Definiciones

Propiedad refactor: renombra clases y cambia todo lo que tenga el ese nombre.

JVM CLASS LOADER pone las clases en memoria

GARBAGE COLLECTOR - sacar los objetos en memoria que no se usen

JDK: Tiene un compilador, interpretador y un generador de documentación

MODEL BUNDLE: Sirve para modificar los label del proyecto

**GENERIC APLICATION**

ERRORES:

- debe insertar la librería ORACLE.JDBC

- entrada y salida, revisar la sentencia de la conexión establecida.

SI ELEGIMOS GENERIC APLICATION LUEGO DEBEMOS ELEGIR ADF VIEW CONTROLLER

**LIBRERIAS IMPORTANTES PARA LA CONEXIÓN A LA BASE DE DATOS**

Selecciono el proyecto -> Project properties -> libraries and classpath y agregar la librería manualmente

Import java.sql.Connection;

Import java.sql.DriverManager;

Import java.sql.SQLException;

**CLASE JAVA TEORIA:**

SOP , SISTEM OUTPUT PRINTLN, crtl + enter

*Generate accessors*: crea los métodos GET Y SET para el encapsulamiento de los atributos

*FOREACH* : for( clase objeto : lista)

**VIEW CONTROLLER**

Definiciones:

**DATA CONTROLS**: encapsula lo que hay por detrás

Bind**:** son clases de java de la cual tiene requisitos para ser utilizados.

Invocamos el constructor de un bind, el constructor no puede recibir parámetros

Las validaciones se pueden hacer en las vistas y en las entidades

**REQUESTS** – Scope que mapea, le damos un valor y después nos devuelve un valor , dura un ciclo de vida de una página.

**SESSION** – Scope que MAPEA, dura la sesión http del usuario.

**SCOPE**: Es una estructura en memoria que tiene determinada duración , todos los objetos tienen que vivir en algún lado.

Put (key, valor), get (key)

**APLICATTION**: dura desde que se inicia la app hasta que termina, cada página está asociada a un componente.

**BEAN**: es una CLASE JAVA, tiene constructor atributos privados, los get y set.

**ValueChangeListener**: cuando un componente cambia de valor, se lo pasan a este manejador (AutoSubmit = true)

**ActionListener**: va a refrescar ese evento en particular del componente, usando también el parcial submit.

**Action**: Posteo de todos los datos y refrescos de la página Completa, le da una action al componente.

**Managed bean**: es donde se declara los backing bean para ser usado con el JSF managed facility

**Backing bean**: es una clase java que tiene asociados componentes y vive en el REQUEST.

Creo un backing en el cual se crea una Clase JAVA y pueda definir los componentes y bindearlos.

Cada componente se bindea con un backing bean y un método(el método le dará nombre al campo).

**Auto Submit**: setea los datos del componente inmediatamente y cambia el valor

**PARCIAL SUBMIT**: submitea algunos componentes

BINDING:

-Tipo

- A nivel componentes: por medio de java accedo a las propiedades Enlazando a algo,

- Evento: (Action, ActionListener, ValueChangeListener) a la espera del cambio del componente.

- A nivel Valor: el valor lo gestiona el framework. No necesita código extra.

View: solo para componentes, NO USAR JAMAS

PARCIAL TRIGERS: dispara varias acciones, un componente escuche a otro/s ej.: botones

INMEDIATE: saltea las validaciones

**PagDef**:

-El pagdef es la relación entre el modelo de negocio (entidades, vistas) y el fragmento, pagina.

depende de los parámetros que ejecuta en el módulo se lo voy a pasar al pagDef

**Data Bindings:** es un archivo donde indica los **pageDef** para cada **TaskFlow** y Fragmentos

**Fixed Outcome**: se agrega a donde se destina el control flow case

Agregar Parametros en TaskFlow

Overview->Parameters

Uso de clases para tipo de datos:

Oracle.jbo.domain.number

Java.lang.boolean

**- Cambiar label**

Utilizar select text Resource

Hacer un replace

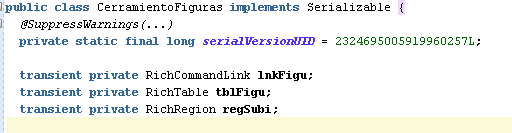
**- Replace in files**

-poner preview para visualizar antes del cambio

- match case distingue mayúsculas y minúsculas.

- match whole Word only , si encuentra la misma palabra con mas caracteres no lo reemplaza.

Transient : es un modificador que sirve para aclarar que no serialice los atributos.



**C**lientListener:

Escucha el lado del cliente, escuchando el evento de javascript

MaximumLength

Para adicionarle al largo se le puede poner + (un numero)

**LOGICA DE IMPLEMENTACION**

Replicación de una **integración** importa 3 archivos:

-la jspx

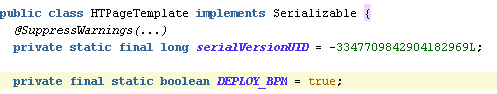
-pagDef.xml

-java asociado a la pagina

-BpmFlow corre el mediante el HTimplementacion (TestTemplate) y se conecta con SavePointTemplateFlow por el mismo test.

- Para correr el proyecto se corre BPMFLOW

- en caso que no corra el proyecto pasar la variable DEPLOY\_BPM a false de la clase HTPageTemplate.



Cambiar el HTimplementacion por TestTemplate2 en caso que no ejecute la aplicacion