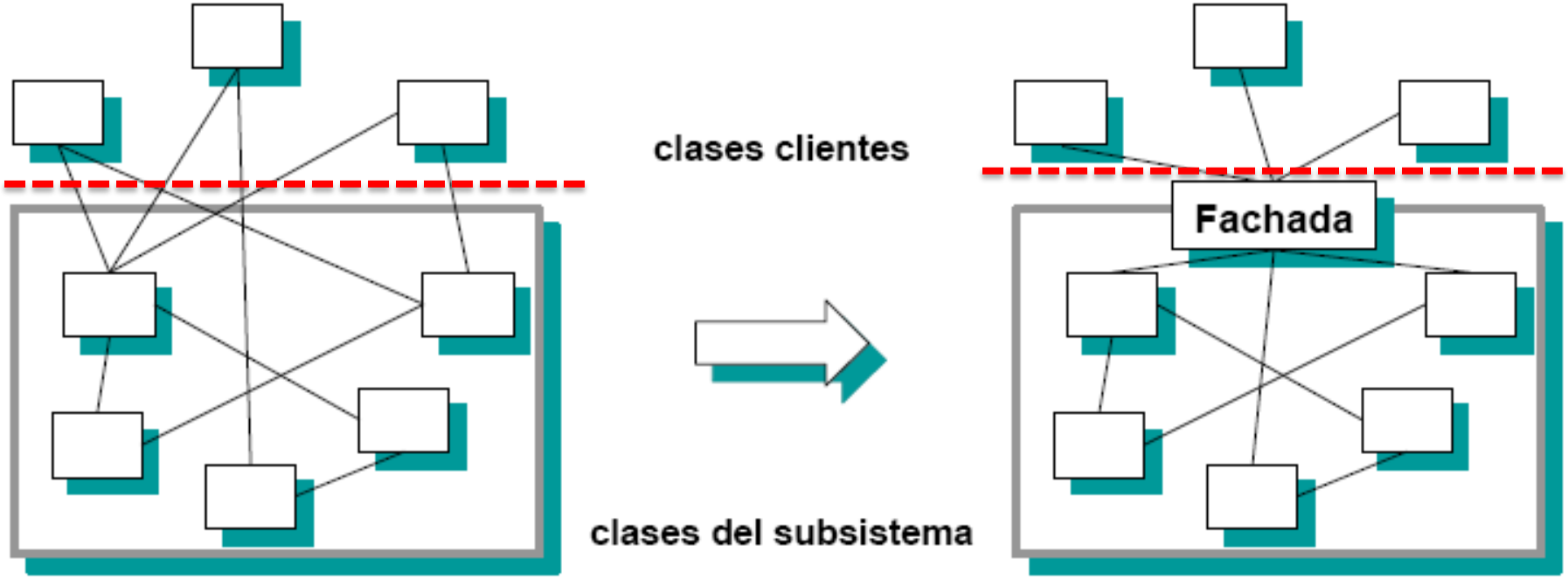
- Como se usa:
  - Reducción del acoplamiento cliente-subsistema (alternativa a la herencia).
  - Clases del subsistema públicas o privadas. No todos los lenguajes los soportan.



Capa de lógica: patrón fachada (*facade*)



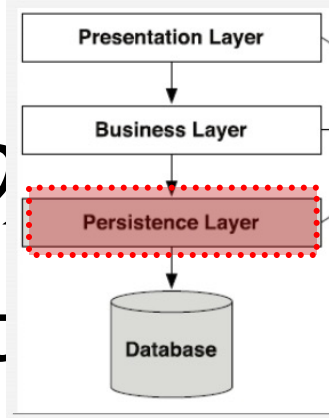
Desacomplamiento de capas



Capa de negocio: patrón factoría (*factory*)

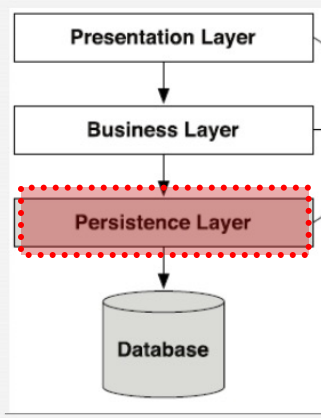
- Una factoría es un objeto encargado de la creación de otros objetos
- Utilizados en las ocasiones en las que hacerse con un objeto implica algo más complejo que crearlo
  - Crear la clase del objeto dinámicamente
  - Obtenerlo de un "pool" de objetos
  - Realizar una configuración compleja del mismo
  - Etc.
- El cliente no conoce el tipo concreto del objeto a crear
  - Sólo los conoce a través de su interfaz

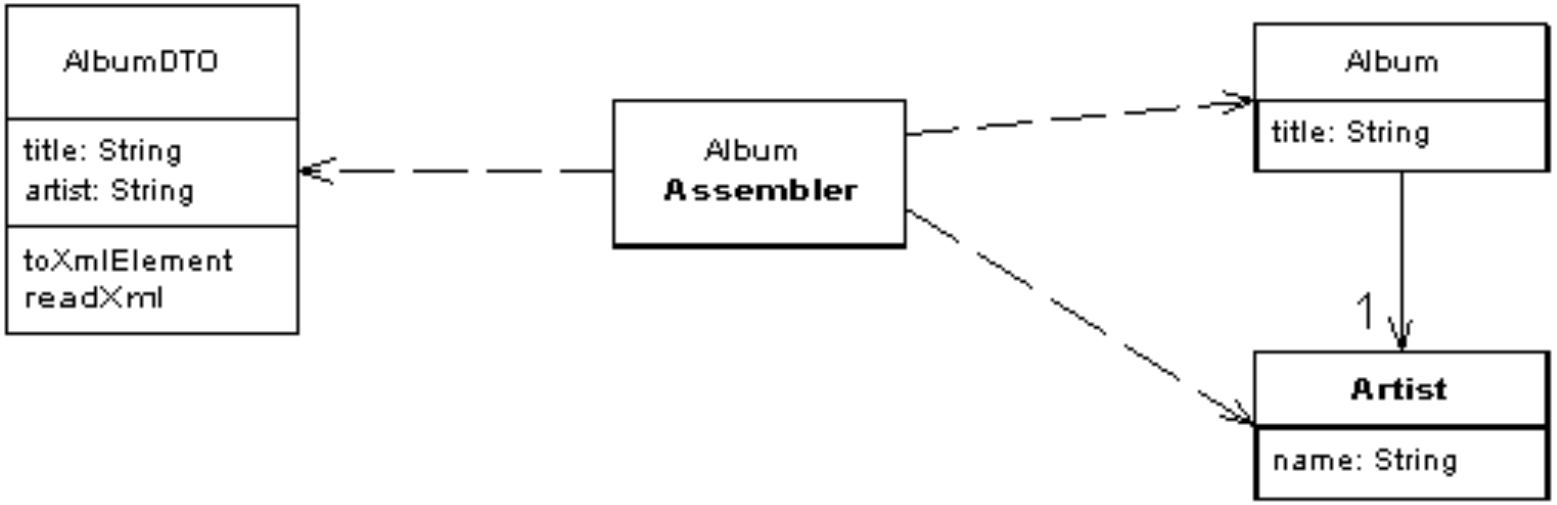
Patrón *Data Transfer Object (DTO*



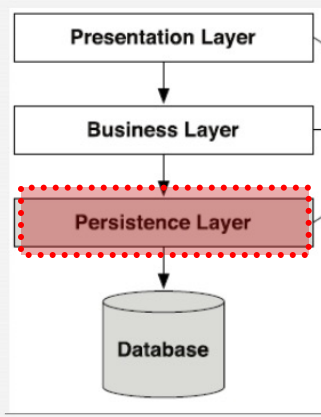
- Utilizado para transferir datos ent
  subsistemas
  - Para reducir el número de llamadas a método
  - Su único comportamiento viene dado básicamente por *getters* y *setters*
- Se utilizan a menudo en combinación con objetos DAO (persistencia) para obtener datos de una base de datos

Capa de persistencia: patrón *Data Transfer Object (DTO)*





DAO



- DAO →Data Access Object
- DAO proporciona una interfaz única de acceso a los datos, de forma independiente a dónde se hallen almacenados.
- Independiza la lógica de negocio del acceso a los datos.
- Ofrece operaciones CRUD para cada objeto persistente del dominio

Interfaces DAO: ejemplo

```
public interface GenericDao<T> {
    void save(T t);
    T update(T t);
    void delete(T t);

    T findById(Long id);
    Collection<T> findAll();
}
```

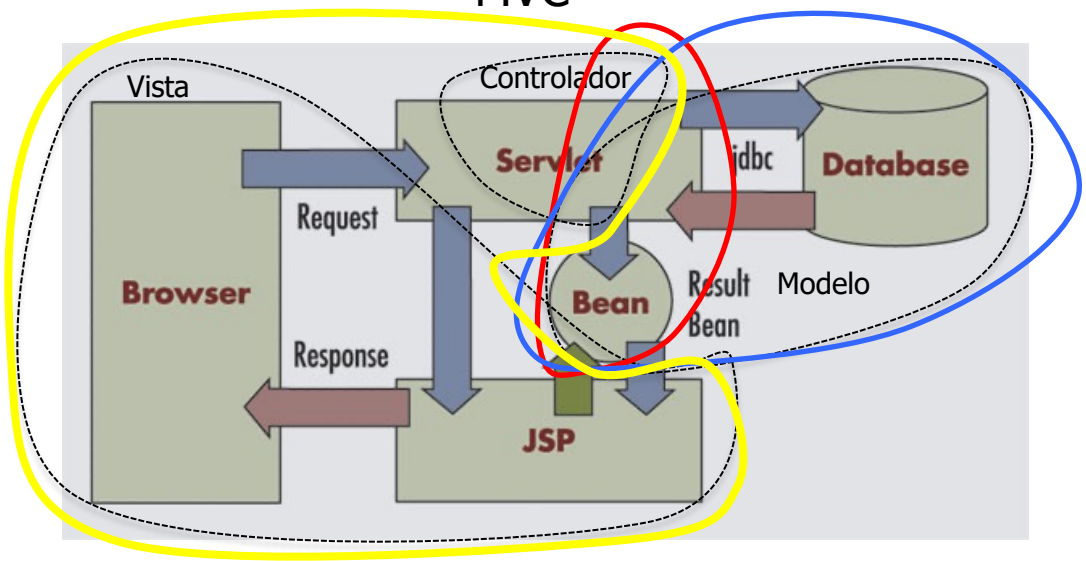
Métodos CRUD básicos

Métodos CRUD específicos para cada entidad del modelo

```
public interface StudentDao extends GenericDao<Student>{
    Collection<Student> findByNameSurname(String name, String surname);
    Collection<Student> findByEnrollmentId(Long enrollmentId);
    Collection<Student> findMatesByNameSurnameCourse(String name,
            String surname, String course);
    Set<Student> findUnenrolledAfterDate(Date date);
}
```

Acoplamiento entre MVC y N-Capas

MVC



N-CAPAS

Correspondencia de capas

MVC	N-Capas
Vista	Presentación
Controlador	Presentación
Modelo	Negocio/Persistencia

Referencias

- URLs
  - http://jakarta.apache.org/Struts
  - http://theserverside.com
- Libros
  - *Programming Jakarta Struts* de O'Reilly
  - *Mastering Tomcat Development* de WILEY
  - *Java Server Programming J2EE Edition* de Wrox
  - Marty Hall, Java Core Servlets
  - GOF94
  - Fowler93