**JEE**

////////////////////////////////////

Provide developers with a powerful set of APIs while shortening development time, reducing application complexity, and improving application performance

Java Community Process (JCP)

Java Enterprise Edition (Java EE)

Java Specification Requests (JSRs)

Uses a simplified programming model. XML deployment descriptors are optional. Instead, a developer can simply enter the information as an annotation directly into a Java source file, and the Java EE server will configure the component at deployment and runtime. These annotations are generally used to embed in a program data that would otherwise be furnished in a deployment descriptor. With annotations, you put the specification information in your code next to the program element affected

Enterprise JavaBeans (EJB)

/////////////////////////////////

La hora es <%= new java.util.Date() %>

Eclipse > File > New > Other > Dynamic web project

Se guardan los jsp en WebContent

Click der en el arch > Run as > Run on server

Checar Always use this server… > Siguiente

Error puerto ocupado:

Dentro de la carpeta donde está Tomcat > conf > server.xml >> buscar los puertos ocupados >> cambiarlos

Reiniciar el sv

/////////////////////////////////////////

Chat

Servidor.java

Public static void main(String[] args) {

MarcoServidor **mimarco** = new MarcoServidor();

**mimarco**.setDefaultCloseOperation(Jframe.EXIT\_ON\_CLOSE);

}

class MarcoServidor extends Jframe implements Runnable{

public MarcoServidor() {

setBound(1200, 300, 280, 350);

Jpanel milamina = new Jpanel();

milamina.setLayout(new BorderLayout());

areatexto = new JtextArea();

milamina.add(areatexto, BorderLayout.CENTER);

add(milamina);

Thread mihilo = new Thread(this);

mihilo.Start();

}

Private JtextField campo1, nick, ip;

Private JTextArea areatexto;

Private JtextArea campochat;

Private Jbutton miboton;

@Override

Public void run() {

try {

ServerSocket servidor = new ServerSocket(9999);

String nick, ip, mensaje;

PaqueteEnvio paquete\_recibido;

While (true) {

Socket misocket = servidor.accept();

//DataInputStream flujo\_entrada = new DataInputStream(misocket.getInputSteam());

//String mensaje\_texto = flujo\_entrada.readUTF();

//areatexto.append(“\n” + mensaje\_texto);

ObjectInputStream **paquete\_datos** = new ObjectInputStream(misocket.getInputStream());

paquete\_recibido = (PaqueteEnvio) **paquete\_datos**.readObject();

nick = paquete\_recibido.getNick();

ip = paquete\_recibido.getIp();

mensaje = paquete\_recibido.getMessage();

areatexto.append(“\n” + nick + “: “ + mensaje + “ para “ + ip);

Socket enviaDestinatario = new Socket(ip, 9090):

ObjectOutputStream paqueteReeenvio = new ObjectOutputStream(enviaDestinatario.getOutputStream());

paqueteReenvio.writeObject(paquete\_recibido);

paqueteReenvio.close();

enviaDesinatario.close();

misocket.close();

}

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

try {

Socket misocket = new Socket(“192.168.56.1”, 9999);

PaqueteEnvio **datos** = new PaqueteEnvio();

**datos**.setNick(nick.getText());

**datos**.setIp(ip.getText());

**datos**.setMensaje(campo1.getText());

ObjectOutputStream paquete\_datos = new ObjectOutputStream(misocket.getOutputStream());

Paquete\_datos.writeObject(**datos**);

misocket.close();

} catch (UnknownHostException e1) {

E1.printStackTrace();

} catch (IOException e1) {

System.out.println(e1.getMessage());

}

/////////////////////////////////////////////

Cliente.java

Public class Cliente {

Public static void main(String[] args) {

MarcoCliente mimarco = new MarcoCliente();

mimarco.setDefaultCloseOperation(Jframe.EXIT\_ON\_CLOSE);

}

}

Class MarcoCliente extends Jframe {

public MarcoCliente() {

setBounds(600, 300, 280, 350);

LaminaMarcoCliente milamina = new LaminaMarcoCliente();

add(milamina);

setVisible(true);

}

}

Class LaminamarcoCliente extends Jpanel implements Runnable {

Public LaminaMarcoCliente() {

nick = JtextField(5);

add(nick);

Jlabel texto = new Jlabel(“-CHAT-“);

add(texto);

ip = new JtextField(8);

add(ip);

campochat = new JtextArea(12, 20);

add(campochat);

campo1 = new JtextField(20);

add(campo1);

miboton = new Jbutton(“Enviar”);

EnviaTexto mievento = new EnviaTexto();

miboton.addActionListener(mievento);

add(miboton);

Thread mihilo = new Thread(this);

Mihilo.start();

}

@Override

Public void run() {

try {

ServerSocket servidor\_cliente = new ServerSocket(9090);

Socket cliente;

PaqueteEnvio paqueteRecibido;

while(true) {

cliente = servidor\_cliente.accept();

ObjectInputStream flujoentrada = new ObjectInputStream(cliente.getInputStream());

paqueteRecibido = (PaqueteEnvio) flujoentrada.readObject();

campochat.aped(“\n” + paqueteRecibido.getNick() + “: “ + paqueteRecibido.getMensaje());

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}

Class PaqueteEnvio implements Serializable {

Private String nick, ip, mensaje;

Poner getter y setter :

Eclipse > click derecho > Source > Generate Getters and Setters > Select All

}

}

private class EnviaTexto implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

campochat.append(“\n” + campo1.getText());

try {

Socket misocket = new Socket(“127.0.0.1”, 9999);

PaqueteEnvio datos = new PaqueteEnvio();

datos.setNick(nich.getText());

datos.setIp(ip.getText());

datos.setMensaje(campo1.getText());

ObjectOutputStream paquete\_datos = new ObjectoutputStream(misocket.getOutputStream());

paquete\_datos.writeObject(datos);

misocket.close();

} catch (Exception e) {

}

}

}

/////////////////////////

Chat uso

Hacer un jar ejecutable:

File > Export > Java > Runnable JAR file > Next > Launch configuration: dropdown -> Cliente – Sockets > Finish

//////////////////////////////////

Socket

Public static void main(String[] args) {

**MarcoServidor** mimarco = new **MarcoServidor**();

mimarco.setDefaultCloseOperation(Jframe.EXIT\_ON\_CLOSE);

}

class **MarcoServidor** extends Jframe implements Runnable{

public MarcoServidor() {

setBound(1200, 300, 280, 350);

Jpanel milamina = new Jpanel();

milamina.setLayout(new BorderLayout());

areatexto = new JtextArea();

milamina.add(areatexto, BorderLayout.CENTER);

add(milamina);

Thread mihilo = new Thread(this);

mihilo.Start();

}

Private JtextArea areatexto;

@Override

Public void run() {

try {

ServerSocket servidor = new ServerSocket(9999);

While (true) {

Socket miscocket = servidor.accept();

DataInputStream flujo\_entrada = new DataInputStream(misocket.getInputSteam());

String mensaje\_texto = flujo\_entrada.readUTF();

areatexto.append(“\n” + mensaje\_texto);

misocket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

try {

Socket misocket = new Socket(“192.168.56.1”, 9999);

DataOutputStream flujo\_salida = new DataOutputStream(misocket.getOutputStream());

flujo\_salida.writeUTF(campo1.getText());

flujo\_salida.close();

} catch (UnknownHostException e1) {

E1.printStackTrace();

} catch (IOException e1) {

System.out.println(e1.getMessage());

}

/////////////////////////////////////////////////////

Polymorphism means "**many forms**"

The purpose of nested classes is to group classes that **belong together**, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class:

OuterClass.**InnerClass** myInner = myOuter.new InnerClass();

Unlike a "regular" class, an inner class **can be** private or protected

class OuterClass {

int x = 10;

class InnerClass {

public int myInnerMethod() {

return x;

///////////////////////////////////////////

Abstraction is the process of hiding certain details and **showing only essential information**

abstract class Animal {

public **abstract** void animalSound();

public void sleep() {

To access the abstract class, it must be inherited **from another class**

Interface methods are **by default abstract and public**

Interface attributes are **by default public, static and final**

class DemoClass implements **FirstInterface**, **SecondInterface** {

//////////////////////////////////////////////////////////////////////

public class MyClass {

enum Level {

LOW,

MEDIUM,

HIGH

}

public static void main(String[] args) {

Level myVar = Level.MEDIUM;

System.out.println(myVar);

Salida: MEDIUM

for (Level myVar : Level.values()) {

enum constants are public, static and final (**unchangeable** - cannot be overridden).

An enum cannot be used to create objects, and it cannot extend other classes (**but it can** implement interfaces).

Use enums when you have values that you know aren't going to change, like month days, days, colors, deck of cards, etc.

//////////////////////////////////////////////////////////////////

**Stored procedures**

Permiten pasar parámetros a las sentencias sql

Previenen ataques de inyección sql

Tienen mejor rendimiento

Son precompiladas

Reutilizables

PreparedStatement sentencia = conexión.preaperStatement(“Select \* from productos where seccion = ? and paisDeOrigen = ?);

Sentencia.setString(1, ‘Deportes’);

ResultSet rs = sentencia.executeQuery();

try {

Connection miConexion = DriverManager.getConnection(“jdbc:mysql://localhost:3306/pruebas”, “root”, “”);

PreparedStatement miStatement = miConexion.preparedStatement(“Select NombreArticulo from Productos where”

+ “ seccion = ? and PaisDeOrigen = ?”);

miSentencia.setString(1, “deportes”);

miSentencia.setString(2, “USA”);

ResultSet rs = miSentencia.executeQuery();

While(rs.next()) {

System.out.println(rs.getString(1) + “ “ + rs.getString(2) + “ “ + rs.getString(3));

}

rs.close();

System.out.println(“Ejecución segunda consulta”);

System.out.println(“”);

miSentencia.setString(1, “deportes”);

miSentencia.setString(2, “USA”);

ResultSet rs = miSentencia.executeQuery();

While(rs.next()) {

System.out.println(rs.getString(1) + “ “ + rs.getString(2) + “ “ + rs.getString(3));

}

rs.close();

} catch(Exception e) {

System.out.println(“No conecta”);

}

///////////////////////////////////////

**JSP**

Expresiones:

<%= expresión Java %>

Scriplets:

<% varias líneas de código Java %>

Declaraciones:

<%! Declaración de variables o métodos %>

A mayúsculas: <%= new String(“juan”).toUpperCase() %>

<%

for (int i=0; i<10; i++) {

out.println(i);

}

%>

<%!

Private int resultado;

Public int metodoSuma(int num1, int num2) {

Resultado = num1 + num2;

}

%>

<%= metodoSuma(7, 5) %>

/////////////////////////////////

JSP

Carpeta Java Resources > src > New > Package

Arriba del html:

<%@ page import = “pildoras.com.CalculosMatematicos.\*” %>

**Request pertenece a javax.servlet.ServletRequest**

**Response pertenece a javax.servlet.ServletResponse**

**Session pertenece a javax.servlet.http.HttpSession**

**Out recursos para escribir contenido en una página web, pertenece a javax.servlet.jsp.JspWtriter**

Datos del navegador: <%= request.getHeader(“User-Agent”) %>

Idioma: <%= request.getLocale() %>

///////////////////////////////////

**JSP Formulario registro**

Formulario-registro.jsp

<form action = “Registro.jsp”>

Nombre: <input type = “text” name = “nombre”><br />

Apellido: <input type=”text” name=”apellido”><br/>

<input type=”submit” value=”Registrar”>

</form>

**Registro.jsp**

Nombre: <%= request.getParameter(“nombre”) %> <br/>

Apellido: <%=request.getParameter(“apellido”) %>

**Registro-Personas.jsp**

<jsp:include page=”Formulario-Registro.html”></jsp:include><br/>

/////////////////////////////////////////////////////////////////////////////

**JSP formulario**

<form action=”Recoge\_datos.jsp” method=”post”>

<table width=”35%” border=”o”>

<tr>

<td width=”13%”><label for=”nombre”>Nombre: </label></td>

<td width=”87%”><input type=”text” name=”nombre” id=”nombre”></td>

</tr>

<tr>

<td><label for=”pais”>País: </label></td>

<td>

<select name=”pais” id=”pais”>

<option>México</option>

<option>Colombia</option>

</select>

</td>

</tr>

<tr>

<td>

<label>

<input type=”radio” name=”tecnologias” value=”java” id=”tecnologias\_0”>

</label>

</td>

</tr>

<tr>

<td colspan=”2” align=”center”>

<input type=”submit” name=”button” value=”Enviar” id=”button”>

</td>

</tr>

</table>

</form>

Agregar el driver:

Click derecho en el proy > Properties > Java Build Path > Librarries > Add External Jars > seleccionar el conector de mysql > no darle OK, ira a Deployment assembly > Add > Java Build Path Entries > Next

**Recoge\_datos.jsp**

<%@ page import = ”java.sql.\*” %>

<%

String nombre = request.getParameter(“nombre”);

String pais = request.getParameter(“pais”);

String tecnologias = request.getParameter(“tecnologias”);

Class.forName(“com.mysql.jdbc.Driver”);

try {

Connection miConexion = java.sql.DriverManager.getConnection(“jdbc:mysql://localhost:3306/proyecto\_jsp”, “root”, “”);

Statement miStatement = miConexion.createStatement();

//comilla sencilla al inicio en values

String instruccionSql = “Insert into Usuarios (Nombre, Pais, Tecnologias) values (‘” + nombre + “’, ‘” + pais + “’, ‘” + tecnologias “’);

miStatement.executeUpdate(instruccionSql);

out.println(“Registrado con éxito”);

} catch (Exception e) {

out.println(“Ha habido un error”);

}

%>

////////////////////////////////////

**JSP Comprueba\_Usuario.jsp**

**Recoge\_datos.jsp**

<%@ page import = ”java.sql.\*” %>

<%

String usuario = request.getParameter(“usuario”);

String contra = request.getParameter(“contra”);

Class.forName(“com.mysql.jdbc.Driver”);

try {

Connection miConexion = java.sql.DriverManager.getConnection(“jdbc:mysql://localhost:3306/proyecto\_jsp”, “root”, “”);

PreparedStatement c\_preparada = miConexion.preparedStatement(“Select \* form Usuarios where Usuario = ? and Contrasena = ?”);

c\_preparada.setString(1, usuario);

c\_preparada.setString(2, contra);

ResultSet miResulset = c\_preparada.executeQuery();

If(miResultset.absolute(1)) out.println(“Usuario autorizado”);

else out.println(“No hay usuarios con esos datos”);

} catch (Exception e) {

out.println(“Ha habido un error”);

}

%>

**Formulario\_Login.html**

<form action=”Comprueba\_usuario.jsp” method=”post”>

<table width=”35%” border=”o”>

<tr>

<td width=”13%”><label for=”usuario”>Usuario: </label></td>

<td width=”87%”><input type=”text” name=”usuario” id=”usuario”></td>

</tr>

<tr>

<td width=”13%”><label for=”contra”>Contra: </label></td>

<td width=”87%”><input type=”text” name=” contra” id=” contra”></td>

</tr>

<tr>

<td colspan=”2” align=”center”>

<input type=”submit” name=”button” value=”Login” id=”button”>

</td>

</tr>

</table>

</form>

/////////////////////////////////////

**JSP sesiones**

Lista\_Compra.jsp

<%@ page import=”java.util.\*” %>

<form action=”Lista\_Compra.jsp”>

<table width=”35%” border=”o”>

<tr>

<td width=”13%”>

<label>

<input type=”checkbox” name=”articulos” value=”agua”>

Agua

</label>

</td>

</tr>

<tr>

<td width=”13%”>

<label>

<input type=”checkbox” name=”articulos” value=”leche”>

Leche

</label>

</td>

</tr>

<td colspan=”2” align=”center”>

<input type=”submit” name=”button” value=”Enviar” id=”button”>

</td>

</tr>

</table>

</form>

getParameter devuelve un string

getParameterValues devuelve un array

setAttribute(String->nombre de la sesión, Object)

getAttribute devuelve el objeto grabado en la sesión de tipo Object (String)

<ul>

<%

List<String> ListaElementos ListaElementos = (List<String>)session.getAttribute(“misElementos”);;

if(ListaElementos == null) {

ListaElementos = enw ArrayList<String>();

Session.setAttribute(“misElementos”, ListaElementos);

}

String[] elementos = request.getParameterValues(“articulos”);

If(elementos ¡= null) {

for(String elemTemp: elementos) {

ListaElementos.add(elemTemp);

}

}

for(String elemTemp: ListaElementos) {

out.println(“<li>” + elemTemp + “</li>”);

}

%>

</ul>

<https://www.youtube.com/watch?v=S4PTyGHjQOM&list=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2lk&index=238>

//////////////////////////////////////////////////

**JSP cookies**

Archivo de texto que se almacena en el ordenador durante la navegación

Sirven para personalizar un sitio

Hay que informar que se van a utilizar cookies

Write:

Class Cookie

Interface HttpServletResponse

setMaxAge(int segundos)

addCookie(Cookie lacookie)

Read:

Interface HttpServletRequest

Class Cookie

getCookies()

getValues()

getName()

Formulario\_envio\_preferencias\_viaje.html

<form action=”recibe\_peticion.jsp”>

<select name=”ciudad\_favorita”>

<option>Madrid</option>

<option>Barcelona</option>

</select>

<input type=”submit” value=”Enviar”/>

</form>

//////////////////////////////////////////////////

Cookie pertenece a javax.servlet.http

Constructor:

Cookie(String->nombre, String->valor)

Void setMaxAge(int) vida de la cookie en segundos

Void addCookie(Cookie)

recibe\_peticion.jsp

<%@ page language=”java” contenType=”text/html; charset=ISO-8859-1” pageEncoding=”ISO-8859-1” %>

<meta http-equiv=”Content-Type” content=”text/html; charset=ISO-8859\*-1”>

<%

//Leyendo datos del formulario

String ciudad\_favorita = request.getParameter(“ciudad\_favorita”);

Cookie laCookie = new Cookie(“agencia\_viajes.ciudad\_favorita”, ciudad\_favorita);

laCookie.setMaxAge(365\*24\*60\*60); //1 año

response.addCookie(laCookie);

%>

<a href=”agencia\_viajes.jsp”>Ir a la agencia de viajes</a>

/////////////////////////////////////////////////////////////////////

Clase HttpServletRequest

Cookie[] getCookies()

String getName()

agencia\_viajes.jsp

<%

//Por defecto

String viudad\_favorita = “Madrid”;

Cookie[] lasCookies = request.getCookies();

If(lasCookies ¡= null) {

for(Cookie cookie\_temporal: lasCookies) {

if(“agencia\_viajes.ciudad\_favorita”.equals(cookie\_temporal.getname())) {

ciudad\_favorita = cookie\_tempoal.getValue();

break;

}

}

}

%>

Vuelos a <%= ciudad\_favorita %>

//////////////////////////////////////////////////

**JSP Tags**

Similares a xml

Hacen más simple la programación, lectura y modificación de páginas JSP

Separan el código del html

Jsp tags -> Jsp CustomTags (hechas por el programador), Jstl standard tag library

jstl -> Core (bucles, condicionales…), Formato (fecha, moneda), Xml (manipulación de xml), Sql (No se recomienda, usar mejor Beans)

Descargar la librería jstl > agregar a Eclipse > importarla:

<% taglib prefix=”c” uri=http://java.sun.com/jsp/jstl/core %>

jstl java api

Crear el paquete pildorasinformaticas.com.jsptags

Crear la clase Empleado

Empleado.java

public class Empleado {

private String nombre;

private String apellido;

private String puesto;

private Double salario;

Crear getters y setters

public Empleado(Strin nombre, String apellido, String puesto, double salario) {

super();

this.nombre = nombre;

this.apellido = apellido;

this.puesto = puesto;

this.salario = salario;

}

}

///////////////////////////////

RecorreEmpleados.jsp

<%@ taglib uri=<http://java.sun.com/jsp/jstl/core> prefix=”c” %>

<%@ page import=”java.util.\*, java.sql.\*, pildorasinformaticas.com.jsptags.Empleado” %>

<%

ArrayList<Empleado> datos = new ArrayList<>();

Class.forName(“com.mysql.jdbc.Driver”);

try {

Connection miConexion = DriverManager.getConnection(“jdbc:mysql://localhost:3306/proyecto\_jsp”, “root”, “”);

Statement miStatement = miConexion.createStatement();

String instruccionSql = “Select \* from Empleados”;

Resulset rs = miStatement.executeQuery(instruccionSql);

while(rs.next()) {

datos.add(new Empleado(rs.getString(1), rs.getString(2), rs.getString(3), rs.getDouble(4)));

}

rs.close();

miConexion.close();

} catch (Exeption e) {

pageContext.setAttribute(“losEmpleado”, datos);

}

%>

<html>

<table border=1>

<tr><td>Nombre</td>

<td>Apellido</td>

<td>Puesto</td>

<td>Salario</td>

<td>Bonus</td>

</tr>

<c:forEach var=”EmpTemp” items=”${losEmpleados}”>

<tr>

<td> ${EmpTemp.nombre}</td>

<td> ${EmpTemp.apellidos}</td>

<td> ${EmpTemp.puesto} </td>

<td> ${EmpTemp.salario} </td>

<td>

<c:if test=”${EmTemp.salario < 40000}”>5000</c:if>

<c:if test=”${EmTemp.salario >= 40000}”>----</c:if>

</td>

</tr>

</c:forEach>

<c:forEach var=”EmpTemp” items=”${losEmpleados}”>

<tr>

<td> ${EmpTemp.nombre}</td>

<td> ${EmpTemp.apellidos}</td>

<td> ${EmpTemp.puesto} </td>

<td> ${EmpTemp.salario} </td>

<td>

<c:if test=”${EmTemp.salario < 40000}”>

${EmpTemp.salario + 5000}

</c:if>

<c:if test=”${EmTemp.salario >= 40000}”>

${EmpTemp.salario}

</c:if>

</td>

</tr>

</c:forEach>

<c:forEach var=”EmpTemp” items=”${losEmpleados}”>

<tr>

<td> ${EmpTemp.nombre}</td>

<td> ${EmpTemp.apellidos}</td>

<td> ${EmpTemp.puesto} </td>

<td> ${EmpTemp.salario} </td>

<td>

<c:choose>

<c:when test = “${EmpTemp.salario < 40000 && EmpTemp.salario > 20000 }”>

${EmpTemp.salario + 5000}

</c:when>

<c:otherwise>

${EmpTemp.salario}

</c:otherwise>

</c:choose>

</td>

</tr>

</c:forEach>

</table>

</html>

//////////////////////////////////////

EjemploFunctions.jsp

<%@ taglib uri=<http://java.sun.com/jsp/jstl/funstions> prefix=”fn” %>

<%@ taglib uri=<http://java.sun.com/jsp/jstl/core> prefix=”c” %>

<body>

<c:set var=”datos” value=”Ana, Lopez, Directora, 75000”/>

<c:set var=”datosArray” value=”${fn:split(datos, ‘,’)}”/>

<input type=”text” value=”${datosArray[0]}”><br/>

<input type=”text” value=”${datosArray[1]}”><br/>

<input type=”text” value=”${datosArray[2]}”><br/>

<input type=”text” value=”${datosArray[3]}”><br/>

</body>

////////////////////////////////////////////////////

**MVC**

Se hace un paquete para el controlador, uno para la vista, otro para el modelo y no llamado principal que es donde iniciará el programa

Marco\_Aplicacion2.java

import java.awt.\*;

import javax.swing.\*;

import controlador.\*;

Public class Marco\_Aplicacion2 extends JFrame {

Public Marco\_Aplicacion2() {

setTitle(“Consulta BBDD”);

setBounds(500, 300, 400, 400);

Jpanel menus = new Jpanel();

menus.setLayout(new FlowLayout());

**secciones** = new JcomboBox();

**secciones**.setEditable(false);

**secciones**.addItem(“Todos”);

paises = new JcomboBox();

paises.setEditable(false);

paises.addItem(“Todos”);

resultado = new JtextArea(4.50);

resultado.setEditable(false);

add(resultado);

menus.add(**secciones**);

menus.add(paises);

add(menus, BorderLayout.NORTH);

add(resultado, BoderLayout.CENTER);

Jbutton botonConsulta = new Jbutton(“Consulta”);

Add(botonConsulta, BorderLayout.SOUTH);

addWindowListener(new ControladorCargaSecciones(this));

}

public JComboBox **secciones**;

private JComboBox paises;

private JTextArea resultado;

}

//////////////////////////////////////

En el paquete Principal, nva clase:

Ejecuta\_Modelo\_Vista\_Controlador.java

Import vista.\*;

public class Ejecuta\_Modelo\_Vista\_Controlador {

public static void main(String[] args) {

Marco\_Aplicacion2 mimarco = new Marco\_Aplicacion2();

mimarco.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mimarco.setVisible(true);

}

}

/////////////////////

Dentro del paquete controlador:

Conexión.java

import java.sql.\*;

public class Conexion {

Connection miConexion = null;

public Conexion() {

}

public Connection dameConexion() {

try {

miConexion = DriverManager.getConnection(“jdbc:mysql://localhost:3306/curso\_sql”, “root”, “”);

} catch (Exception e) {

}

return miConexion;

}

}

////////////////////////////

En el paquete modelo:

Productos.java

public class Productos {

//Encapsular

public Productos() {

nArticulo = “”;

seccion = ””;

precio = “”;

pOrigen = “”;

}

private String nArticulo;

private String seccion;

private String precio;

private String pOrigen;

Poner getters y setters:

Click derecho en el espacio de trabajo > Source > Generate Getters and Setters >> Select All

}

/////////////////////////////////

En el paq modelo:

CargaSecciones.java

import ajva.sql.\*;

public class CargaSecciones {

Conexion miConexion;

public Resulset rs;

public CargaSecciones() {

miConexion = new Conexion();

}

public String ejecutaConsultas() {

Productos miProducto = null;

Connection **accesoBBDD** = miConexion.dameConexion();

try {

Statement secciones = **accesoBBDD**.createStatement();

rs = secciones.executeQuery(“Select Distinctrow Seccion from Productos”);

miProducto = new Productos();

miProducto.setSeccion(rs.getString(1));

rs.close();

} catch (Exeption e) {

}

return miProducto.getSeccion();

}

Otra opción:

public ResulSet ejecutaConsultas() {

Connection **accesoBBDD** = miConexion.dameConexion();

try {

Statement secciones = **accesoBBDD**.createStatement();

return rs = secciones.executeQuery(“Select Distinctrow Seccion from Productos”);

} catch (Exeption e) {

}

return null;

}

}

/////////////////////////////////////////////////

En el paquete controlador

**ControladorCargaSecciones.java**

import java.awt.event.\*;

import modelo.\*;

import vista.\*;

public class ControladorCargaSecciones extends WindowAdapter {

CargaSecciones obj = new CargaSecciones();

private marco\_Aplicacion2 = elmarco;

public ControladorCargaSecciones(Marco\_Aplicacion2 elmarco) {

this.elmarco = elmarco;

}

public void windowOpened(WindowEvent e) {

obj.ejecutaConsultas();

try{

while(obj.rs.next()) {

elmarco.secciones.addItem(obj.rs.getString(1));

}

} catch (Exception e) {

e2.printStackTrace();

}

}

}

/////////////////////////////////////////////////