Coco 1° Proyecto

Investigación.

Contenido

[Facelets 4](#_Toc498043400)

[Que es: 4](#_Toc498043401)

[Para que sirve: 4](#_Toc498043402)

[Relación: 4](#_Toc498043403)

[Ejemplo: 4](#_Toc498043404)

[Mapeo 6](#_Toc498043405)

[Que es: 6](#_Toc498043406)

[Para que sirve: 6](#_Toc498043407)

[Relación: 6](#_Toc498043408)

[Ejemplo: 6](#_Toc498043409)

[Primefaces 7](#_Toc498043410)

[Que es: 7](#_Toc498043411)

[Para que sirve: 7](#_Toc498043412)

[JSF 8](#_Toc498043413)

[Que es: 8](#_Toc498043414)

[Para que sirve: 8](#_Toc498043415)

[Ejemplo…Creación de un sitio web utilizando JSF, PrimeFaces y Netbeans 9](#_Toc498043416)

[POJO 17](#_Toc498043417)

[Que es: 17](#_Toc498043418)

[Relación 17](#_Toc498043419)

[Ejemplo: 17](#_Toc498043420)

[JPA 19](#_Toc498043421)

[Que es: 19](#_Toc498043422)

[Para que sirve: 19](#_Toc498043423)

[Relación: 19](#_Toc498043424)

[Ejemplo 19](#_Toc498043425)

[JSTL 24](#_Toc498043426)

[Qué es: 24](#_Toc498043427)

[Para que sirve: 24](#_Toc498043428)

[Ejemplo 25](#_Toc498043429)

[EJB 26](#_Toc498043430)

[Que es: 26](#_Toc498043431)

[Ejemplo: 26](#_Toc498043432)

[Bean Administrado 27](#_Toc498043433)

[Módulos de navegación 28](#_Toc498043434)

[Entity Bean 29](#_Toc498043435)

[Triggers 30](#_Toc498043436)

[Entity Factory 33](#_Toc498043437)

[JPQL 34](#_Toc498043438)

[Que es: 34](#_Toc498043439)

[Para que sirve: 34](#_Toc498043440)

[Relación: 34](#_Toc498043441)

[Ejemplo: 35](#_Toc498043442)

[XHTML 37](#_Toc498043443)

[Que es: 37](#_Toc498043444)

[Para que sirve: 37](#_Toc498043445)

[Temas más importantes. 38](#_Toc498043446)

[Facelets 38](#_Toc498043447)

[Mapeo 38](#_Toc498043448)

[Primefaces 38](#_Toc498043449)

[POJO 38](#_Toc498043450)

[JPA 38](#_Toc498043451)

[JSTL 38](#_Toc498043452)

[EJB 39](#_Toc498043453)

# Facelets

## Que es:

Facelets es un sistema de código abierto de plantillas web bajo la Licencia Apache y la tecnología de controlador de JavaServer Faces (JSF). El lenguaje requiere documentos XML de entrada válida para trabajar.

Facelets es un lenguaje de declaración de página poderoso pero liviano que se usa para construir vistas de JavaServer Faces usando plantillas de estilo HTML y para construir árboles de componentes. Las características de Facelets incluyen lo siguiente:

* Uso de XHTML para crear páginas web
* Soporte para bibliotecas de etiquetas Facelets además de JavaServer Faces y bibliotecas de etiquetas JSTL
* Soporte para el Lenguaje de Expresión (EL)
* Plantilla para componentes y páginas

## Para que sirve:

Las ventajas de Facelets para proyectos de desarrollo a gran escala incluyen lo siguiente:

* Soporte para la reutilización de código a través de plantillas y componentes compuestos
* Extensión funcional de componentes y otros objetos del lado del servidor mediante personalización
* Tiempo de compilación más rápido
* Validación EL en tiempo de compilación
* Rendimiento de alto rendimiento

## Relación:

Facelets es compatible con todos los componentes de la interfaz de usuario de JSF y se centra por completo en la construcción del árbol de componentes, lo que refleja el punto de vista de una aplicación JSF.

## Ejemplo:

En Facelets se pueden introducir plantillas de etiquetas desde una biblioteca de etiquetas de dos formas:

1. Directamente como un elemento xml calificado
2. Indirectamente a través del JSFC como atributo de un elemento no calificado arbitrariamente.

En este último caso el compilador Facelet ignorará el elemento actual y procesará el elemento que era dado por JSFC.

El siguiente ejemplo muestra el uso directo de las etiquetas calificadas:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0

Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://java.sun.com/jsf/html"**>**

**<body>**

**<h:form>**

**<h:outputText** value="Welcome, #{loggedInUser.name}" disabled="#{empty loggedInUser}" **/>**

**<h:inputText** value="#{bean.property}" **/>**

**<h:commandButton** value="OK" action="#{bean.doSomething}" **/>**

**</h:form>**

**</body>**

**</html>**

Uso del atributo JSFC, el mismo código también se puede expresar como el ejemplo dado a continuación:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0

Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<**html** xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://java.sun.com/jsf/html">

<**body**>

<**form** jsfc="h:form">

<**span** jsfc="h:outputText" value="Welcome, #{loggedInUser.name}" disabled="#{empty loggedInUser}" />

<**input** type="text" jsfc="h:inputText" value="#{bean.property}" />

<**input** type="submit" jsfc="h:commandButton" value="OK" action="#{bean.doSomething}" />

</**form**>

</**body**>

</**html**>

# Mapeo

## Que es:

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

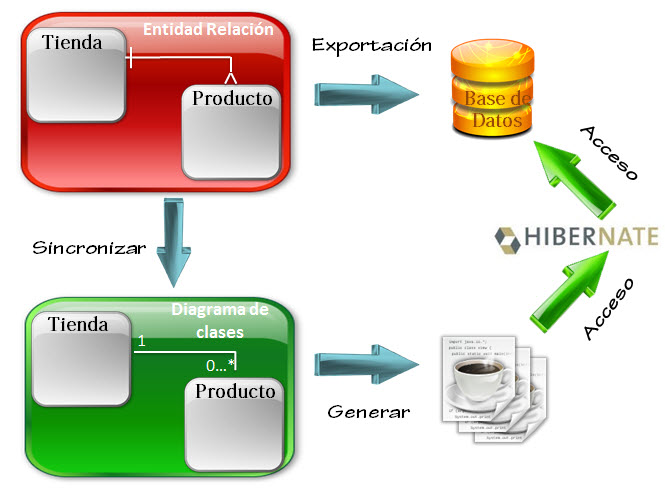
## Para que sirve:

Una implementación del mapeo relacional de objetos podría necesitar elegir de manera sistemática y predictiva qué tablas usar y generar las sentencias SQL necesarias.

## Relación:

Otra solución puede ser el uso de un sistema de administración de base de datos orientada a objetos (OODBMS: Object-oriented database management system), lo cual, como el nombre lo sugiere, es una base de datos diseñada específicamente para trabajar con valores orientados a objetos.

## Ejemplo:



https://es.wikipedia.org/wiki/Mapeo\_objeto-relacional

# Primefaces

## Que es:

PrimeFaces es una librería de componentes visuales open source desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing.

## Para que sirve:

Las principales características de Primefaces son:

* soporte nativo de Ajax, incluyendo Push/Comet.
* kit para crear aplicaciones web para móviles.
* es compatible con otras librerías de componentes, como JBoss RichFaces.
* uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).
* es un proyecto open source, activo y bastante estable entre versiones.

Algunos inconvenientes podrían ser:

* para utilizar el soporte de Ajax tenemos que indicarlo explícitamente, por medio de atributos específicos de cada componente.
* no podemos utilizar el soporte de Ajax de JSF 2 (mediante <f:ajax>) con los componentes de Primefaces.

https://www.adictosaltrabajo.com/tutoriales/introduccion-primefaces/

https://www.primefaces.org/

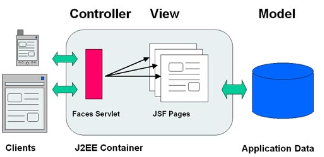
# JSF

## Que es:

JavaServer Faces (JSF) es un framework basado en el patrón MVC (Modelo Vista Controlador) para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

## Para que sirve:

JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web. Ofrece una clara separación entre el comportamiento y la presentación, lo que permite a cada miembro del equipo de desarrollo web enfocarse en su parte del proceso de desarrollo, y proporciona un modelo de programación para enlazar todas las piezas.



Tal y como hemos comentado, JSF se ejecuta sobre la tecnología de Servlets y no requiere ningún servicio adicional, por lo que para ejecutar aplicaciones JSF sólo necesitamos un contenedor de servlets tipo Tomcat o Jetty.

Algunas de las características más importantes de JSF en la actualidad:

* Framework estándar definido en la especificación Java EE.
* Soporte en todos los servidores de aplicaciones y en las herramientas de desarrollo: Eclipse, GlassFish, etc.
* Entornos gráficos para desarrollar rápidamente aplicaciones JSF.
* Gran variedad de implementaciones de componentes.
* Fácil integración con frameworks en la capa de negocio y de persistencia: Spring, JPA, etc.
* Comunidad muy activa en la actualidad; podemos encontrar fácilmente soporte en foros, artículos, tutoriales, etc.

## Ejemplo…Creación de un sitio web utilizando JSF, PrimeFaces y Netbeans

|  |
| --- |
| A continuación crearemos un pequeño ejemplo que ejemplificará el uso de JavaServer Faces 2.1 y PrimeFaces 3.1.1 utilizando el IDE Netbeans 7.1.1.  Lo primero que vamos a hacer es crear un proyecto  [Creación de proyecto Maven](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/crear%20proyecto%20maven.png?attredirects=0)  Colocamos los datos del proyecto para mi caso colocaré los que se muestran en la imagen:  [Datos del proyecto](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/datos%20proyecto.png?attredirects=0)  Oprimimos el botón de siguiente, seleccionamos Glassfish 3.x como servidor de aplicaciones y seleccionamos Finalizar.  Una vez que se crea el proyecto podemos ver la estructura del proyecto que tenemos:      -Web Pages: En esta carpeta se guardarán los archivos referentes a las paginas de Internet, tales como JSP'S, JSF Pages,HTML, XHTML, entre otros, así como archivos de configuración de JSF y de la aplicación tales como : web.xml y faces-config.xml(en este momento no existen pero los crearemos más adelante) .      -Source package: En esta carpeta crearemos todo el código Java que funcionará dentro de nuestra aplicación.      -Dependencies: En esta carpeta podremos ver las dependencias incluidas dentro de nuestro proyecto, se puede ver que por default se incluye una dependencia llamada javaee-web-api-6.0.jar, esta dependencia contiene el api de JavaEE 6.      -Java dependencies: En esta carpeta podremos obtener información sobre la versión de java que utilizaremos, en este caso voy a utilizar JDK 7.      -Proyect Files: En este lugar se crearán los archivos del proyecto, aquí podremos encontrar el archivo clave para maven, este archivo es el POM.xml del cual hablaremos más adelante.  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341816633231/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/estructura%20maven.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/estructura%20maven.png?attredirects=0)  Una vez creado el proyecto vamos a dar click secundario al proyecto ->Properties->Frameworks->Add->JavaServer Faces.  En este lugar vamos a agregar la configuración para JavaServer Faces.      -En la pestaña de Libraries vamos a dar click en Registred Libraries y seleccionaremos JSF 2.1.      -En la pestaña de Configuración vamos a JSF Servlet URL Pattern escribiremos \*.faces.      -En la pestaña de Components podemos ver que aparece PrimeFaces, pero no lo seleccionaremos ya que la versión que aparece disponible es la versión 2.2.1 y nosotros utilizaremos la versión más reciente que es la 3.1.1.  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341816997336/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/jsf%20config.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/jsf%20config.png?attredirects=0)  Oprimimos Ok y veremos algunos cambios en el proyecto, el primero y el más importante es la creación de una carpeta dentro de Web Pages llamada WEB-INF y el archivo web.xml, este archivo es un archivo clave dentro de las aplicaciones web en java y es donde podremos ver la configuración de JSF, el segundo es la creación de la carpeta Other Sources, es aquí donde colocaremos archivo que utilizaremos dentro de nuestra aplicación tales como archivos .properties, el tercer cambio es la inclusión de las dependencias de JSF en la carpeta de dependencies y el cuarto es la creación de una pagina llamada index.xhtml, la cual es una página de ejemplo de JSF.Todos estos cambios los veremos a detalle.  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341817287996/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/cambios.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/cambios.png?attredirects=0)  1.-Archivo web.xml  Como ya lo mencionamos este archivo es el corazón de nuestra aplicación web y es en este lugar donde encontraremos la configuración de JavaServer Faces. Vamos a analizar a detalle   los puntos importantes de este archivo.  Lo primero que vamos a analizar son las siguientes líneas las cuales representan un parámetro que nos dice en que modo vamos a estar trabajando con JSF, en este momento podemos ver que esta en modo Development o Desarrollo, pero este puede tomar los valores: Production, Development, UnitTest, SystemTest y Extensión. Estos valores lo único que representan es el tipo de mensajes que veremos dentro de la aplicación.   <context-param>          <param-name>javax.faces.PROJECT\_STAGE</param-name>          <param-value>Development</param-value>    </context-param>  La siguiente parte es la configuración del servlet de JavaServer Faces, a través del cual tienen que pasar todas las peticiones, en estas líneas lo que podemos ver es  que si se recibe una petición de un recurso cuya extensión sea .faces debe pasar por el Servlet llamado Faces Servlet cuya clase es  javax.faces.webapp.FacesServlet. Esto lo que nos indica es que cuando vallamos a referirnos dentro de la aplicación a una pagina hecha con JavaServer Faces debemos colocar la extensión .faces.    <servlet>          <servlet-name>Faces Servlet</servlet-name>          <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>          <load-on-startup>1</load-on-startup>      </servlet>      <servlet-mapping>          <servlet-name>Faces Servlet</servlet-name>          <url-pattern>\*.faces</url-pattern>      </servlet-mapping>  Con esto tenemos la configuración de JSF.  2.-La carpeta de Other sources es el lugar donde colocaremos los archivos de configuración del proyecto, tales como archivos .xml o .properties.  3.-Las dependencias que aparecen en la carpeta de dependencies ya estan incluidas dentro de la dependencia javaee-web-api-6.0.jar, por tal motivo las podemos quitar de la aplicación dando click secundario sobre ella y oprimiendo el botón "Remove Dependency" esto se debe hacer en todas las dependencias menos en javaee-web-api-6.0.jar.  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341818047762/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/borrar%20dependencia.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/borrar%20dependencia.png?attredirects=0)  4.-La pagina index.jsp ya no será necesaria dentro de nuestra aplicación debido a que a partir de ahora vamos a utilizar archivos con extensión .xhtml llamados (Facelets) por tal motivo nos podemos sentir libres de borrar el archivo index.jsp. En la pagina index.xhtml es donde pondremos nuestro código JSF.  Con esto tenemos lista la configuración de JSF correctamente, hasta este momento nuestro proyecto se debe ver como se muestra en la siguiente imagen:  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341818241416/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/jsf%20configuracion%20ok.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/jsf%20configuracion%20ok.png?attredirects=0)  Como ya se mencionó antes con esto tenemos configurado JSF para poder trabajar con él, lo único que nos hace falta es la configuración de PrimeFaces para esto entraremos en la página de  http://primefaces.org/downloads.html  en la cual podremos encontrar las dependencias y el repositorio necesario para configurar PrimeFaces.  Lo primero que debemos hacer es copiar el repositorio que se encuentra en la página, este repositorio lo debemos de agregar al archivo de configuración pom,xml de nuestro proyecto. Para hacer esto dentro de la etiqueta <repositories>  agregaremos el repositorio de maven que copiamos de la página de primefaces.org, una vez hecho esto vamos a agregar dentro de nuestras dependencias la dependencia de PrimeFaces que vamos a descargar.  Repositorio de PrimeFaces:  <repository>      <id>prime-repo</id>      <name>PrimeFaces Maven Repository</name>      <url>http://repository.primefaces.org</url>      <layout>default</layout>  </repository>  Dependencia de PrimeFaces:  <dependency>      <groupId>org.primefaces</groupId>      <artifactId>primefaces</artifactId>      <version>3.3.1</version>  </dependency>  Una vez hecho esto nuestro proyecto se debe ver de la siguiente forma:  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341818947156/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/dependency%20prime.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/dependency%20prime.png?attredirects=0)  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341818935953/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/repository%20prime.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/repository%20prime.png?attredirects=0)  Una vez hecho esto damos click secundario al proyecto y seleccionamos la opción "Build with Dependencies" para descargar la dependenca de PrimeFaces. Una vez hecho esto nos iremos al archivo index.xhtml y agregaremos el namespace de primefaces, esto es agregar dentro de la etiqueta <html> el atributo xmlns:p="http://primefaces.org/ui", con esto estará configurado primefaces dentro de nuestra página utilizando el prefijo "p".Para probar que PrimeFaces fue configurado correctamente vamos a escribir dentro de la etiqueta body lo siguiente:  <p:calendar/>  Nuestro archivo index.xhtml debe quedar de la siguiente forma :  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341819259922/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/archivo%20xhtml%20listo.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/archivo%20xhtml%20listo.png?attredirects=0)  Listo esto nos iremos al nombre del proyecto, click secundario Run  y veamos PrimeFaces en acción :  [https://sites.google.com/site/geekjavamexico/_/rsrc/1341819352974/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/prime%20en%20accion.png?height=250&width=400](https://sites.google.com/site/geekjavamexico/home/creacion-de-un-sitio-web-utilizando-jsf-2-1-primefaces-3-1-1-y-netbeans-7-1-1/prime%20en%20accion.png?attredirects=0)  Como podemos ver ya tenemos lista nuestra aplicación para trabajar con JSF 2.1 Primefaces 3.1.1 y Maven. |

https://www.adictosaltrabajo.com/tutoriales/jstl-jsf/

http://www.it.uc3m.es/spickin/docencia/comsoft/presentations/spanish/07-08/JSF.pdf

http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html

https://pajarokillo.wordpress.com/2011/11/23/empezando-con-jsf-2-0-i/

https://www.adictosaltrabajo.com/tutoriales/introduccion-primefaces/

# POJO

## Que es:

POJO son las iniciales de “Plain Old Java Object”, que puede interpretarse como “Un objeto Java Plano Antiguo”. Un POJO es una instancia de una clase que no extiende ni implementa nada en especial. Para los programadores Java sirve para enfatizar el uso de clases simples y que no dependen de un framework en especial.

Por ejemplo, un Controller de Spring tiene que extender de SimpleFormController, e implementar los métodos abstractos de ese tipo: por eso, no es un POJO. Un Servlet, tiene que extender de HttpServlet por lo que tampoco es un POJO. En cambio, si defines una clase Cliente con atributos y unas cuantas operaciones, tienes un simple y modesto POJO.

## Relación

POJO con EJB  
Aunque los objetos POJO ya existían, desempeñan un rol más importante en la metodología de programación de Java™ EE 5. Ahora puede crear aplicaciones EJB con posibilidades de persistencia mediante beans de empresa y entidades creadas a partid de objetos POJO.

La complejidad de la infraestructura de Java 2 Enterprise Edition solía presentar más dificultades para la adopción anteriormente. La especificación Java EE 5 trató de ofrecer una experiencia de desarrollo más simple haciendo que los objetos POJO constituyeran la base de su paradigma de diseño. El modelo de programación POJO le permite realizar pruebas unitarias fuera del servidor de aplicaciones, facilitando todo el proceso de programación.

## Ejemplo:

Un objeto POJO simple

A continuación se proporciona un ejemplo de código de un objeto POJO simple. Tenga en cuenta que no hay referencias a ninguna interfaz. Para utilizar este objeto POJO como base para una aplicación EJB 2.1 se necesitan más clases de infraestructura de soporte y la clase en sí debería implementar más interfaces.

public class Test {

String name;

/\*\*

\* This is a constructor for a Test Object.

\*\*/

public Test(){

name = "Jane";

}

}

Para crear un bean EJB 3.0, inyecte una anotación de definición en el nivel de clase. El ejemplo siguiente convierte un objeto POJO en un bean de sesión sin estado añadiendo la anotación @Stateless.

@Stateless

String class Test {

String name;

/\*\*

\* This is a constructor for a Test Object.

\*\*/

public Test () {

name = "jane";

}

}

En una aplicación real, el objeto POJO necesita lógica de negocio adicional. La idea fundamental de utilizar objetos POJO en el contexto de la especificación Java EE es asociar metadatos sobre el componente que debe asociarse directamente en el POJO. Este procedimiento reduce el número de artefactos con los que necesita tratar y permite asegurar más fácilmente la integridad de los metadatos.

El nuevo modelo de programación basado en POJO también se centra en funcionar en los componentes de Java EE 5 en el editor de Java en contraposición con la escritura de lógica en XML. Al trabajar con este software de desarrollo de aplicaciones, puede sacar partido de las características que pueden simplificar más aún el proceso de desarrollo de aplicaciones Java EE 5 (como por ejemplo la validación durante el tecleo, la asistencia de contenido y la refactorización).

http://carlospesquera.com/que-es-un-pojo-ejb-y-un-bean/

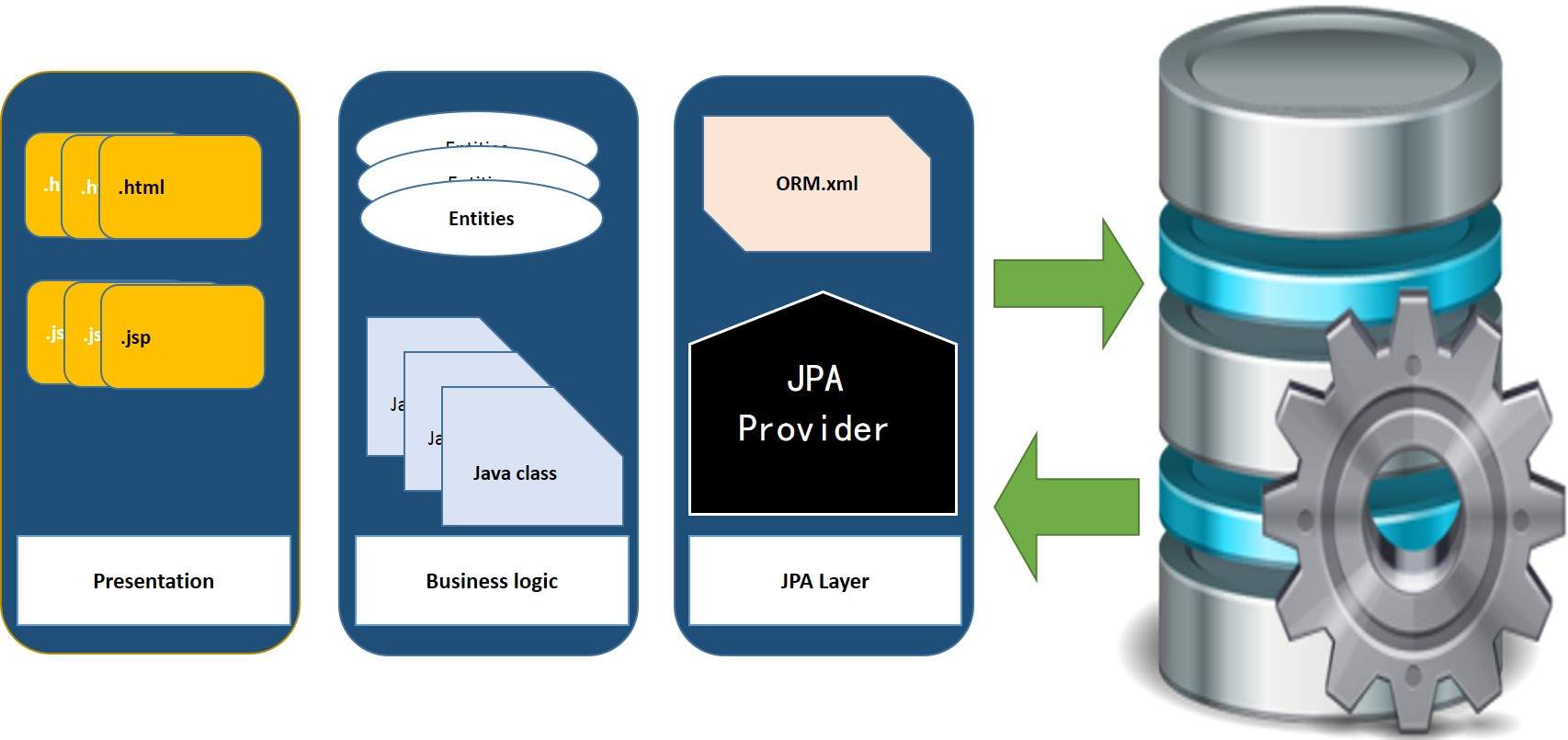
# JPA

## Que es:

JPA es un conjunto de clases y métodos para almacenar gran cantidad de datos en una base de datos. Gracias a JPA, la carga de interactuar con la base de datos se reduce considerablemente. Esto constituye un puente entre los modelos de objetos (programa Java) y modelos relacionales (programa de base de datos).

## Para que sirve:

Para reducir la carga de escribir códigos relacionales para la gestión de objetos, el programador utiliza el framework del “Proveedor JPA”, el cuál le permite la fácil interacción con la instancia de la base de datos.



## Relación:

La mayoría de las nuevas aplicaciones utiliza una base de datos relacionada para almacenar datos, es por ello que un mapeo objeto-relacional (ORM) es indispensable.

Muestra cómo definir un Plain Oriented Java Object (POJO) como una entidad y la forma de gestionar las entidades con las relaciones.

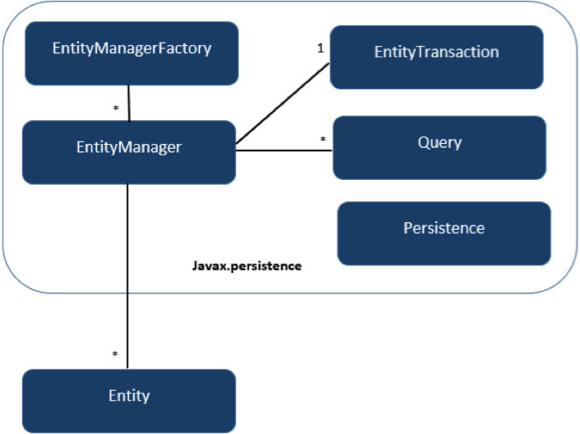
## Ejemplo

Antes de comenzar con el ejemplo, cabe mencionar las unidades más importantes de JPA:

* **Entity:** las entidades son los objetos persistentes, almacena los registros en la base de datos.
* **EntityManager:** Es una interfaz que gestiona la persistencia de las operaciones en objetos. Funciona como un fábrica para consultas.
* **EntityManagerFactory:** Esta es una clase de fábrica de EntityManager. Crea y gestiona múltiples instancias de EntityManager.
* **EntityTransaction:** Tiene una realción uno a uno con EntityManager. Por cada EntityManager, las operaciones son mantenidos por una clase Entity Transaction.
* **Persistence:** Esta clase contiene métodos estáticos para obtener instancias de EntityManagerFactory.
* **Query:** Esta interfaz es implementada por cada Proveedor JPA para obtener objetos relaciones que cumplan los criterios.

Relaciones de una clase con otra:

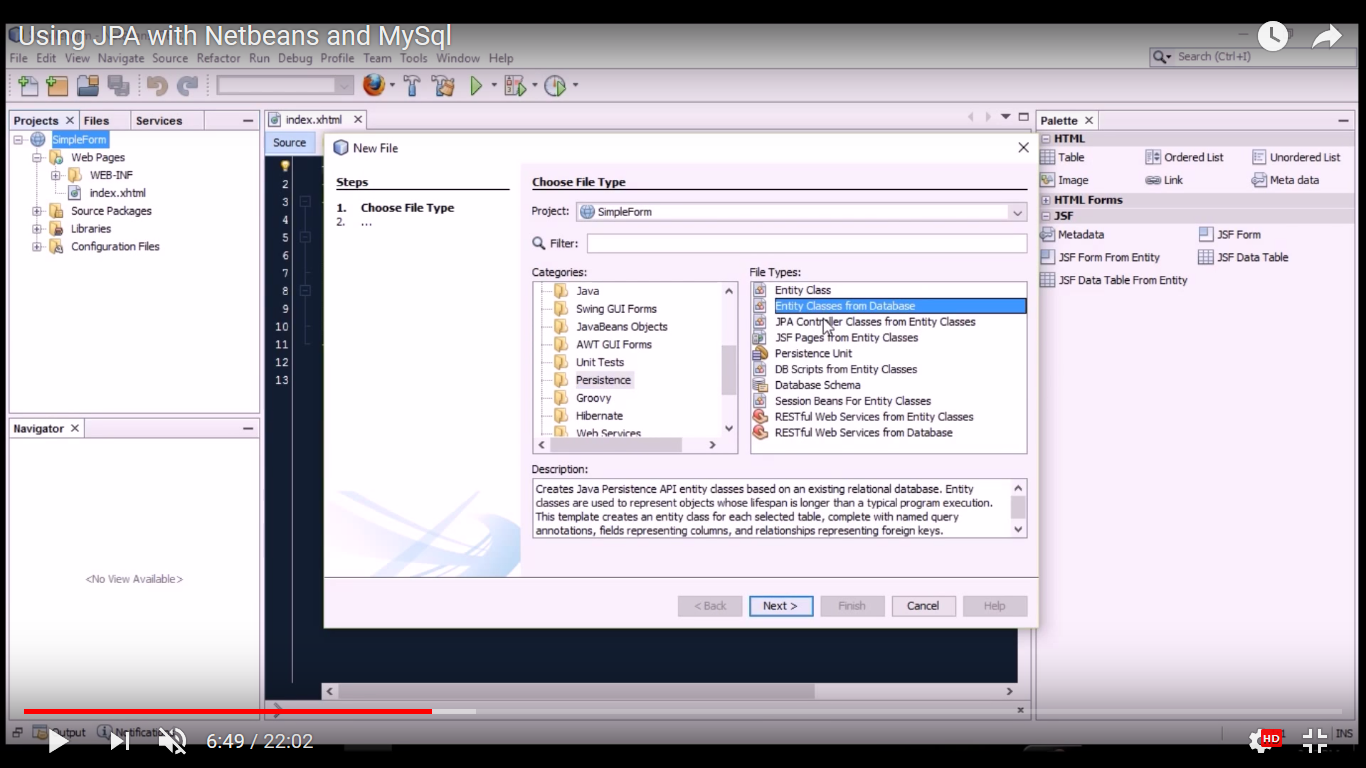
* La relación entre EntityManagerFactory y EntityManager, es de uno a varios. Se trata de una clase que fabrica instancias de EntityManager.
* La relación entre el método EntityManager y EntityTransaction es uno a uno. Por cada operación de EntityManager, hay una interacción con EntityTransaction.
* La relación entre EntityManager y Query es de uno a varios. Varias consultas se pueden ejecutar mediante una instancia de EntityManager.
* La relación entre EntityManager y Entity es uno a muchos. Una instancia de EntityManager puede gestionar múltiples Entities.



1. Comencemos creando una base de datos de nombre: jpa\_db

Usuario (id INT AUTOINCREMENT PRIMARY KEY NOT NULL, nombre VARCHAR(20), a\_paterno VARCHAR(20), a\_materno VARCHAR(20));

**Inserte por lo menos una fila de datos.**

1. A continuación se creará un Java Web Application en NetBeans, llamado FormularioJPA, utilice el servidor de su preferencia. Al llegar al paso 4 (Frameworks), **seleccionará JavaServer Faces** y terminar.
2. Después, creará una conexión con la base de datos.
3. Paso seguido, crear un Persistence Entity. Haga clic derecho en el proyecto y en la opción Nuevo, seleccione **“Otros”,** se abrirá una ventana, seleccione la categoría **“Persistence”** y busque **“Entity Classes from Database”**.  
   A continuación, creará un nuevo **“Data Source”** con la conexión de nuestra base de datos. Añada la tabla de Usuario y continúe. Utilice el nombre de **“Entidad”** para el paquete y finalice. Para terminar el paso, sobrecargue el constructor de Usuario().
4. Creamos una Sesión para la Entidad. Haga clic derecho en el proyecto y en la opción Nuevo, seleccione **“Session Beans For Entity Classes…”.** Agregue la tabla de Usuario y el nombre del paquete será “Sesión.
5. Procedemos a crear la función para guardar datos, en la clase de UsuariosFacade, creamos una función de “Guardar”:

*Public void Guardar (Int id, String nombre, String a\_paterno, String a\_materno) {*

*Try{*

*Usuario mUsusario = new Usuario(id, nombre, a\_paterno, a\_materno);*

*em.persist(mUsuario);*

*}* catch(Exception e) {

Throw new EJBException(e.getMessage());

}

1. Creamos una nueva clase de Java llamada **“Web”**:

package formulario.View;

import Session.UsuarioFacade;

import jacax.ejb.EJB;

import javax.enterprise.context.RequestScoped;

import javax.inject.Named;

/\*\*

\*

\* @author enriq

\*/

@Named

@RequestScoped

public class Web {

@EJB

private int id;

private String nombre;

private String a\_paterno;

private String a\_materno;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getA\_paterno() {

return a\_paterno;

}

public void setA\_paterno(String a\_paterno) {

this.a\_paterno = a\_paterno;

}

public String getA\_materno() {

return a\_materno;

}

public void setA\_materno(String a\_materno) {

this.a\_materno = a\_materno;

}

public void guardar(){

try {

}catch(Exception e) {

}

}

}

https://www.tutorialspoint.com/jpa/index.htm

https://docs.oracle.com/javaee/6/tutorial/doc/bnbqa.html

https://www.youtube.com/watch?v=xgXypqbjaDs

# JSTL

Qué es:  
La tecnología JavaServer Pages Standard Tag Library (JSTL) es un componente de Java EE. Extiende las ya conocidas JavaServer Pages (JSP) proporcionando cuatro bibliotecas de etiquetas (Tag Libraries) con utilidades ampliamente utilizadas en el desarrollo de páginas web dinámicas.

Estas bibliotecas de etiquetas extienden de la especificación de JSP (la cual a su vez extiende de la especificación de Servlet). Su API permite además desarrollar bibliotecas propias de etiquetas.

Las bibliotecas englobadas en JSTL son:

•core: iteraciones, condicionales, manipulación de URL y otras funciones generales.

•xml: para la manipulación de XML y para XML-Transformation.

•sql: para gestionar conexiones a bases de datos.

•fmt: para la internacionalización y formateo de las cadenas de caracteres como cifras.

Para que sirve:  
La especificación JSP ahora se ha convertido en una tecnología estándar para la creación de sitios Web dinámicos en Java, y el problema es que han aparecido algunas debilidades:

•El código Java embebido en scriptlets es desordenado.

•Un programador que no conoce Java no puede modificar el código Java embebido, anulando uno de los mayores beneficios de los JSP: permitir a los diseñadores y personas que escriben la lógica de presentación que actualicen el contenido de la página.

•El código de Java dentro de scriptlets JSP no pueden ser reutilizados por otros JSP, por lo tanto la lógica común termina siendo re-implementado en múltiples páginas.

•La recuperación de objetos fuera del HTTP Request y Session es complicada. Es necesario hacer el Casting de objetos y esto ocasiona que tengamos que importar más Clases en los JSP.

## Ejemplo

<https://www.youtube.com/watch?v=Sje5IpPh5vk>

http://java.ciberaula.com/articulo/introduccion\_jstl

http://desarrollo-java.readthedocs.io/es/latest/tutorial6.md.html

# EJB

## Que es:

Un Enterprise Java Bean (EJB) es un componente de negocio Java Enterprise, y para su ejecución necesita un contenedor EJB/J2EE (JBoss, WAS, OAS, etc). El hecho de usar EJB’s te da acceso a los servicios del Contenedor EJB (manejo de transacciones, seguridad, persistencia, etc) que simplifican bastante la construcción de soluciones empresariales.

Un JavaBean (o también conocido simplemente como Bean) es una clase simple en Java que cumple con ciertas normas con los nombres de sus propiedades y métodos.

Un JavaBean debe tener un constructor sin argumentos, tiene declarados todos sus atributos como privados y para cada uno de ellos un método setter y getter, añadiéndole la palabra “set” o “get” al nombre del atributo y deben ser serializables.

## Ejemplo:

**public** **class** ClienteBean **implements** java.io.Serializable {

**private** String nombre;

**private** **int** edad;

**public** ClienteBean() {

*// Constructor*

   }

**public** String getNombre() {

**return** nombre;

   }

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

   }

**public** **int** getEdad() {

**return** edad;

   }

**public** **void** setEdad(**int** edad) {

**this**.edad = edad;

   }

}

https://www.ibm.com/support/knowledgecenter/es/SS5JSH\_9.1.2/org.eclipse.jst.ejb.doc.user/topics/cpojosandee5.html

# Bean Administrado

Lorem ipsum

# Módulos de navegación

## Que es:

Un módulo debe declarar que otros módulos requiere para poder ser compilado y para ejecutarse. También debe declarar que paquetes dentro del módulo son exportados como su interfaz.

*module com.foo.bar {*

*requires com.foo.baz;*

*com.foo.bar.alpha;*

*com.foo.bar.beta;*

*}*

## Para que sirve:

Tanto la declaración de los módulos que requiere un módulo, como de los paquetes que exporta, es opcional. La definición más básica de un módulo es por tanto:

*module com.foo.bar { }*

El código fuente de la declaración de un módulo por convenio se pondrá en un archivo con nombre module-info.java que se encontrara en la raíz de la jerarquía de paquetes donde comienza el módulo. Dicho archivo será compilado, como cualquier archivo Java, a un archivo .class

## Ejemplo:

Empaquetando módulos

Los módulos se podrán empaquetar en archivos jar especiales que se denominarán "jar modulares" ("modular JAR files"). Estos archivos eran exactamente iguales que los archivos jar actuales, sólo que en su raíz deben contener el archivo module-info.class. Por ejemplo, un jar modular podría contener:

*META-INF/*

*META-INF/MANIFEST.MF*

*module-info.class*

*com/foo/bar/alpha/AlphaFactory.class*

*com/foo/bar/alpha/Alpha.class*.

En Java 9 una implementación de Java SE podrá contener todos o parte de los módulos de la plataforma. El único módulo que va a ser obligatorio va a ser el módulo java.base, cuyo contenido principal es:

*module java.base {*

*exports java.io;*

*exports java.lang;*

*exports java.lang.annotation;*

*exports java.lang.invoke;*

*exports java.lang.module;*

*exports java.lang.ref;*

*exports java.lang.reflect;*

*exports java.math;*

*exports java.net;*

*....*

*}*

Aunque todavía no es completamente seguro, probablemente habrá otros módulos independientes que contengan java.sql.\*, java.xml.\* y java.logging.\*, entre otros.

Dependencias entre módulos

Cuando un módulo depende de otro, indirectamente las dependencias del segundo también son necesarias para el primero. Así por ejemplo, si tenemos el módulo:

*module com.foo.app {*

*requires com.foo.bar;*

*requires java.sql;*

*}*

Y a su vez el módulo java.sql ha sido definido como:

*module java.sql {*

*requires java.logging;*

*requires java.xml;*

*exports java.sql;*

*exports javax.sql;*

*exports javax.transaction.xa;*

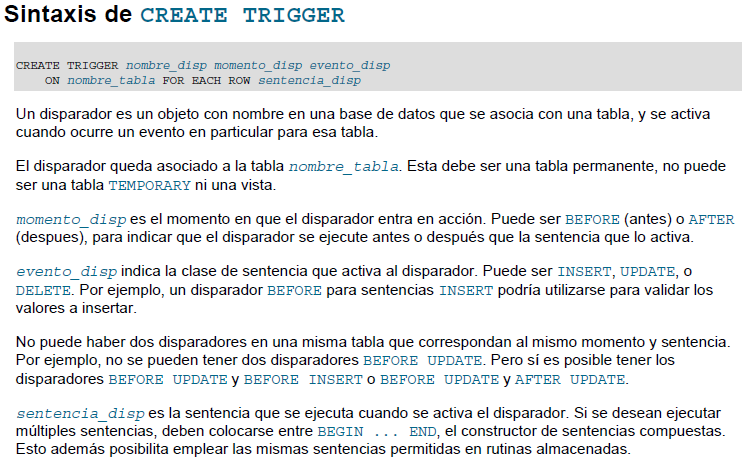
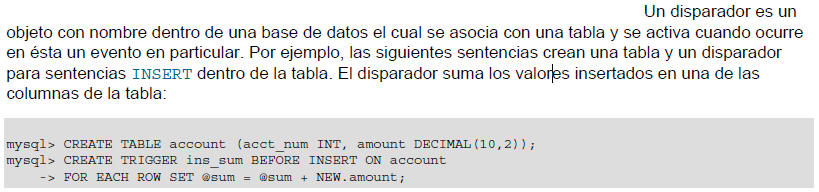
*}*

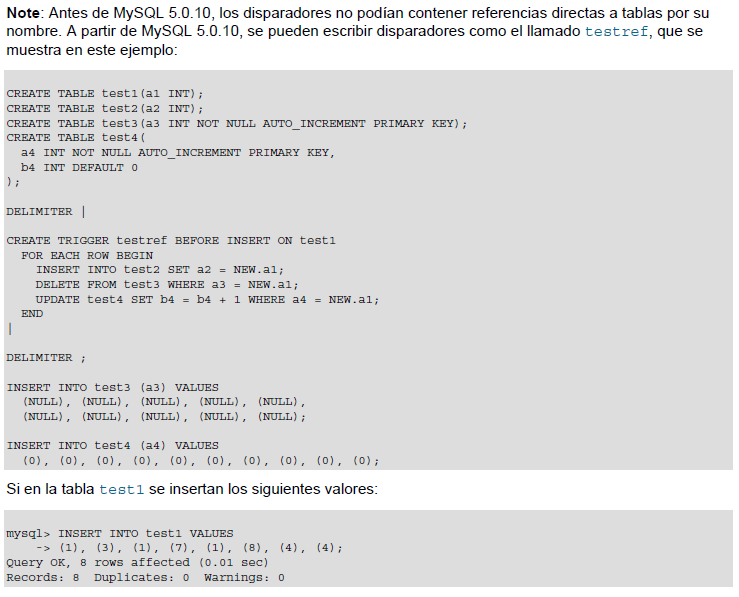
http://www.javahispano.org/portada/2015/9/14/resumen-del-funcionamiento-del-sistema-de-modulos-de-java-9.html

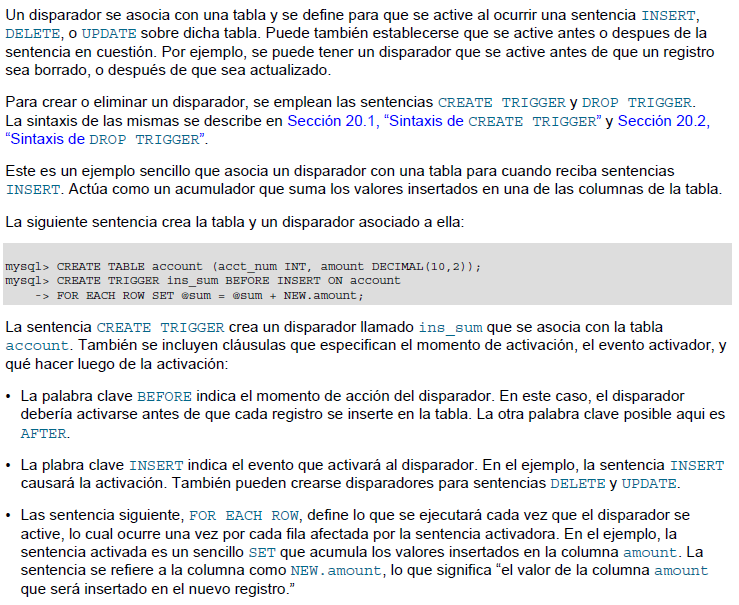
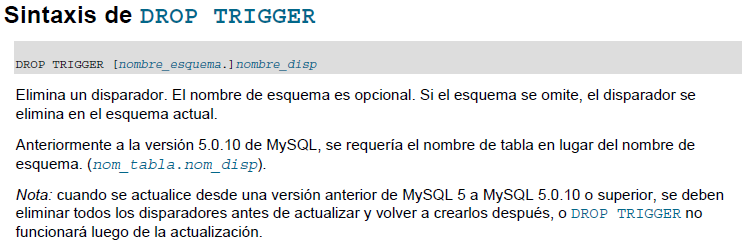
# Entity Bean

Lorem ipsum

# Triggers







Referencias:

https://downloads.mysql.com/docs/refman-5.0-es.a4.pdf

# Entity Factory

Lorem ipsum

# JPQL

## Que es:

Java Persistence Query Language (JPQL) es un lenguaje de consulta orientado a objetos independiente de la plataforma definido como parte de la especificación Java Persistence API (JPA).

## Para que sirve:

JPQL es usado para hacer consultas contra las entidades almacenadas en una base de datos relacional. Está inspirado en gran medida por [SQL](https://es.wikipedia.org/wiki/SQL), y sus consultas se asemejan a las consultas SQL en la sintaxis.

## Relación:

Opera con objetos entidad de JPA en lugar de hacerlo directamente con las tablas de la base de datos.

## Ejemplo:

Ejemplo de clases JPA, los getters y setters han sido omitidos por simplicidad.

@Entity

**public** **class** **Autor** {

@Id

**private** Integer id;

**private** String nombre;

**private** String apellidos;

@ManyToMany

**private** List<Libro> libros;

}

@Entity

**public** **class** **Libro** {

@Id

**private** Integer id;

**private** String titulo;

**private** String isbn;

@ManyToOne

**private** Editorial editorial;

@ManyToMany

**private** List<Autor> autores;

}

@Entity

**public** **class** **Editorial** {

@Id

**private** Integer id;

**private** String nombre;

**private** String direccion;

@OneToMany(mappedBy = "editorial")

**private** List<Libro> libros;

}

Dadas esas clases, una consulta simple para obtener la lista de todos los autores, ordenados alfabéticamente, sería:

**SELECT** a **FROM** Autor a **ORDER** **BY** a.nombre, a.apellidos

Para recuperar la lista de autores que han sido publicados alguna vez por XYZ Press:

**SELECT** **DISTINCT** a **FROM** Autor a **INNER** **JOIN** a.libros b **WHERE** b.editorial.nombre = 'XYZ Press'

JPQL admite parámetros con nombre, que comienzan por dos puntos (:). Se podría escribir una función que devuelve una lista de autores con el apellido indicado de la siguiente manera:

**import** **javax.persistence.EntityManager**;

**import** **javax.persistence.Query**;

**import** **org.apache.commons.lang.StringUtils**;

...

@SuppressWarnings("unchecked")

**public** List<Autor> getAutoresPorApellidos(String apellidos) {

String queryString = "SELECT a FROM Autor a " +

"WHERE :apellidos IS NULL OR LOWER(a.apellidos) = :apellidos";

Query query = getEntityManager().createQuery(queryString);

query.setParameter("apellidos", StringUtils.lowerCase(apellidos));

**return** query.getResultList();

}

https://docs.oracle.com/javaee/6/tutorial/doc/bnbtg.html

# XHTML

## Que es:

Es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

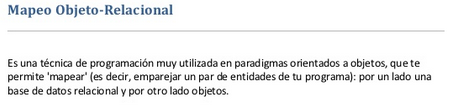
## Para que sirve:

* Se pueden incorporar elementos de distintos espacios de nombres XML (como MathML y Scalable Vector Graphics).
* Un navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que el parser puede ser mucho más sencillo.
* Como es XML se pueden utilizar fácilmente herramientas creadas para procesamiento de documentos XML genéricos (editores, XSLT, etc.).

# Temas más importantes.

Facelets  
Facelets es un framework basado en el servidor que permite definir la estructura general de las páginas (su *layout*) mediante plantillas. Se trata de un framework similar a Tiles, Velocity, Tapestry o Sitemesh. Facelets se adapta perfectamente al enfoque de JSF y se incorpora a la especificación en la última revisión 2.1. La estructura de la página se define utilizando las etiquetas Facelets y los componentes específicos que deben presentar los datos de la aplicación utilizando etiquetas JSF.

## Mapeo

  
 Una aplicación Web basada en JSF permite mapear peticiones HTTP a eventos y gestionar el estado de objetos.

Primefaces   
PrimeFaces es una librería de componentes visuales open source desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing.

POJO  
POJO son las iniciales de **“Plain Old Java Object”**, que puede interpretarse como “Un objeto Java Plano Antiguo”. Un POJO es una instancia de una clase que no extiende ni implementa nada en especial.

JPA  
Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java.   
Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

JSTL  
Java Server Pages Standard Tag library (JSTL) es un conjunto de librerías de etiquetas simples y estándares que encapsulan funcionalidad habitualmente utilizada en aplicaciones Web. Soportan funciones como iteraciones, procesamiento condicional, procesamiento XML, internacionalización, etc. Su uso está extendido debido sobre todo a que se integran en las páginas JSP de forma sencilla y limpia, además de proporcionar la mayoría de funcionalidades necesarias en un JSP.

EJB  
Un Enterprise Java Bean (EJB) es un componente de negocio Java Enterprise, y para su ejecución necesita un contenedor EJB/J2EE (JBoss, WAS, OAS, etc). El hecho de usar EJB’s te da acceso a los servicios del Contenedor EJB (manejo de transacciones, seguridad, persistencia, etc) que simplifican bastante la construcción de soluciones empresariales.