

R



CORPORACIÓN
UNIVERSITARIA
REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

**FACULTAD DE CIENCIAS BÁSICAS E
INGENIERÍA**
Ingeniería de Sistemas
Asignatura: Lenguaje de Programación III

Dirección de Educación a Distancia y Virtual

Este material es propiedad de la Corporación Universitaria Remington (CUR),
para los estudiantes de la CUR en todo el país.

2013

CRÉDITOS



El módulo de estudio de la asignatura Lenguaje de Programación III del Programa Ingeniería de Sistemas es propiedad de la Corporación Universitaria Remington. Las imágenes fueron tomadas de diferentes fuentes que se relacionan en los derechos de autor y las citas en la bibliografía. El contenido del módulo está protegido por las leyes de derechos de autor que rigen al país.

Este material tiene fines educativos y no puede usarse con propósitos económicos o comerciales.

AUTOR

Edgar Eusebio López Murillo

Especialista Gerencia en sistemas de información

Ingeniero de sistemas, Tecnólogo en sistemas

Docente Investigador Corporación Universitaria Remington

Desarrollo de software Informática educativa

Tema: Lógica de programación orientada a objeto Tasisdev

edgar.lopez@remington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

Actualizaciones: fueron creadas a través de animaciones de repaso, ejercicios de autoevaluación interactivos, mapas conceptuales y pruebas iniciales por:

César Augusto Jaramillo Henao

Ingeniero de Sistemas

Docente universitario desde 1996

Cesar.jaramillo@Remington.edu.co

RESPONSABLES

Director de la Facultad de Ciencias Básicas e Ingeniería

Dr. Jorge Mauricio Sepúlveda Castaño

jsepulveda@remington.edu.co

Tomás Vásquez Uribe

Director Educación a Distancia y Virtual

tvasquez@remington.edu.co

Angélica Ricaurte Avendaño

Coordinadora de Remington Virtual (CUR-Virtual)

rricaurte@remington.edu.co

GRUPO DE APOYO

Personal de la Unidad Remington Virtual (CUR-Virtual)

EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011. Segunda versión Marzo 2012

Derechos Reservados



Esta obra es publicada bajo la licencia Creative Commons. Reconocimiento-No Comercial-Compartir Igual 2.5 Colombia.

TABLA DE CONTENIDO

1.	MAPA DE LA ASIGNATURA.....	6
2.	PLATAFORMA J2EE	7
2.1.	Relacion de conceptos	8
2.2.	Prueba Inicial	9
2.3.	J2EE APIS	11
2.4.	Arquitectura de tres capas.....	16
2.5.	Modelo vista control MVC	20
3.	BASES DE DATOS CON JDBC	27
3.1.	Relacion de conceptos	28
3.2.	Prueba Inicial	29
3.3.	Bases de datos con JDBC.....	31
3.4.	El Sistema manejador de base de datos SGDB	58
3.5.	La implementación RMI	70
4.	SERVLET	77
4.1.	Relacion de conceptos	78
4.2.	Prueba Inicial	79
4.3.	Servlet	81
4.4.	Java Server Pages	86
4.5.	Móvil J2ME.....	95
5.	PROGRAMACIÓN Y TECNOLOGÍA	105
5.1.	Relacion de conceptos	106
5.2.	Prueba Inicial	107
5.3.	Struts.....	108
5.4.	JSF	120
5.5.	Tomcat	130
5.6.	Integrando Tomcat y Apache.....	135

6.	PISTAS DE APRENDIZAJE	152
7.	GLOSARIO	158
8.	FUENTES BIBLIOGRÁFICAS	160
8.1.1.	Referencias	160
8.1.2.	Internet	161
8.1.3.	Trabajos citados	162
9.	BIBLIOGRAFÍA	164

1. MAPA DE LA ASIGNATURA

LENGUAJE DE PROGRAMACIÓN III

PROPÓSITO GENERAL DEL MÓDULO

Este módulo apropiado para estudiantes de ingeniería de sistemas y afines como los de profesionalización de ingeniería de sistemas. Pretende ampliar el conocimiento y la practica básica en el desarrollo de aplicación de servicio Web por medio del lenguaje de programación para diferentes plataformas, empleando persistencia y a nivel de cliente servidor como despliegue con manejo dinámico.

OBJETIVO GENERAL

Desarrollar ambientes de mono o multiplataforma para aplicaciones de servicio Web persistentes o no, empleando tecnología cliente servidor, de manera robusta, dinámica, ajustada a las peticiones del mercado y a su hospedaje, por un periodo específico

OBJETIVOS ESPECÍFICOS

- ❑ Desarrollar aplicaciones Web utilizando algunas de las tecnologías de la plataforma J2EE.
- ❑ Implementar la persistencia de los Datos en la tecnología JDBC en base a capas de forma segura.
- ❑ Proporcionar los conceptos básicos para la aplicación y comunicación de los Servlet.
- ❑ Desarrollar la arquitectura de alojamiento Tomcat, Tomcat –apache, en los servicios Web.

UNIDAD 1

Desarrollo de aplicaciones Web utilizando algunas de las tecnologías de la plataforma J2EE.

UNIDAD 2

Implementación de la persistencia de los Datos en la tecnología JDBC en base a capas de forma segura.

UNIDAD 3

Proporciona los conceptos básicos para la aplicación y comunicación de los Servlet.

UNIDAD 4

Desarrollo de la arquitectura de alojamiento Tomcat, Tomcat –apache, en los servicios Web

2. PLATAFORMA J2EE

<http://www.youtube.com/watch?v=KocrgtaJOMA>

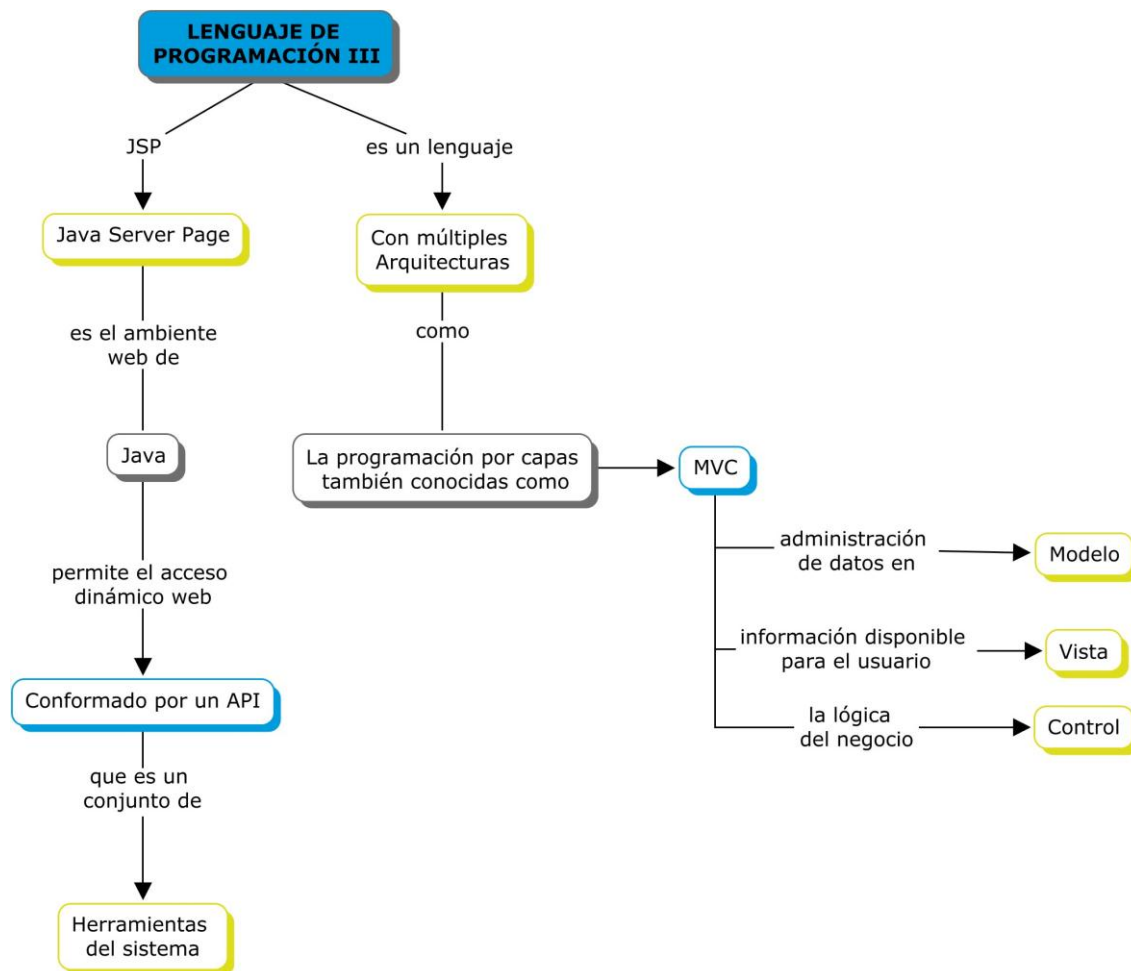


The screenshot shows a YouTube video player with a purple-themed interface. The video title is "Tutorial Java-web Servlet y JSP". The description states: "En este tutorial aprenderemos como usar java para producir paginas webs. Esto es conocido como **'java del lado del servidor'** y esta basado principalmente en las tecnologías conocidas como **Servlet y JSP Java Server Pages**."

Below the description is a diagram illustrating the architecture of a web application. It shows a "Browser o cliente http" (Browser or http client) connected to an "Internet" cloud. The cloud is connected to a "Servidor Http" (Http server) box, which contains a "Programa Java" (Java program). The "Programa Java" is connected to a "BD" (Database) cylinder. The diagram is labeled "Importancia" (Importance).

Imagen relacionada de video de YouTube

2.1. Relacion de conceptos



OBJETIVO GENERAL

Desarrollar aplicaciones Web utilizando algunas de las tecnologías de la plataforma J2EE.

OBJETIVOS ESPECÍFICOS

- ❑ Describir las tecnologías para la utilización en las aplicaciones J2EE.
- ❑ Diferenciar las capas de programación requeridos para la funcionalidad correspondiente a la arquitectura.
- ❑ Desarrollar las componentes tecnológicas para la combinación del modelo vista control del lado del cliente.

2.2. Prueba Inicial

Explorar los conceptos previos para el funcionamiento con las APIs, los procesos por capas y el MVC proporcionados por la tecnología J2EE (Camou Riveroll & Keogh, 2004).

Se validan los preconceptos de las APIs para iniciar al estudiante en la tecnología para la plataforma J2EE (Art, Brian, & Randy, 2004)

Seleccione la respuesta correcta a las siguientes preguntas según su conocimiento previo:

1. La tecnología del lenguaje java se basa en una plataforma mas conocida como:
 - a) El COMPILADOR
 - b) El INTERPRETE
 - c) Los APIs
 - d) La MVJ
 - e) El JDBC

2. Desde la visión del programador que es un Framework?
 - a) Es el diseño de código de java.
 - b) Es el diseño de código de C#.
 - c) Es el diseño de un patrón (molde) que utiliza otra aplicación.
 - d) Es el diseño de un patrón (molde) que emplea un compilador
 - e) Es el diseño de un patrón (molde) que facilita el uso de otra aplicación.

3. La API(interfaz de programación de aplicaciones (application program interface)) desde su conocimiento considera que :
 - a) Es una instrucción
 - b) Es un conjunto de paquetes
 - c) Es un conjunto de paquetes que son útiles para la elaboración de aplicación
 - d) Es un conjunto de paquetes que son útiles para la elaboración de aplicación de tercera dimensión.
 - e) Es un conjunto de paquetes clases que son útiles para la elaboración de aplicación.

4. Un desarrollador de software caracteriza un modelo de tres capas por tener:
 - a) Una Presentación, Aplicación, Repositorio
 - b) Una Presentación, Repositorio
 - c) Una Presentación, Aplicación, Dominio de la aplicación,
 - d) Una Aplicación, Dominio de la aplicación, Repositorio
 - e) Una Aplicación, Dominio de la aplicación, Presentación

5. El MVC permite al desarrollador de software separar su aplicación en tres elementos a saber :
 - a) La lógica de programación, la presentación en la impresora, el manejador de eventos
 - b) Los operadores, la presentación en pantalla ,a el manejador de eventos
 - c) La lógica, la presentación en pantalla, el manejador de sucesos.
 - d) La lógica , la presentación en pantalla, el manejador de eventos
 - e) La solicitud , la presentación de salida, el manejador de eventos

6. Los Apis son diferentes clases agrupadas que contienen componentes y contenedores los cuales son llamados:
 - a) Paquetes

2.3. J2EE APIS

Son especificaciones tecnológicas realizadas por JCP (java community Process) Procesos comunitarios de especificaciones, dirigido por Sun Microsystems. Este maneja una serie de paquetes de gran utilidad en las aplicaciones. (Maassen & Stelting, 2004) En las aplicaciones se abstraen el dominio de la función ubicándolo en el primer cuadrante, como el resultado requerido. Algunas de las librerías aplicadas Java Servlet (paginas dinámicas), EJB (persistencia automática) y JavaBeans (almacenamiento temporal)

Definición J2EE APIS

El J2ee se caracteriza por su aplicabilidad en entornos empresariales sobre todo cuando se procesan en red de manera distribuida o remota, dicho de otra forma un servicio web. Basado en tecnologías fundamentales como los Servlet y los JSP que se describen en los temas posteriores al igual que los Servidores su entorno de desarrollo es Jcreator y/ o eclipse, netBeans.

J2EE (Java 2 Enterprise Edition) adiciona los elementos de comunicación para desarrollo empresarial (flujo de datos almacenamiento permanente, redes entre otros) esto es justificado por ser cada elemento independiente en su implementación y requeridas solo al momento de usarlas encajando en su estructura objetual.

Arquitectura J2EE APIS

Las aplicaciones realizadas en J2EE se pueden dividir desde 2, 3 o más capas.

En la primera capa es donde se encuentran las interfaces como paginas Jsp, Servlet y Applet. La segunda capa se encuentra los componentes EJB, lo Web Services y toda la lógica de negocio. La última capa es para acceder a Bases de Datos. Cada una de estas capas se pueden subdividir en sub capas (http://java.ciberaula.com/articulo/disenio_patrones_j2ee/, 2003)

Java 2 Enterprise Edition es la arquitectura creada por Sun para el desarrollo de todo tipo de aplicaciones para empresas y usuarios en general. Sun lo define como un estándar para el desarrollo de aplicaciones empresariales multicapa. A diferencia de la plataforma .NET (Framework de .NET Es un conjunto de servicios de programación diseñados para simplificar, el desarrollo de aplicaciones en el entorno altamente distribuido de Internet).

J2EE solamente soporta el lenguaje Java. Las aplicaciones Java están típicamente compiladas en un lenguaje intermedio llamado bytecode, que es normalmente interpretado o compilado a código nativo mediante la maquina virtual de java, está JVM es una de las piezas fundamentales de la

plataforma Java. Básicamente se sitúa en un nivel superior al hardware del sistema, y este actúa como un puente que entiende tanto el bytecode, como el sistema sobre el que se pretende ejecutar tomado de operativos (<http://es.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET->, 2003) . Cuando se realizan aplicaciones J2EE por lo general se crean extensiones de archivos como Jar, War, Ear, Rar. Para movilizarlos y ejecutarlo en otros sistemas operativos

Arquitectura J2EE

Las razones que empujan a la creación de la plataforma JEE según (Lago, 2007) quien establece la eficiencia de la programación por medio de la estandarización de los procesos de desarrollo en cada una de las capas (cliente, servidor web, entre otras). Siendo indispensable las herramientas para este caso de la web los servlets, páginas JSP, HTML, Javascript, CGI entre otros. De igual modo la contingencia para cuando se este creciendo la clientela ampliar la transabilidad por medio de multiples maquina y servidores.

Las aplicaciones realizadas en J2EE se pueden dividir desde 2, 3 o más capas. En la primera capa es donde se encuentran las interfaces como paginas Jsp, Servlet , HTML, Applet, aplicaciones Java.

La segunda capa contiene subcapas (el contenedor web y el contenedor EJB) encontramos componentes EJB, lo Web Services y toda la lógica de negocio.

La última capa es para acceder a Bases de Datos, ERP, EIS.

Generalmente los proyectos de servicios web realizado en estos IDE generan una extensión (.WAR) que contiene toda la aplicación; este archivo se puede transportar a diferentes sistemas operativos y es posible desplegarlo para ser accedido mediante un cliente, utilizando algún servidor de aplicación que soporte las característica con que se creó el servicio web.

Comparando esta plataforma con la .NET la decisión esta en los desarrolladores o en la organización que pretenda implementar los servicios web, aun que Java lleve una ventana en su entorno de seguridad.

Al implementar servicios web J2EE y .NET y consumirlos desarrollando clientes en ambas plataformas, se comprobó en efecto que los servicios web permiten la interoperabilidad. Por otro lado, Sun y Microsoft están trabajando mancomunado en algunos aspectos específicos, para aumentar el nivel de interoperabilidad entre las dos.

Los Contenedores servidores J2EE APIS

Los contenedores incluyen descriptores de despliegue archivos XML encargados de configurar el entorno de ejecución: rutas de acceso a aplicaciones, control de transacciones, parámetros de inicialización, entre otros.

Las de persistencia de uso empresarial JDBC es el API para acceso a GBDR desde Java.

La JTA API para manejo de transacciones a traves de sistemas heterogeneos.

JNDI API para acceso a servicios de nombres y directorios.

JMS es el API para el envio y recepción de mensajes por medio de sistemas de mensajería empresarial como IBM MQ Series.

JavaMail es el API para envio y recepción de email.

Java IDL es el API para llamar a servicios CORBA tomado (Microsystem, 2007).

Servlet/JSP Contenedor de aplicaciones el mas común Tomcat .

Java Beans contiene aplicativos Java de interacción con bases de datos

Se presentan algunos servidores

El Apache para HTTP (también denominado servidor Web o servidor de páginas). Un ejemplo, el servidor Apache.

La descripción deTomcan se vera en el tema correspondiente

Las Capas de la arquitectura J2EE APIS

Se especifican de manera lógica los diferentes servicios realizados en la maquina, para este caso cuatro capas las cuales encajan dentro del tema anterior asi:

La del cliente o la capa de presentación o también llamada aplicación, dado que se encuentra básicamente en el ambiente básico java o en el básico applets quien a su vez incluye al html o xml.

La del servicio web se encarga de establecer el puente entre el cliente y las capas aquí se incluyen los servlet y los jsp.

La capa de java Bean que involucra varias aplicaciones para el flujo de los datos específicamente de carácter permanente están los EJB.

La capa de sistemas de información se da a nivel empresarial.

Un ejemplo J2EE APIS

Un ejemplo de servlet se presenta a continuación. Para empezar a familiarizarnos con lo que es un servlet vamos a ver uno en el que se crea una página que contiene el saludo de "Bienvenido al mundo de J2ee" Tomado de (Microsystem, 2007).

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Saludo extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Bienvenido al primer servlet</title></head>");
        out.println("<body          bgcolor=\"#FFFF9D\"><FONT          color=\"#000080\"
FACE=\"Arial,Helvetica,Times\" SIZE=2>"+
                "<CENTER><H3>Servlets</H3></CENTER>");
        Date d = new Date();
        out.println("<HR><p>i Bienvenido al mundo de J2ee!. Fecha y hora: " + d.toString() + ". Esto es
una aplicación Java.</p>");
        out.println("</body></font></html>");
    }
}
```

La evaluación de J2EE Apis (1 valor de 7.7%)

Describir conceptos obtenidos en J2EE.

Se validan los conceptos de las Apis al terminar la unidad de la plataforma J2EE.

Describe la respuesta a las siguientes preguntas según su conocimiento
Adquirido:

1. Desde la visión del programador que es un Framework?
2. Defina que es un Framework?
3. Un patrón se puede aplicar en una sola aplicación si o no y porque?

EJERCICIOS DE APRENDIZAJE

Nombre del ejercicio de aprendizaje: J2EE API	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: La interfaz de programación de aplicaciones (API), es un conjunto de funciones o métodos que ofrecen las bibliotecas de los lenguajes de programación para ser utilizados por el usuario desarrollador. Enumere una clase, un tipo primitivo, asignación de un valor interno y la impresión del resultado.	
Solución del taller: Un ejemplo de este tipo de casos: Import java.lang.*; Este tipo de paquetes (lang) se carga por defecto cuando creamos un aplicativo en java, mediante él podemos utilizar: Int sw = 0; sw = 1; System.out.print ("\\nvalor del switch " + sw); Todo lo que hagamos dentro de un aplicativo o dentro de una clase que maneje los procesos básicos están utilizando lang, esto es parte del API, función clave si queremos conocer cada aspecto de los comandos y funciones de la herramienta de programación.	

Pista de aprendizaje:

Tener en cuenta: el API, es un componente de todos los lenguajes de programación.

Tenga presente: que debemos de conocer y manejar el API con cierto detalle para aprovechar los recursos que nos ofrece.

Traer a la memoria: que esto está en la documentación del lenguaje, no hay que comprar un libro para cada tema que sale.

2.4. Arquitectura de tres capas

Estilos de programación que en esencia requieren separar las funcionalidades y lógicas más comunes dependiendo de los objetivos.

Como definir el dominio, los requerimientos (normas del negocio), la trazabilidad, los informes, formularios entre otros además permite la escalabilidad, la integración, la disponibilidad y la seguridad de los datos.

Definición TRES CAPAS

Actividad estudio de caso para tres capas

Para esta ocasión se realizará un ejemplo de Mantenimiento de Datos desde Java utilizando la Base de Datos Ventas.

Con respecto a la base de datos llamada BD Venta Artículo crearemos sus respectivas tablas y campos pero utilizaremos solo la Tabla Artículos para realizar los diversos mantenimientos desde el formulario en Java.

En primer lugar creamos un Proyecto de Aplicación Java (VentasN-Capas), después de ello se tiene que agregar el manejador de base de datos (Access o mysql Jdbc a la Librería del Proyecto (Para conectarnos con la base de datos llamada BD Venta Artículo), luego creamos los 3 paquetes (Capas) con sus respectivas Clases, descritas a continuación:

El Paquete Idat.DAO = Representa a la Capa Datos

El Paquete Idat.BEANS = Representa a la Capa Negocio

El Paquete Idat.Forms = Representa a la Capa Presentación

Dentro de la Capa Idat.DAO en la clase Modulo.java se realiza la conexión a BD Venta Artículo mediante 2 funciones.

Ejemplo Tomado de (Retamozo, 2005)

```
package Idat.DAO;
import java.sql.*;
public class Modulo {
String driver="com.mysql.jdbc.Driver";
String url="jdbc:mysql://localhost:3306/ventas";
String user="root";
```



```
String pwd="";
```

```
public ResultSet Listar(String Cad){  
try{  
Class.forName(driver).newInstance();  
Connection cn=DriverManager.getConnection(url,user,pwd);  
PreparedStatement da = cn.prepareStatement(Cad);  
ResultSet tbl = da.executeQuery();  
return tbl;  
}catch(Exception e){  
javax.swing.JOptionPane.showMessageDialog(null,e.getMessage());  
return null;  
}  
}  
  
public String Ejecutar(String Cad){  
try{  
Class.forName(driver).newInstance();  
Connection cn=DriverManager.getConnection(url,user,pwd);  
PreparedStatement da = cn.prepareStatement(Cad);  
int r=da.executeUpdate();  
return "Se afectaron " + r + " filas";  
}catch(Exception e){  
javax.swing.JOptionPane.showMessageDialog(null,e.getMessage());  
return "Error "+e.getMessage();  
}  
}  
}
```

Dentro de la Capa Idat.BEANS en la clase Articulos.java se realizarán las funciones de Grabar, Editar, Eliminar y Listar Artículos y además de ello se declararán las variables y se encapsularán.

```
package Idat.BEANS;  
import java.util.ArrayList;  
import java.sql.*;  
import Idat.DAO.Modulo;  
public class Articulos {  
//Variables a encapsular  
private String art_cod;  
private String art_nom;  
private String art_uni;
```

```
private double art_pre;  
private int art_stk;  
private String art_marca;  
public String EliminarArticulo(){  
Modulo objmod=new Modulo();  
String cad="delete from articulos where  
art_cod='"+this.getArt_cod()+"';"  
return objmod.Ejecutar(cad);  
}  
public String GrabarArticulo(){  
Modulo objmod=new Modulo();  
String cad="insert into articulos  
values('"+this.getArt_cod()+"','"+this.getArt_nom()+  
"','"+this.getArt_uni()+"','"+this.getArt_pre()+  
"','"+this.getArt_stk()+"','"+this.getArt_marca()+"')";  
return objmod.Ejecutar(cad);  
}
```

La evaluación de Arquitectura de tres capas (2 valor 7.7 %)

Describir conceptos obtenidos en Arquitectura de tres Capas.

Se validan los conceptos de la funcionalidad lógica y los objetivos trasados.

Responda correctamente las siguientes preguntas según su conocimiento:

1. Desde la visión del programador es posible la separación de las funcionalidades de la lógica si o no y porque?
2. Defina como establecería la capa de persistencia?
3. Donde estableceria las actividades concernientes al dominio y a los requerimientos y porque?

EJERCICIO DE APRENDIZAJE

Nombre del ejercicio de aprendizaje: Arquitectura	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Existen muchas y muy variadas arquitecturas aplicadas al desarrollo de software, estas arquitecturas buscan unificar o estandarizar los formatos de programación, organizar y dar mayor versatilidad a la hora de encontrar un recurso o un proceso, cada archivo tiene ciertas características que permiten su fácil ubicación y modificación. Enumere las capas más comunes y una breve descripción.	
Solución del taller: Tenga presente Capa lógica Es el lugar donde se crean los archivos que contenga los procesos, las funciones, las tareas o métodos que se desean aplicar, es el equivalente al motor de un vehículo. Capa de presentación Comprende todo lo que el usuario visualizará durante el manejo del programa, si lo compramos con el vehículo, sería lo que el conductor puede ver de él, toda la estructura que queremos mostrarle a los demás Capa de datos Es el lugar donde reposaran los datos que deseamos procesar.	

Pista de aprendizaje:

Tener en cuenta: son metodologías que ayudan a unas buenas prácticas de desarrollo.

Tenga en cuenta: existen muchas metodologías, no se las aprenda todas, conozca una o dos de ellas para su uso personal.

Traiga a la memoria: que esta herramienta es la mejor forma de optimizar espacio y recursos, además de ser muy ordenado para el programador.

2.5. Modelo vista control MVC

Aplicar componentes tecnológicas para realizar la arquitectura de MVC. Llevarlo a la combinación de capas.

Conectividad

Aplicación de MVC

Definición mvc

Es un patrón de diseño en el cual se describe la estructura de una aplicación, para ello emplea la arquitectura de capas que van a componer la misma y la funcionalidad de cada una.

El modelo incluye la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación.

En una aplicación J2EE el modelo puede ser implementado mediante clases estándar de Java.

La vista son los resultados generados dinámicamente y proporcionados para dar respuesta al cliente, para ello se debe emplear cualquiera de los servicios web, entre otros los servlets o los jsp.

El controlador administra la aplicación, distribuyendo los requerimientos a las respectivas capas tomado de (Rincon, 2005)

Se presentan ventajas como:

Desarrollo sencillo y limpio.

Desarrollo de agil mantenimiento y escalable

Gran posibilidad para detección de errores.

Trabajo colaborativos

Se realizará el siguiente taller donde se elabora un proyecto llamado modelovistacontrol donde se adicionan las siguientes aplicaciones en su orden respectivo clases tomada de (Rincon, 2005).

Debe crear un proyecto el cual tendrá como principal CalcController posteriormente adicione cada una de las siguientes clases. Debe compilarlas y para su funcionamiento debe ejecutar se desde CalcController:

```
//Primero en ejecutar
```

```
import java.awt.event.*;

public class CalcController {
    private CalcModel m_model;
    private CalcView m_view;

    CalcController(CalcModel model, CalcView view) {
        m_model = model;
        m_view = view;
        view.addMultiplyListener(new MultiplyListener());
        view.addClearListener(new ClearListener());
    }

    class MultiplyListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String userInput = "";
            try {
                userInput = m_view.getUserInput();
                m_model.multiplyBy(userInput);
                m_view.setTotal(m_model.getValue());
            } catch (NumberFormatException nfex) {
                m_view.showError("Bad input: " + userInput + "");
            }
        }
    }
}

//end inner class MultiplyListener

    class ClearListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            m_model.reset();
            m_view.reset();
        }
    }
}

// end inner class ClearListener
}
```

```
// Segundo Clase
```

```
import java.math.BigInteger;
```

```
public class CalcModel {
    private static final String INITIAL_VALUE = "0";

    private BigInteger m_total;

    CalcModel() {
        reset();
    }

    public void reset() {
        m_total = new BigInteger(INITIAL_VALUE);
    }

    public void multiplyBy(String operand) {
        m_total = m_total.multiply(new BigInteger(operand));
    }

    public void setValue(String value) {
        m_total = new BigInteger(value);
    }

    public String getValue() {
        return m_total.toString();
    }
}

//Tercera clase

import javax.swing.*;

public class CalcMVC {
    public static void main(String[] args) {

        CalcModel    model    = new CalcModel();
        CalcView     view     = new CalcView(model);
        CalcController controller = new CalcController(model, view);

        view.setVisible(true);
    }
}
```

// La Cuarta

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class CalcView extends JFrame {
    private static final String INITIAL_VALUE = "1";

    private JTextField m_userInputTf = new JTextField(5);
    private JTextField m_totalTf = new JTextField(20);
    private JButton m_multiplyBtn = new JButton("Multiply");
    private JButton m_clearBtn = new JButton("Clear");

    private CalcModel m_model;

    CalcView(CalcModel model) {
        m_model = model;
        m_model.setValue(INITIAL_VALUE);

        m_totalTf.setText(m_model.getValue());
        m_totalTf.setEditable(false);

        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());
        content.add(new JLabel("Input"));
        content.add(m_userInputTf);
        content.add(m_multiplyBtn);
        content.add(new JLabel("Total"));
        content.add(m_totalTf);
        content.add(m_clearBtn);

        this.setContentPane(content);
        this.pack();

        this.setTitle("Simple Calc - MVC");
        // The window closing event should probably be passed to the
        // Controller in a real program, but this is a short example.
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
void reset() {  
    m_totalTf.setText(INITIAL_VALUE);  
}  
  
String getUserInput() {  
    return m_userInputTf.getText();  
}  
  
void setTotal(String newTotal) {  
    m_totalTf.setText(newTotal);  
}  
  
void showError(String errMessage) {  
    JOptionPane.showMessageDialog(this, errMessage);  
}  
  
void addMultiplyListener(ActionListener mal) {  
    m_multiplyBtn.addActionListener(mal);  
}  
  
void addClearListener(ActionListener cal) {  
    m_clearBtn.addActionListener(cal);  
}  
}
```

EJERCICIO DE AUTOEVALUACIÓN

LA EVALUACIÓN DE MVC (3 VALOR 7.7 %)

Describir conceptos obtenidos en MVC.

Se validan los conceptos de MVC.

Describe cada una de las siguientes preguntas según su conocimiento adquirido:

1. Desde la visión del MVC cuales serán las ventajas al realizar una aplicación de esta manera?
2. Defina en que consiste un modelo de dos capas?

3. Un modelo involucra la persistencia de los datos si o no y porque?

Nombre del ejercicio de aprendizaje: Modelo Vista Control	Datos del autor del taller: César Augusto Jaramillo Henao
<p>Escriba o plantee el caso, problema o pregunta:</p> <p>La programación por capas tiene una que es muy vistosa y famosa, tal es el caso de MVC, también conocida como MODELO VISTA CONTROL o CONTROLADOR. En esas 3 capas (por lo menos) se especifica la información clasificada, con el fin de que los procesos sean más fáciles de encontrar y acceder.</p> <p>Clasifique las capas con sus nombres más comunes.</p> <p>Su clasificación es la siguiente:</p> <p>MODELO: es la encargada de procesar los datos de uso común (información)</p> <p>VISTA: se refiere a todo lo que el usuario puede visualizar, formulario de ingreso, reportes</p> <p>CONTROL o CONTROLADOR: es la encargada de la lógica del negocia, los procesos central y decisivos</p>	

Pista de aprendizaje:

Tener en cuenta: una metodología muy utilizada, tiene muchas más alternativas y alcances.

Tenga presente: es por el bienestar del programador y una técnica organizada.

Traer a la memoria: que por lo menos son 3 capas, se pueden utilizar muchas más según la necesidad del desarrollador.

Taller de entrenamiento:

Nombre del taller: Programación por Capas	Datos del autor del taller: César Augusto Jaramillo Henao
Actividad previa: lenguaje de Programación III – Unidad 2.5 Arquitectura de tres Capas.	
Describe la actividad: Cree un esquema de desarrollo para un sitio web, clasifique los directorios según su uso. Los formularios al que el usuario accede van en la carpeta vista, los procesos lógicos y de toma de decisiones van en la carpeta control y la información va en la carpeta modelo.	

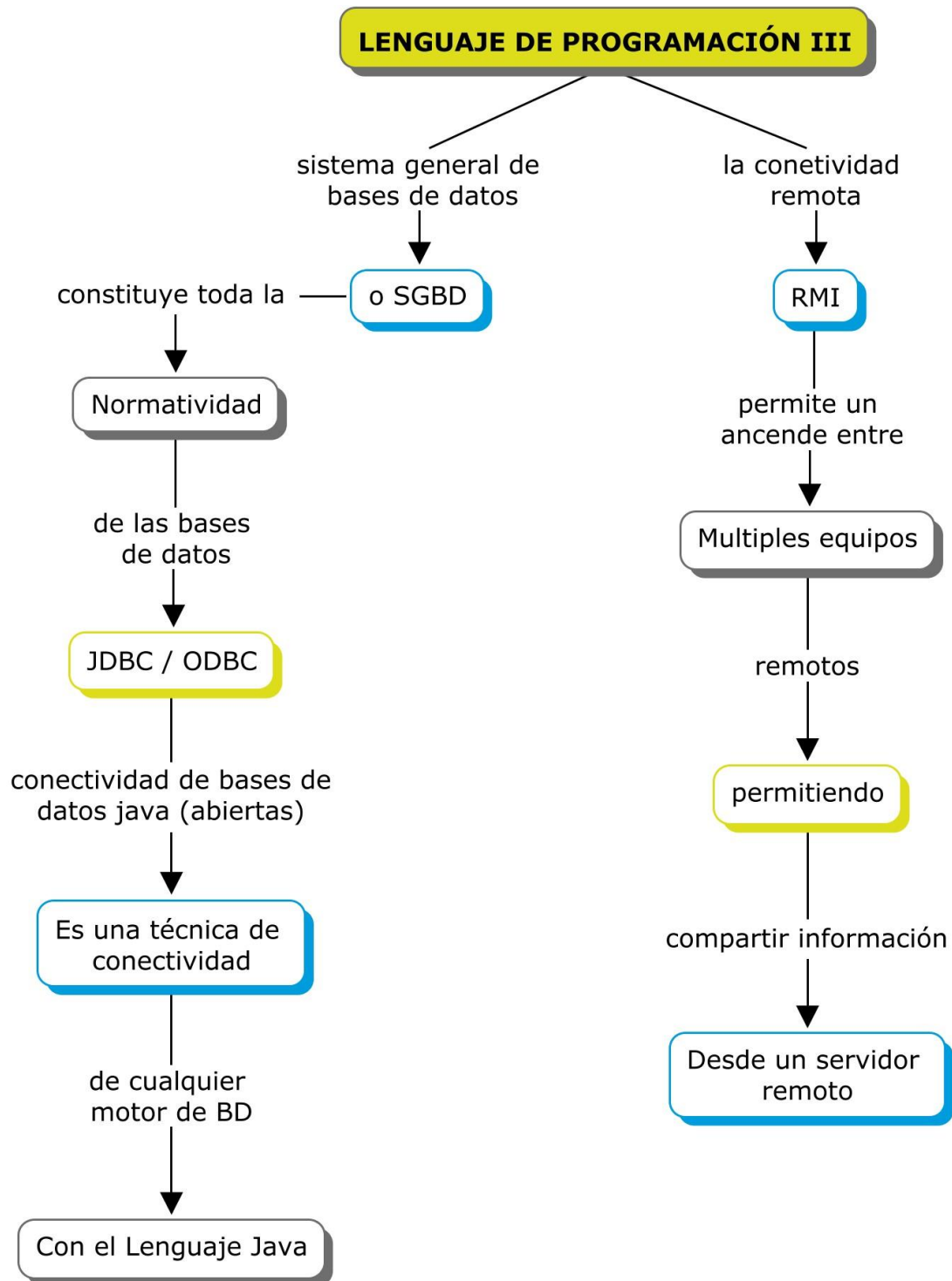
3. BASES DE DATOS CON JDBC

<http://www.youtube.com/watch?v=RzntNv1vbDE>



Imagen relacionada del video de youtube

3.1. Relacion de conceptos



OBJETIVO GENERAL

Implementar la persistencia de los Datos en la tecnología JDBC en base a capas de forma segura.

OBJETIVOS ESPECÍFICOS

- ❏ Diferenciar los conceptos en el despliegue y la persistencia con la Web
- ❏ Gestionar el sistema de base de datos y su servicio Web.
- ❏ Desarrollar las estrategias del ciclo de vida del software bajo la plataforma de soporte del lado del cliente.

3.2. Prueba Inicial

Explorar los conceptos para el funcionamiento de las estructuras de datos estáticas y dinámicas sobre almacenamiento permanente y la implementación en la interfaz gráfica de java (Booch, 1996). Implementar los conocimientos alcanzados en lógica de programación

Se validan preconceptos de bases de datos en java, las estructuras y métodos en SQL, para la estructuración del JDBC.

1. A cual tipo de almacenamiento corresponden las bases de datos
 - a) Temporal momentáneo estática
 - b) Permanente sin estructura
 - c) Permanente con estructura
 - d) Temporal dinámico
 - e) Permanente con estructura y relación
2. JDBC es una API de bajo nivel para API que permite la comunicación(conexión) con:
 - a) El desarrollador de software
 - b) El operador de eventos.
 - c) El manejador de base de datos.
 - d) El archivo de datos Secuencial
 - e) El manejador de Servidor
3. La función del JDBC:

- a) Es realizar conexión con una Bd y aplicar métodos de SQL, proporcionar resultados.
 - b) Es realizar conexión con un archivo y aplicar métodos de SQL, proporcionar resultados
 - c) Es realizar conexión con una lista y aplicar métodos de SQL, proporcionar resultados
 - d) Es realizar conexión con una Bd y aplicar métodos de listas, proporcionar resultados
 - e) Es realizar aplicación con almacenamiento temporal y aplicar métodos de SQL, proporcionar resultados
4. Una base de datos:
- a) Almacena información de diferente tipo de manera estructurada, permanente y segura.
 - b) Almacena información del mismo tipo de manera estructurada, temporal y segura.
 - c) Almacena información de manera estructurada, temporal y segura.
 - d) Almacena información de diferente tipo de manera secuencial y segura.
 - e) Almacena información de manera estructurada, permanente y sin seguridad.
5. Que entiende por SQL :
- a) Lenguaje que permite efectuar consultas para recuperar y modificar datos, almacenarlos de manera estructurada permanentemente, manejando el algebra relacional.
 - b) Lenguaje que no permite efectuar consultas para recuperar y modificar datos, almacenarlos de manera estructurada permanentemente, manejando el algebra relacional.
 - c) Lenguaje que permite efectuar consultas para recuperar y modificar datos, no permite almacenarlos de manera estructurada permanentemente, manejando el algebra relacional.
 - d) Lenguaje que permite efectuar consultas para recuperar y modificar datos almacenados, almacenarlos de manera estructurada permanentemente, sin manejar el algebra relacional.
 - e) Lenguaje que permite efectuar consultas para recuperar y modificar datos almacenados, almacenarlos de manera dispersa manejando el algebra relacional.
6. La función del sistema manejador de base de datos :
- a) Es realizar conexión con una matriz, aplicar métodos de SQL, proporcionar resultados, Gestionar el almacenamiento de los datos, ahorrar costos de inversión en el manejo de los datos, seguridad y disponibilidad de acceso a los datos

- b) Es realizar conexión con un archivo, aplicar métodos de SQL proporcionar resultados Gestionar el almacenamiento de los datos, ahorro en costos de inversión en el manejo de los datos, seguridad y disponibilidad de acceso a los datos .
- c) Es realizar conexión con una variable, aplicar métodos de SQL proporcionar resultados ahorro en costos de inversión en el manejo de los datos, seguridad y disponibilidad de acceso a los datos.
- d) Es realizar conexión con una Base de datos aplicar métodos de SQL proporcionar resultados ahorro en costos de inversión en el manejo de los datos, seguridad y disponibilidad de acceso a los datos
- e) Es realizar conexión con un vector, aplicar métodos de SQL proporcionar resultados ahorro en costos de inversión en el manejo de los datos, seguridad y disponibilidad de acceso a los datos.

3.3. Bases de datos con JDBC

Es el API estándar ofrecido por Java para gestionar bases de datos (Hernández, Adela, & Bobadilla Sancho, 2005). Ilustra la arquitectura de los diferentes tipos de drivers JDBC ((Java Database Connectivity) sus correspondientes clases y cada una de las clases e interfaces que componen el API JDBC

Definición JDBC

Es la propiedad de un objeto por la que su existencia trasciende el tiempo es decir, el objeto continúa existiendo después de que su creador deja de existir; el espacio es decir la posición del objeto varía con respecto al espacio de direcciones en el que fue creado.

La persistencia abarca algo mas que la mera duración de los datos en las bases de datos orientada a objetos, no solo persiste el estado de un objeto; si no que su clase debe trascender a cualquier programa individual, de forma que todos los programas interpreten de la misma manera el estado almacenamiento

Es una biblioteca de clases que permite la conexión con bases de datos utilizando java, las operaciones que se realizan se agrupan en (consultas, actualizaciones) se apoya en el SQL (structure query language). Requiere de el paquete sql para obtener las clases básicas de Conexion, de Instrucción y de resultados donde se incluyen respectivamente los métodos para las actualizaciones como `executeUpdate ()` o de consulta como el `executeQuery ()`, combinados con

el sql, básicamente esta en marcado en la estructura de dos capas pero podría variar a tres dependiendo de los requerimientos

La conectividad de la base de datos de Java (JDBC, Java Database Connectivity) es un marco de programación para los desarrolladores de Java que escriben los programas que tienen acceso a la información guardada en bases de datos específicamente. JDBC se utiliza comúnmente para conectar un programa del usuario con una base de datos por “detrás de la escena”, sin importar qué software de administración o manejo de base de datos se utilice para controlarlo. De esta manera, JDBC es una plataforma-cruzada.

En que consiste JDBC?

JDBC es un API de Java para acceder a manejadores de bases de datos, y prácticamente a cualquier tipo de dato tabular. El API JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea

Nuestra aplicación Java debe tener acceso a un controlador (driver) JDBC adecuado. Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real. De una manera muy simple.

Los drivers que maneja Access, SQL, Oracle:

Para realizar una conexión con alguna base de datos debemos de importar el paquete: JAVA.SQL. Este paquete contiene clases e interfaces diseñadas teniendo en mente la arquitectura tradicional cliente-servidor. Su funcionalidad se centra primordialmente en servicios de programación básicos de bases de datos, como creación de conexiones, ejecución de Instrucciones e instrucciones preparadas.

Como se identifica la clase de conexión?

Para este caso debo saber primero como se llama, Cuando se intenta conectar Java, con cualquier motor de base de datos necesitamos conocer si la relación de Conexión de Java con el motor de base de datos es de tipo directa o indirecta.

Antes de esto definamos que es un controlador DBC(Java Database Connectivity), el cual es un interfase de comunicación que permite la ejecución entre Java y cualquier motor de Base de Datos.

Entre los manejadores de base de datos que necesitan de un puente ODBC para conectarse con Java, tenemos a las marcas, Access, Microsoft SQL Server, Informix, entre otros.

Cuales son las clases para realizar la conexión?

Connection: Es el que realiza la conexión a la base de datos

Statement : Para las instrucciones u ordenes

PreparedStatement: Es el que prepara y contiene la sentencia SQL

ResultSet: Es la que contiene el resultado.

Importante resaltar los controladores de excepciones para el flujo de los datos try/Catch().

El ODBC es el puente o conexión para este debemos establecer o tener el nombre del origen de datos, también es una interface que proporciona un acceso a base de datos; en SQL se ha proporcionando una interface consistente para comunicarse no solo con los datos si no también para acceder a la database una metadata información sobre las tablas que componen la base de datos, sus campos y características que lo definen.

En el código del servlet se puede observar los atributos de la clase visto desde un applet:

```
public class Conexion extends HttpServlet {  
    private Connection con = null;    // Conexion  
    private boolean driver = false;    // true si se ha cargado driver en init()  
    private Propiedades acceso;    // Clase para saber driver, host
```

El primer atributo es una referencia a la conexión.

El segundo es una bandera (flag) que nos sirve para indicar que se ha producido un error al cargar el driver en init (). En las aplicaciones con web o en el main en las básicas

El tercero es una referencia a un objeto de la clase Propiedades. Veremos lo que hace esta clase.

Pero antes veremos lo que hace el método init () de nuestro servlet: para nuestro caso con Tomcat el cual veremos en el tema correspondiente próximamente.

Es importante recordar que init () sólo se ejecuta en la primera invocación al servlet. Por tanto cualquier modificación al archivo .properties surtirá efecto una vez que recarguemos el servlet (el manager de Tomcat viene con una opción de 'reload').

En doPost () respondemos a cada invocación al servlet desde el formulario HTML.

La tarea más importante que se realiza aquí es la apertura de una conexión a la base de datos.

¿Dónde poner la conexión a la base de datos? ¿En init () o en doXX()? Dicho de otra forma, ¿nos conectamos en init () o cada vez que se hace una solicitud (un "request" o invocación al servlet)? Lo normal es que queramos que cada usuario abra una conexión a la base de datos, por tanto

intentaremos la conexión en respuesta a una solicitud get o post. Si pusieramos la conexión en el init (), dicha conexión sólo se realizaría la primera vez que se invocase el servlet.

En doPost() realizamos las siguientes tareas:

Empezamos con lo habitual, definiendo el tipo de salida: `response.setContentType ("text/html; charset=iso-8859-1");` // Definir tipo de salida

`PrintWriter out = response.getWriter ();` // Obtener flujo salida

A continuación se imprime el inicio de página:

```
try {      //// Imprimir inicio página
    out.println("");
```

Comprobamos que el driver se ha cargado correctamente, avisando al usuario. Si la carga no fue correcta terminamos la página (escribimos etiquetas de fin) y salimos (return) del método:

```
if ( !driver ) {
    out.println("<br>No se ha cargado el driver");
    terminarPagina( out );
    return;
}
```

Realizamos la conexión a la base de datos por medio del mensaje `DriverManager.getConnection ()` que lanza una excepción del tipo `SQLException` si hay un error.

Aquí es interesante observar que los argumentos de este mensaje se obtienen de los parámetros del formulario HTML, por medio de `request.getParameter ("nombre_del_parámetro");` salvo el host, que se obtiene del objeto de la clase

En nuestro programa java, todos los import que necesitamos para manejar la base de datos están en `java.sql.*`. Puesto que casi todas los métodos relativos a base de datos pueden lanzar la excepción `SQLException`, meteremos todo nuestro programa en un try-catch.

Para el caso de no trabajar con Access debemos tener el servidor de MySQL (Tomcat) arrancado. Si hemos instalado y dejado esa opción como estaba, cada vez que encendamos el ordenador, se arrancará el servidor de MySQL, por lo que no tenemos que preocuparnos por ello.

El servidor de MySQL abre por defecto el puerto 3306 para aceptar conexiones de posibles clientes, de programas que quieran conectarse y acceder a la base de datos. Nuestro programa java, si quiere consultar la tabla de base de datos que hemos creado, deberá conectarse a este servidor.

Para establecer la conexión, la clase DriverManager tiene métodos getConnection(). Usaremos uno de ellos

```
// Establecemos la conexión con la base de datos.
```

```
Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/prueba","root",  
"la_clave");
```

El primer parámetro del método getConnection () es un String que contiene la url de la base de datos:

Jdbc:mysql porque estamos utilizando un driver jdbc para MySQL, que es el que nos hemos bajado.

Localhost porque el servidor de base de datos, en mi caso, está en el mismo ordenador en el que voy a correr el programa java. Aquí puede ponerse una IP o un nombre de máquina que esté en la red.

Prueba es el nombre de la base de datos que he creado dentro de mysql. Se debe poner la base de datos dentro del servidor de MySQL a la que se quiere uno conectar. Es el nombre que pusimos cuando desde SQL hicimos create database prueba;

Los otros dos parámetros son dos String. Corresponden al nombre de usuario y password para acceder a la base de datos. Al instalar MySQL se crea el usuario root y se pide la password para él. Como no hemos creado otros usuarios, usaremos este mismo.

Para realizar cualquier acción sobre la base de datos (consulta, insertar nuevos registros, modificar los existentes o borrar), necesitamos una clase Statement. Para obtenerla, se le pide dicha clase a la conexión. La forma de hacerlo, para una consulta, es la siguiente:

La parte de createStatement () no tiene ningún secreto, salvo que puede lanzar una excepción que hay que capturar.

El Statement obtenido tiene un método executeQuery (). Este método sirve para realizar una consulta a base de datos

En realidad, ResultSet no contiene todos los datos, sino que los va consiguiendo de la base de datos según se van pidiendo. Por ello, el método executeQuery () puede tardar poco, pero el recorrer los elementos del ResultSet no es tan rápido. De esta forma se evita que una consulta que dé muchos resultados tarde mucho tiempo y llene la memoria del programa java.

El ResultSet contiene dentro los registros leídos de la base de datos. Inicialmente, tal cual nos lo devuelve el Statement.executeQuery (), tiene internamente un "puntero" apuntando justo delante del primer registro. El método next() del ResultSet hace que dicho puntero avance al siguiente registro, en este caso, al primero. Si lo consigue, el método next() devuelve true. Si no lo consigue (no hay siguiente registro que leer), devuelve false.

Por tanto, una forma de ir leyendo los registros en meternos en un while.

// Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en pantalla.

```
while (rs.next())
```

```
{
```

```
    System.out.println (rs.getInt (1) + " " + rs.getString (2)+ " " + rs.getDate(3));
```

```
}
```

// Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en pantalla.

```
while (rs.next())
```

```
{
```

```
    System.out.println (rs.getInt (1) + " " + rs.getString (2)+ " " + rs.getDate(3));
```

```
}
```

Una vez que el "puntero" está apuntando a un registro, los métodos getInt(), getString(), getDate(), etc nos van devolviendo los valores de los campos de dicho registro. Podemos pasar a estos métodos un índice (que comienza en 1) para indicar qué columna de la tabla de base de datos deseamos. También podemos usar un String con el nombre de la columna (tal cual está en la tabla de base de datos).

Es responsabilidad nuestro saber qué tipo de dato hay en cada columna, aunque si nos equivocamos y RecordSet es capaz de hacer la conversión, la hará por nosotros. Por ejemplo, en cualquiera de los campos anteriores podemos pedir un getString() y nos devolverán los números como String y la fecha como String.

También podemos usar getObject(), y el RecordSet nos devolverá el Object más adecuado para el tipo de campo que pedim

Lo primero que necesitamos para conectarnos con una base de datos es un Driver (o Connector) con ella. Ese Driver es la clase que, de alguna forma, sabe cómo hablar con la base de datos. Desgraciadamente (y hasta cierto punto es lógico), java no viene con todos los Drivers de todas las posibles bases de datos del mercado. Debemos ir a internet y obtener el Driver, normalmente en la página de nuestra base de datos

DriverManager tiene muchos métodos `getConnection()` con parámetros variados. Todos son variantes de lo mismo y la información que suministramos es la misma. Aquí hemos utilizado uno con tres parámetros String, que vamos a explicar.

url: Es una cadena que nos permite localizar la base de datos. Para mysql, el formato es

"jdbc:mysql://ordenador_donde_corre_la_base_de_datos/nombre_base_datos". Donde se pone el nombre o IP del ordenador en el que se encuentra nuestro servidor de base de datos y el nombre de la base de datos. En nuestro ejemplo, tenemos el servidor de base de datos corriendo en el mismo ordenador que el programa java, por lo que ponemos localhost. La base de datos la he llamado agenda. El comando SQL para crear la base de datos agenda sería
mysql> CREATE DATABASE agenda;

user: Un usuario válido para la base de datos.

password: La clave del usuario.

ACTIVIDAD ESTUDIO DE CASO

```
import java.sql.*;
class Select {
    public static void main (String argv[ ]) {
        try {
            // Busca el driver para el tipo de DBMS en cuestion
            Class.forName("com.mysql.jdbc.Driver");
            //definiendo url de conexion con la base de datos
            String url="jdbc:mysql://localhost/agenda","root", "clave ";

            //Abriendo conexion
            Connection con = DriverManager.getConnection(url, "root ","Clave");

            // crear enunciado
            Statement stmt = con.createStatement();
            //ejecutar query
            ResultSet rs = stmt.executeQuery ("select * from users;");
            // Ciclo para recorrer los resultados
            System.out.println("Got results:");
            while (rs.next( )) {

                int ID= rs.getInt(1);
                String Name= rs.getString(2);
```

```
        System.out.print(ID + " " + Name );
        System.out.print("\n");
    }
    //cerrando el statement
    stmt.close( );
    //cerrando conexion
    con.close( );
}
catch( Exception e) {
    e.printStackTrace( );
}
}
}
```

una aplicacion en JDBC usando PreparedStatement

```
import java.sql.*;
class Select {
    public static void main (String argv[ ]) {
        try {
            // Busca el driver para el tipo de DBMS en cuestion
            Class.forName("com.mysql.jdbc.Driver");
            //definiendo url de conexion con la base de datos
            String url="jdbc:mysql://localhost/agenda","root", "la_clave ";

            //Abriendo conexion
            Connection con = DriverManager.getConnection(url, "root","Clave");

            //ejecutar query
            // Teniendo la tabla datafiles(id int, description varchar, category int, files blob)
            // blob ~ byte
            String sql = "insert into datafiles values(?,?,?,?);";

            // crear enunciado
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, 34);
            ps.setString(2,"rock");
            ps.setString(3,"music");
            FileInputStream inputStream = null;
```

```
int fileLength = Integer.MIN_VALUE;
try
{
    File file=new File("/develop/users/root/archivo.txt");
    inputStream = new FileInputStream(file);
    fileLength= (int) file.length();
    ps.setBinaryStream(4, inputStream, fileLength);
}
catch(FileNotFoundException fnfe)
{ fnfe.printStackTrace();}

ps.executeUpdate();
//-----
//recuperamos esa informacion
// crear enunciado
Statement stmt = con.createStatement();
//ejecutar query
ResultSet rs = stmt.executeQuery ("select * from datafiles;");
// Ciclo para recorrer los resultados
System.out.println("Got results:");
while (rs.next() ) {
    InputStream inputStream= rs.getBinaryStream("files");
    ByteArrayOutputStream baos=new ByteArrayOutputStream();
    int abyte;
    while(true)
    {
        try
        {
            abyte=inputStream.read();
            if (abyte== -1)
                break;
            baos.write(abyte);
        }
        catch(Exception e1)
        { break; }
    }
    //while
    System.out.println(baos.toByteArray());
}
//cerrando el statement
```

```
stmt.close( );  
//cerrando conexion  
con.close( );  
}  
catch( Exception e) {  
    e.printStackTrace( );  
}  
}  
}
```

A continuación se proporciona una aplicación ejemplo básico de MySQL con Java

Una vez instalado MySQL, descargado el driver para java de MySQL y con una base de datos y una tabla creada en MySQL, vamos a hacer un pequeño programa en java que nos permita conectarnos a la base de datos MySQL y consultar la tabla que hemos creado.

Instalar el Driver

En nuestro programa java, todos los import que necesitamos para manejar la base de datos están en `java.sql.*`. Puesto que casi todos los métodos relativos a base de datos pueden lanzar la excepción `SQLException`, meteremos todo nuestro programa en un try-catch.

Además, necesitamos la clase `org.gjt.mm.mysql.Driver` que viene con el driver de MySQL. Por ello, en nuestro CLASSPATH o incluido en nuestro proyecto con nuestro IDE favorito, debemos incluir el jar que contiene el driver MySQL (`mysql-connector-java-3.1.7-bin.jar`) o la versión más moderna y compatible con la versión de nuestro servidor de MySQL.

Lo primero que tenemos que hacer es asegurarnos que el Driver se inicializa y se registra, para ello

```
try  
{  
    Class.forName("com.mysql.jdbc.Driver");  
} catch (Exception e)  
{  
    e.printStackTrace();  
}
```

Establecer la conexión con la base de datos

Debemos tener el servidor de MySQL arrancado. Si hemos instalado y dejado esa opción como estaba, cada vez que encendamos el ordenador, se arrancará el servidor de MySQL, por lo que no tenemos que preocuparnos por ello.

El servidor de MySQL abre por defecto el puerto 3306 para aceptar conexiones de posibles clientes, de programas que quieran conectarse y acceder a la base de datos. Nuestro programa java, si quiere consultar la tabla de base de datos que hemos creado, deberá conectarse a este servidor.

Para establecer la conexion, la clase DriverManager tiene métodos getConnection(). Usaremos uno de ellos

```
// Establecemos la conexión con la base de datos.
```

```
Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/prueba","root",  
"la_clave");
```

El primer parámetro del método getConnection() es un String que contiene la url de la base de datos:

jdbc:mysql porque estamos utilizando un driver jdbc para MySQL, que es el que nos hemos bajado.

localhost porque el servidor de base de datos, en mi caso, está en el mismo ordenador en el que voy a correr el programa java. Aquí puede ponerse una IP o un nombre de máquina que esté en la red.

prueba es el nombre de la base de datos que he creado dentro de mysql. Se debe poner la base de datos dentro del servidor de MySQL a la que se quiere uno conectar. Es el nombre que pusimos cuando desde SQL hicimos create database prueba;

Los otros dos parámetros son dos String. Corresponden al nombre de usuario y password para acceder a la base de datos. Al instalar MySQL se crea el usuario root y se pide la password para él. Como no hemos creado otros usuarios, usaremos este mismo.

Si todo va bien, en conexion tendremos nuestra conexión a la base de datos.

Esta conexión es en realidad un socket entre java y la base de datos, aunque para nosotros es transparente. Lo que sí es importante, es saber que si varios hilos comparten esta conexión, deben usarla sincronizadamente. Si no se hace así, los mensajes que van por el socket se pueden entremezclar y los hilos pueden leer cachos de mensaje destinados al otro hilo. Otra opción es que cada hilo cree su propia conexión. Finalmente, la mejor opción de todas si nuestra aplicación va a tener varios hilos intentando acceder a la base de datos, es usar conexión

Para realizar cualquier acción sobre la base de datos (consulta, insertar nuevos registros, modificar los existentes o borrar), necesitamos una clase Statement. Para obtenerla, se le pide dicha clase a la conexión. La forma de hacerlo, para una consulta, es la siguiente:

```
// Preparamos la consulta
```

```
Statement s = conexion.createStatement();
```

```
ResultSet rs = s.executeQuery ("select * from persona");
```

La parte de `createStatement()` no tiene ningún secreto, salvo que puede lanzar una excepción que hay que capturar.

El `Statement` obtenido tiene un método `executeQuery()`. Este método sirve para realizar una consulta a base de datos.

El parámetro que se pasa en un `String` en el que está la consulta en lenguaje SQL. No hace falta terminarlo con punto y coma. En nuestro caso `"select * from persona"`. siendo `persona` el nombre que hemos puesto a la tabla en la base de datos.

El resultado nos lo devuelve el método como un `ResultSet`. Este `ResultSet` no es más que una clase java similar a una lista en la que está el resultado de la consulta. Cada elemento de la lista es uno de los registros de la base de datos. En realidad, `ResultSet` no contiene todos los datos, sino que los va consiguiendo de la base de datos según se van pidiendo. Por ello, el método `executeQuery()` puede tardar poco, pero el recorrer los elementos del `ResultSet` no es tan rápido. De esta forma se evita que una consulta que dé muchos resultados tarde mucho tiempo y llene la memoria del programa java.

El `ResultSet` contiene dentro los registros leídos de la base de datos. Inicialmente, tal cual nos lo devuelve el `Statement.executeQuery()`, tiene internamente un "puntero" apuntando justo delante del primer registro. El método `next()` del `ResultSet` hace que dicho puntero avance al siguiente registro, en este caso, al primero. Si lo consigue, el método `next()` devuelve `true`. Si no lo consigue (no hay siguiente registro que leer), devuelve `false`.

Por tanto, una forma de ir leyendo los registros en meternos en un `while`.

```
// Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en pantalla.
```

```
while (rs.next())
```

```
{
```

```
    System.out.println (rs.getInt (1) + " " + rs.getString (2)+ " " + rs.getDate(3));
```

```
}
```

Una vez que el "puntero" está apuntando a un registro, los métodos `getInt()`, `getString()`, `getDate()`, etc nos van devolviendo los valores de los campos de dicho registro. Podemos pasar a estos métodos un índice (que comienza en 1) para indicar qué columna de la tabla de base de datos deseamos. También podemos usar un `String` con el nombre de la columna (tal cual está en la tabla de base de datos).

Es responsabilidad nuestra saber qué tipo de dato hay en cada columna, aunque si nos equivocamos y RecordSet es capaz de hacer la conversión, la hará por nosotros. Por ejemplo, en cualquiera de los campos anteriores podemos pedir un `getString()` y nos devolverán los números como String y la fecha como String.

También podemos usar `getObject()`, y el RecordSet nos devolverá el Object más adecuado para el tipo de campo que pedimos.

Cerrar la conexión

Una vez que terminamos de usar la conexión, se debería cerrar, o bien terminar el programa, con lo que se cierra automáticamente.

```
// Cerramos la conexion a la base de datos.  
conexion.close();
```

Vamos ahora a ver ahora cómo meter los resultados de la consulta en un JTable.

Método de proyectos

Los servlets hacen en `init()` la carga del driver y en `doPos()/doGet` se conectan a la base de datos y ejecutan una consulta que se escribe sobre la página.

Practicamos el patrón DAO: los servlets no acceden directamente a la base de datos; sino que usan clases DAO que por medio de JDBC acceden a la base de datos. En un ejemplo anterior hemos tratado el patrón DAO. En este caso usamos como base el mismo código del ejemplo de patrón DAO, pero introduciendo ligeras diferencias (permitimos que cada conexión se haga con un login/password diferente).

Vamos a "anidar" servlets, dicho de otro modo: un servlet tiene como salida un formulario, que invoca a otro servlet. El primer servlet imprime un combobox ('desplegable') con todos los clientes, dando la opción al usuario de seleccionar uno. El segundo muestra las ventas del cliente seleccionado.

Además vamos a manejar la sesión. Es necesario que las llamadas a los servlets mantengan cierta continuidad, como ocurre en una web comercial, donde pasamos por las páginas del catálogo y se mantiene información del carrito de la compra o de cliente. Para mantener información a lo largo de sucesivas llamadas necesitamos usar atributos de sesión.

Hay que tener en cuenta la estructura del ejemplo. El directorio raíz es `docen_servlet01.JDBC01` (de la aplicación /public_html), del que parten los paquetes (directorios):

bean. Las clases (Cliente y Venta) que representan las entidades (tablas) de la base de datos. Todas ellas implementan el interface Bean.

accesoDatos. Las clase DAO (DAOCliente y DAOVenta) que consultan (select) la base de datos, implementan el interface InterfaceDAO y heredan de DAOGeneral (servicios comunes de carga de driver, conexión, desconexión, etc.)

Presentacion. Los servlets (FormClientes y FormVentas) con una clase de utilidad general (UtilGeneral).

Sólo los servlets (FormClientes y FormVentas) deben estar indicados en web.xml.

Los servlets en web.xml:

```
....
<servlet>
  <servlet-name>FormClientes</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormClientes</servlet-class>
</servlet>
<servlet>
  <servlet-name>FormVentas</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormVentas</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>FormClientes</servlet-name>
  <url-pattern>/servlet/FormClientes</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>FormVentas</servlet-name>
  <url-pattern>/servlet/FormVentas</url-pattern>
</servlet-mapping>
```

Puede obtener el código del ejemplo.

El ejemplo: los clientes y sus pedidos

En nuestra base de datos tenemos dos tablas: la primera representa información de clientes y la segunda contiene las ventas que se han realizado. La relación es de 1:N, ya que un cliente puede tener asignadas varias ventas. En el ejemplo:

El primer servlet realiza un formulario con una lista de clientes.

El segundo servlet recibe el cliente seleccionado y consulta en la base de datos las ventas de dicho cliente. El resultado es una página con las ventas del cliente.

Esquema global (nota: el servlet 'inicio' del dibujo se llama en realidad 'FormClientes' y el servlet 'ventas' se llama 'FormVentas'):

El servlet FormClientes

FormClientes empieza iniciando el DAO:

```
package docen_servlet01.JDBC01.presentacion;
import
public class FormClientes extends HttpServlet {

    private DAOCliente dc = null;          // Clase responsable de acceso a base de datos

    /*****
        Al inicializarse el servlet se crea el DAO: con este se carga el driver JDBC y
        se leen propiedades. El argumento del constructor del DAO es el dir raíz de la
        aplicación.
        Si ya se hubiesen cargado driver y propiedades, no se vuelven a cargar.
    *****/

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        dc = new DAOCliente( config.getServletContext().getRealPath("/") );
    }
    ....
}
```

A continuación veamos doPost(). Puede observarse en el código fuente que las salidas de código HTML se envían a la clase auxiliar UtilGeneral. Empezamos comprobando que el DAO ha cargado las propiedades y el driver de base de datos, las propiedades se definen en un archivo properties, que define el host, y dicho archivo se sitúa (desde el directorio raíz de la aplicación, public_html) en public_html/propiedades/docen_servlet01/parametros.properties. Pero esta lectura de archivo properties queda oculta (encapsulada) para el servlet, de ello se encarga la clase Propiedades, que es usada por el DAO:

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    PrintWriter out= response.getWriter();          // Obtener flujo salida;

    try {

        HttpSession sesion = request.getSession(true);
```

```
// Obtener sesión, si no existe, la crea response.setContentType("text/html; charset=iso-8859-1");
```

```
// Definir tipo de salida
```

```
UtilGeneral.imprimirInicioPagina( "Ejemplo de servlet", "Seleccione cliente", out);
```

```
//// Si no se han cargado propiedades, aviso y salgo
```

```
if ( dc.getPropiedades() == null ) {
```

```
UtilGeneral.imprimir( out, "No se han cargado las propiedades.");
```

```
return;
```

```
}
```

```
else
```

```
UtilGeneral.imprimir( out, "Propiedades cargadas: " +
```

```
dc.getPropiedades().getPathPropiedades()+dc.getPropiedades().getFicheroParametros());
```

```
//// Si no he podido cargar el driver JDBC, aviso y salgo
```

```
if ( !dc.estaCargadoDriver() ) {
```

```
UtilGeneral.imprimir( out, "No se ha cargado el driver " +
```

```
DAOGeneral.getPropiedades().getDriver(), true, true );
```

```
UtilGeneral.imprimirFinPagina( out );
```

```
return;
```

```
}
```

```
else
```

```
UtilGeneral.imprimir( out, "Driver cargado: " + dc.getPropiedades().getDriver());
```

En el archivo properties (/public_html/propiedades/docen_servlet01/parametros.properties) tenemos las siguientes líneas que definen el host y la base de datos:

```
basedatos.host=jdbc:mysql://localhost:3306/
```

```
docen_servlet01.JDBC01.basedatos=proactiv_prueba
```

Lo que sigue es definir los atributos de la sesión y mostrar el formulario que lista los clientes de la base de datos. La definición de atributos es sencilla. El login y password se obtienen de la petición (request) y la base de datos se obtiene de la clase Propiedades que usa DAOCliente. El formulario se escribe mediante una llamada al método del propio servlet imprimirFormulario

```
//// Poner atributos (base de datos, login y password en la sesion)
```

```
//// La base de datos se obtiene de un archivo properties y
```

```
//// el login y password de request sesion.setAttribute("basedatos",
```

```
dc.getPropiedades().getHostBaseDatos()+ dc.getPropiedades().getParametro
```

```
("docen_servlet01.JDBC01.basedatos"));
```

```
sesion.setAttribute("login", request.getParameter("login"));
sesion.setAttribute("password", request.getParameter("password"));

////////// IMPRIMIR FORMULARIO Y SESION
imprimirFormulario( request, out );
UtilGeneral.imprimirSesion(sesion, out);
}
catch (Exception e) {
    UtilGeneral.imprimir( out, "Error general. " + e.getMessage(), true, true );
e.printStackTrace();
}
finally {
    /// Cierre de página
    UtilGeneral.imprimirFinPagina( out );
}
}
```

imprimirFormulario() empieza con con dos llamadas a DAOCliente. Primero para definir el usuario (setIdenticacion()) a partir del login y password de la petición (request) HTTP y en segundo lugar obtener un vector de clientes (bean.Cliente) por medio de la llamada dc.select(null). EL argumento null indica que queremos listar todos los clientes, es decir, que no hay clausula WHERE en la sentencia SQL del DAO:

```
void imprimirFormulario( HttpServletRequest request, PrintWriter out ) {
try {
Vector vecClientes = null;

//// Almacenar en DAO la identificación (login-pwd), desde request
dc.setIdenticacion(request.getParameter("login"), request.getParameter("password"));

//// Usar el DAO para conseguir vector de clientes
try {
    vecClientes = dc.select(null);
}
catch ( Exception e) {
    UtilGeneral.imprimir(out, "Error en la consulta. " + e.getMessage());
}
}
```

A continuación imprimirFormulario() escribe en la salida el código HTML. Escribe los dos 'desplegables' que pueden verse en el formulario. Uno tiene los nombres de los clientes y otro tiene sus códigos.

///// Inicio de tabla HTML

```
out.println("<table BORDER=1 align=center cellpadding='10' cellspacing=1>");
```

```
out.println("<tr><td bgcolor=#00FF00>");
```

```
out.println("<FONT color=#000080 FACE='Arial,Helvetica,Times' SIZE=2>");
```

///// Inicio del formulario HTML

```
out.println("<form      action="+      dc.getProperties().getHostHTTP()+"servlet/FormVentas  
method='post'>");
```

```
UtilGeneral.imprimir( out, "Escoja cliente:");
```

///// Recorrer fila a fila el vector de clientes y poner en SELECT de NOMBRES

```
out.println("<SELECT      NAME='cliente'      onchange=\"copiarValor('nombre','codigo');\"  
id='nombre'>");
```

```
for ( int i = 0; i< vecClientes.size(); i++ ) {
```

```
Cliente c = (Cliente) vecClientes.get(i);
```

```
out.println("<OPTION VALUE=" + c.getCodigo()+">" + c.getApe1() + " " + c.getApe2() + " , " +  
c.getNombre()+""");
```

```
}
```

```
out.println("</SELECT>");
```

///// Recorrer fila a fila el vector de clientes y poner en SELECT de CODIGOS

```
out.println("<SELECT      NAME='cliente.codigo'      onchange=\"copiarValor('codigo','nombre');\"  
id='codigo'>");
```

```
for ( int i = 0; i< vecClientes.size(); i++ ) {
```

```
Cliente c = (Cliente) vecClientes.get(i);
```

```
out.println("<OPTION VALUE=" + c.getCodigo()+">" + c.getCodigo()+"</OPTION>");
```

```
}
```

```
out.println("</SELECT>");
```

//// Alternativa: campo oculto para el código de cliente (en vez de SELECT de codigos)

```
//out.println("<INPUT TYPE=HIDDEN NAME='cliente.codigo' id='codigo'>");
```

//// Poner botón y fin de formulario y de tabla

```
UtilGeneral.imprimir( out, "<input      type='submit'      name='Submit'  
value='Enviar'>");
```

```
out.println("</form></font></td></tr></table>");
```



```
    }  
    catch (Exception e) {  
        UtilGeneral.imprimir( out, "EROR EN FORMULARIO. " + e.getMessage(),  
true, true );  
    }  
}
```

Un aspecto a resaltar de los dos 'desplegables' es que están coordinados: si cambia en uno el nombre del cliente, entonces cambia en el otro a su correspondiente código de cliente y viceversa. Esto se consigue gracias a que en UtilGeneral.imprimirInicioPagina() tenemos la siguiente función javascript:

```
<script type='text/javascript'>  
    function copiarValor(idOrigen, idDestino) {  
        document.getElementById(idDestino).value =  
document.getElementById(idOrigen).value;  
    }  
</script>
```

Hay una alternativa, la forma más común de trabajar es tener un 'desplegable' para los nombres de cliente y un campo de texto oculto que refleja el código del nombre seleccionado en el 'desplegable'. Lo ocultamos por una simple razón: al usuario sólo le interesa ver los nombres de los clientes y su clave primaria (el código de cliente) suele resultarle indiferente (a menos que dicho código sea significativo, como por ejemplo el NIF). Para ocultar el código se puede probar a sustituir el 'desplegable' (select) de códigos por:

```
//// Alternativa: campo oculto para el código de cliente (en vez de SELECT de codigos)  
//    out.println("<INPUT TYPE=HIDDEN NAME='cliente.codigo' id='codigo'>");
```

Se queda oculto (hidden) y no borrado, ya que este campo será el que se transmita en la petición (request) al segundo servlet (FormVentas), pues dicho campo contiene el código del cliente del que queremos mostrar las ventas.

Transferencia de datos (session y request)

Pensemos en el paso de información del primer al segundo servlet. Analizando:

En la sesión se transporta la base de datos, login y password. Estos datos serán leídos por el segundo servlet.

En la request se transporta el código de cliente. Se ha podido ver que el segundo 'desplegable' tiene como 'name' 'cliente.codigo', este es el nombre de parámetro que usará el segundo servlet

(FormVentas) para obtener el código de cliente seleccionado. Concretamente por medio de: `request.getParameter("cliente.codigo")`.

Obteniendo la información de sesión (método `imprimirSesion()` de `UtilGeneral`)

Por medio de un iterador (Enumeration) obtenemos todos los atributos de la sesión:

```
static void imprimirSesion(HttpSession sesion, PrintWriter out) {  
    if (sesion != null) {  
        out.println("<P><B>Sesion:</B>" + sesion.getId() + ". Atributos:" + "<OL>");  
        for (Enumeration e = sesion.getAttributeNames(); e.hasMoreElements(); ) {  
            String atrib = (String) e.nextElement();  
            out.print("<LI>Nombre: " + atrib);  
            out.println(". Valor: " + sesion.getAttribute(atrib) + "</LI>");  
        }  
        out.println("</OL>");  
    }  
}
```

El aspecto más importante es que obtenemos el valor de un atributo por medio de:

```
sesion.getAttribute(atrib);
```

que devuelve un objeto del tipo `Object`. Podemos conseguir todos los atributos de una sesión, a partir de una enumeración devuelta por `sesion.getAttributeNames()` (de la misma forma que obteníamos todos los parámetros por medio de `request.getParameterNames()`):

FormVentas

El servlet que debe obtener las ventas de un cliente seleccionado recibe información por dos medios:

Recibe el código de cliente seleccionado como un parámetro de `request`.

Recibe como atributos de la sesión el nombre de la base de datos, login y password.

El procesamiento de la respuesta en el servlet 'FormVentas' es semejante al del servlet anterior, por ello no vamos a reincidir con detalles reiterados; como por ejemplo que se usa un DAO (`DAOVenta`). En `FormVentas.imprimirFormulario()` primero se envía al DAO la identificación (login y password) del usuario por medio de `dv.setIdentificacion()` y en segundo lugar se obtiene un vector de elementos de la clase `Venta`, por medio de una llamada a:

```
vecVentas = dv.select( "codigo = '" + request.getParameter("cliente.codigo")+"'");
```

El argumento de select() indica la cláusula WHERE.

Para que el formateo de la tabla de ventas sea correcto usamos un formateador de números para mostrar los separadores de millares y de decimales.

```
DecimalFormat myFormatter = new DecimalFormat( "###,###,###.##");
myFormatter.setMinimumFractionDigits(1);
....
out.println("<TD align=right>" + myFormatter.format( precio ) + "</TD>");
out.println("<TD align=right>" + myFormatter.format( coste ) + "</TD>");
out.println("<TD align=right>" + myFormatter.format( precio - coste )+ "</TD>");
```

El código completo de FormVentas es:

```
package docen_servlet01.JDBC01.presentacion;

import javax.servlet.ServletException;
import javax.servlet.http.*;
import javax.servlet.ServletConfig;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Vector;
import java.text.DecimalFormat;

import docen_servlet01.JDBC01.accesoDatos.DAOVenta;
import docen_servlet01.JDBC01.presentacion.UtilGeneral;
import docen_servlet01.JDBC01.bean.Venta;

/*****
 * Recibe el cliente (parámetro de formulario). Ejecuta una consulta de las
 * ventas de dicho cliente. Los datos son impresos en la página HTML.
 * Utiliza el DAOVenta para el acceso a la base de datos.
 *****/

public class FormVentas extends HttpServlet {
    private DAOVenta dv = null;

    /*****
     Al inicializarse el servlet se crea el DAO: en éste se carga el driver JDBC y se leen
     propiedades. El argumento del constructor del DAO es el path de la aplicación.
     Si ya se hubiesen cargado driver y propiedades, no se vuelven a cargar.
    *****/
```

```
*****/

public void init(ServletConfig config) throws ServletException {
    super.init(config);
    dv = new DAOVenta( config.getServletContext().getRealPath("/") );
}

/*****
 * Procesar una petición HTTP con el método POST
 * Muestro las ventas del cliente que se pasa como argumento del formulario
 *****/

public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    PrintWriter out= response.getWriter();          // Obtener flujo salida;
    try {

        HttpSession sesion = request.getSession( false );    // Obtener sesión
        response.setContentType("text/html; charset=iso-8859-1"); // Definir tipo
de salida

        UtilGeneral.imprimirInicioPagina( "Ejemplo de servlet", "Ventas del cliente
seleccionado", out);

        /// Si hay sesión, mostrar atributos y resto de página
        if ( sesion != null ) {
            UtilGeneral.imprimirSesion(sesion, out);

            imprimirFormulario( request, out );          // Imprimir salida
        }
        else
            UtilGeneral.imprimir(out, "La sesión no está disponible", true,
true);

    }
    catch (Exception e) {
        UtilGeneral.imprimir( out, "Error general. " + e.getMessage(), true, true );
    }
    finally {
        UtilGeneral.imprimirFinPagina( out );
    }
}
```

```

    }
    /*****
    * Procesar una petición HTTP con el método GET. Reenvia a doPost
    *****/
    public void doGet( HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doPost(request, response);
    }

    /*****
    Imprimo tabla de ventas

    *****/
    void imprimirFormulario( HttpServletRequest request, PrintWriter out ) {
        try {
            Vector vecVentas = null;
            HttpSession sesion = request.getSession(false);    // Obtener sesión

            //// DAO: asignar identificación (login-pwd)
            dv.setIdentificacion(                (String)sesion.getAttribute("login"),
(String)sesion.getAttribute("password"));

            //// Usar el DAO para conseguir vector de clientes. Argumento: el código
de cliente
            try {
                vecVentas      =      dv.select(      "codigo      =      '"      +
request.getParameter("cliente.codigo")+"'");
            }
            catch ( Exception e ) {
                UtilGeneral.imprimir(out,      "Error      en      la      consulta.      "      +
e.getMessage());
            }

            //// Inicio de tabla
            out.println("<P>Ventas al cliente " + request.getParameter( "cliente" ) +
":.");

            out.println("<TABLE BORDER=1 align='center'>");
            out.println("<TR bgcolor=#00CCFF>");
            out.println("<TH>CODIGO</TH>");
            out.println("<TH>PRECIO</TH>");

```

```
out.println("<TH>COSTE</TH>");
out.println("<TH>BENEFICIO</TH>");
out.println("</TR>");

///// Formateador de números
DecimalFormat myFormatter = new DecimalFormat( "##,###,###.##");
myFormatter.setMinimumFractionDigits(1);

if ( vecVentas.size() == 0)
    UtilGeneral.imprimir( out, "No hay ventas registradas", true, true );

///// Recorrer fila a fila y poner en tabla
for ( int i = 0; i < vecVentas.size(); i++ ) {
    Venta v = (Venta) vecVentas.get(i);

    out.println("<TR bgcolor=#00FF00>");
    out.println("<TD>" + v.getCodigo() + "</TD>");
    float precio = v.getPrecio().floatValue();
    float coste = v.getCoste().floatValue();
    out.println("<TD align=right>" + myFormatter.format( precio ) +
"</TD>");
    out.println("<TD align=right>" + myFormatter.format( coste ) +
"</TD>");
    out.println("<TD align=right>" + myFormatter.format( precio -
coste )+ "</TD>");
    out.println("</TR>");
}
out.println("</TABLE>");
}
catch (Exception e) {
    UtilGeneral.imprimir( out, "EROR EN FORMULARIO. " + e.getMessage(),
true, true );
}
}
```

JDBC: Inserción y actualización

Después de haber aprendido lo sencillo que es implementar una sentencia SELECT vamos a estudiar INSERT Y UPDATE.

INSERT

La diferencia más importante respecto a SELECT es que usamos la función `executeUpdate()`. En el siguiente ejemplo damos por supuesto que se han creado previamente los objetos de la clase `Connection` (con) y `Statement` (sentencia):

```
int add_cliente( String codigo, String nombre, String ape1, String ape2, int edad ) {
    try {
        String orden_SQL = "INSERT INTO CLIENTE VALUES " + "(" + codigo +
            ", " + nombre + ", " + ape1 + ", " + ape2 + ", " + edad +
        ")";

        Statement sentencia = con.createStatement();
        int fila = sentencia.executeUpdate(orden_SQL);
        sentencia.close();
        return fila;
    }
    catch (SQLException e) {
        e.printStackTrace();
        return 0;
    }
}
```

`executeUpdate()` devuelve el número de filas afectadas. Evidentemente, en nuestro ejemplo con INSERT si se produce la inserción devuelve uno y cero en caso de error. Conviene recordar que 'cliente.codigo' está definida como clave primaria (si hay intentos de insertar codigos repetidos se desencadena una excepción)

En este ejemplo se han insertado todos los valores en el orden en que están definidos los campos de la tabla. Podemos hacerlo de otra forma para insertar un registro que sólo tenga definido el campo 'cliente.codigo':

```
INSERT INTO cliente SET codigo = '31JK'
```

UPDATE

En el caso de UPDATE vamos a realizar un ejemplo en el que incrementamos el valor de las ventas un 10%:

```
update_venta( 1.1 );
```

```
/***** UPDATE de registros, devuelve el número de registros actualizados *****/
```

```
int update_venta( double incremento ) {  
    try {  
        String orden_SQL = "UPDATE venta SET precio = precio * " + incremento;  
        Statement sentencia = con.createStatement();  
        int fila = sentencia.executeUpdate(orden_SQL);  
        sentencia.close();  
        return fila;  
    }  
    catch (SQLException e) {  
        e.printStackTrace();  
        return 0;  
    }  
}
```

Si deseásemos realizar una actualización selectiva (por ejemplo, para las ventas mayores de 4500), tendremos que utilizar la cláusula WHERE.

EJERCICIO DE UTOEVALUACIÓN

La evaluación de JDBC (4 7.7 %)

Describir conceptos obtenidos en JDBC.

Se validan los conceptos de JDBC.

Seleccione la respuesta correcta a las siguientes preguntas según su conocimiento:

1. Desde la visión del programador cuales tipos de Driver se pueden trabajar en java?
 - a) El de Access java.
 - b) El de Mysql.
 - c) El de Oracle
 - d) El de Prolog
 - e) Todos los anteriores.
2. Defina que es JDBC ?
3. Un pproceso de persistencia se puede realizar minimo con dos capas si o no y por que?

EJERCICIOS DE APRENDIZAJE

Nombre del ejercicio de aprendizaje: JDBC	Datos del autor del taller: César Augusto Jaramillo Henao
<p>Escriba o plantee el caso, problema o pregunta:</p> <p>Desde hace muchos años según la evolución del pc, se han cambiado los métodos de acceder a la información, en algún momento eran los archivos y los tipos de estos que eran varios y de mucho alcance. Después de un tiempo se vio un cambio muy grande al pasar de archivos a BD, en este momento sucedió que existían lenguajes con su propio motor y existían lenguajes sin motor, pero con la posibilidad de acceder a cualquier tipo de BD, esto en su momento se conoció como ODBC, (Conectividad de Bases de Datos Abiertas), con la evolución del Java apareció el JDBD que es la Conectividad de Bases de Datos Java, esta herramienta es la que nos permite acceder a un conector o aun sistema de almacenamiento masivo sin importar el fabricante ya que Java no tiene un gran motor de BD, de esta manera se puede acceder a información almacenada en Oracle, SQL Server, MySQL, Access, entre muchos otros.</p> <p>¿Cuáles son las características comunes de uso?</p>	
<p>Solución del taller</p> <pre>import java.sql.*; ; public class Conexion { Connection cnn = null; public Conexion () { try { Class.forName("com.mysql.jdbc.Driver"); cnn = DriverManager.getConnection("jdbc:mysql://localhost/universidad", "root", "admin"); } catch(SQLException e) { System.out.println(e); } catch(ClassNotFoundException e) { System.out.println(e); } } public Connection getConnection () { return cnn; } }</pre> <p>Aquí encontramos las características más comunes de una conexión, como java no tiene su propio motor se realiza la vinculación mediante un conector, este tiene todas las características para comunicar todos los procesos que requiere el desarrollador, encontramos</p>	

que al establecer este conector invocamos por primera vez el jdbc para el motor MySQL, en la línea siguiente establece la conexión con una BD, su contraseña y su usuario para acceder a los datos que se tengan permiso.

Pista de aprendizaje:

Tener en cuenta: es una herramienta muy común y útil dentro del manejo de datos.

Tenga presente: conocer las características más comunes de la instrucción, le dará más alcance y más dominio del tema.

Traer a la memoria: identifique el API será de mucha ayuda en futuras aplicaciones.

3.4. El Sistema manejador de base de datos SGDB

La gestión del sistema de base de dato consiste en la administración, control y supervisión de la información contenida de manera integra para el servicio en su manipulación y utilización en su entorno para este caso en el servicio de Web.

Definición SGDB

"Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender toda las necesidades de los diferentes usuarios".
(<http://www.mailxmail.com/curso-procesamiento-datos-oracle/sistema-manejador-base-datos>, 2003)(Deen, 1985)

La arquitectura de la base de datos se basa en el estándar dado por ANSI/SPARC que se divide en 3 niveles (interno, conceptual y externo).

El externo es el nivel más cercano al usuario, describe la parte que interesa al usuario.

El conceptual describe cuales son los datos reales de la base y que relaciones existen entre los datos. Este nivel contiene la base de datos en términos de unas relaciones sencillas. Estas simples estructuras del nivel conceptual pueden estar reflejadas en complicadas estructuras físicas. Este es el nivel empleado por el administrador de la base de datos

El interno o físico es diferente de uno lógico. La operación de transformar registros lógicos en físicos y viceversa se llama transformación de datos o mapeo

Las operaciones corresponden a los métodos que realizan la manipulación: Buscar, Añadir, Suprimir y Modificar los datos contenidos en la Base de Datos incluye la creación y eliminación de tablas y base de datos.

Existe además un DBCS que es el sistema de control de la base de datos y permite el acceso a la definición de datos.

Lenguaje de manejo de datos: El DML (Data Management Language) es el que permite a los usuarios manejar o tener acceso a la base de datos. Permite recuperar, insertar o eliminar la información contenida. Existen dos tipos:

Sin procedimiento: donde se indican que datos se necesitan pero no como.

Con procedimiento: donde se indican que datos se necesitan y la forma como se necesitan.

Los lenguajes de 4a. Generación permiten la generación de reportes, visualización de gráficos o procesos de la base de datos de forma fácil y rápida.

Lenguajes de Programación: Estos son programas que pueden ser empleados por los programadores, algunos lenguajes de tercera generación tienen la capacidad de entrar a interactuar con bases de datos.

La estructura lógica, en el ámbito conceptual o externo, es la base para la clasificación de los DBMS en las cuatro categorías siguientes: **jerárquica, red, relacional y orientada a objetos**.

Cualquier categoría debe permitir un acceso aleatorio a los datos requeridos, utilizando para tal fin una estructura de datos: redes, árboles, tablas o listas enlazadas.

Método de Base de datos

Open Database Connectivity (ODBC) es un estándar de acceso a Bases de datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos (DBMS por sus siglas en inglés) almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda. Para que esto funcione tanto la aplicación como el DBMS deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el DBMS debe ser capaz de responder a ellos.

Para Access en forma grafica se establece el origen de datos desde inicio panel de control herramientas administrativas odbc o sea debe presentar esta imagen:

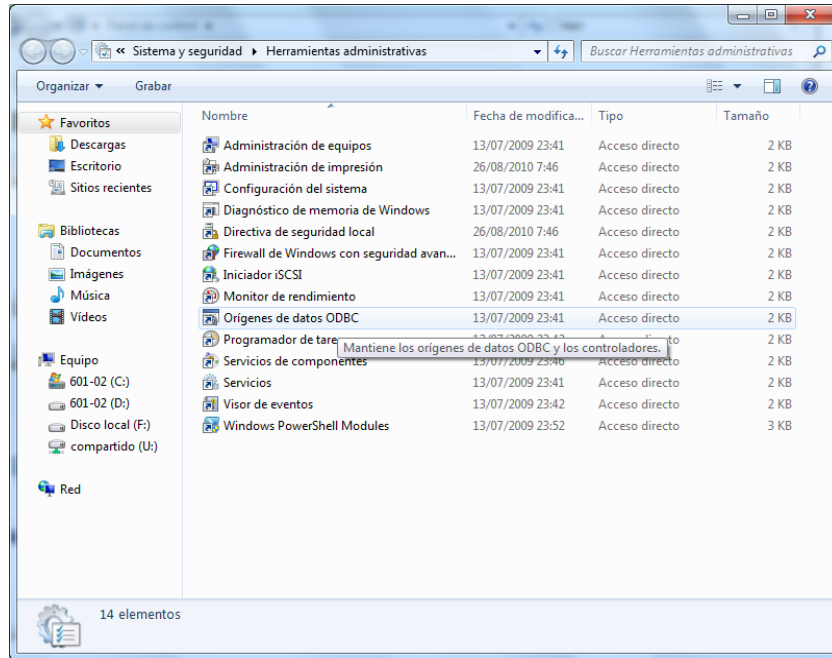


Gráfico Número 1: Selección del origen de datos para realizar el puente entre la aplicación de java bajo la api jdbc y el manejador de datos de Windows específicamente para Access pero se puede para Mysql

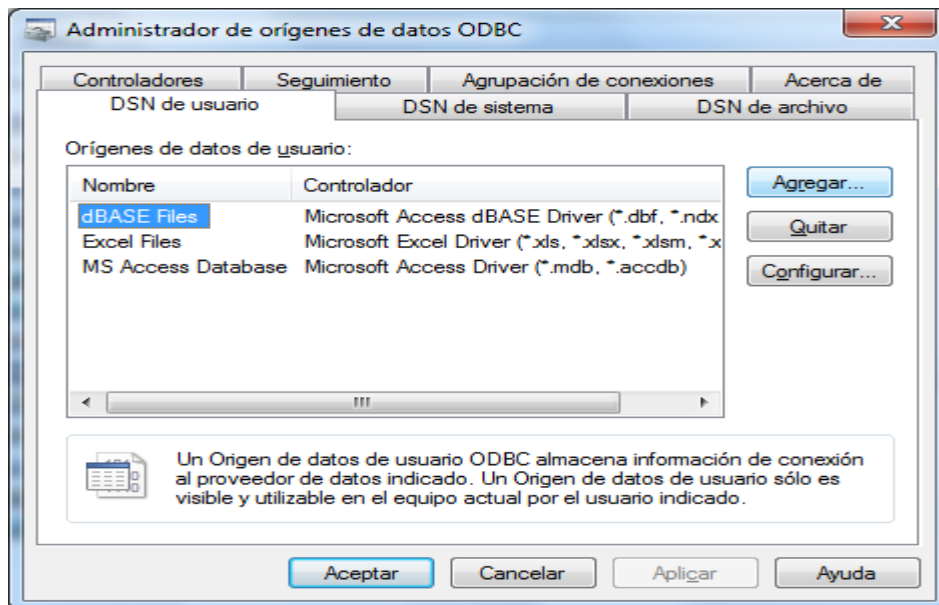


Gráfico Número 2: Agregar para obtener el driver correspondiente a el manejador de datos de Windows

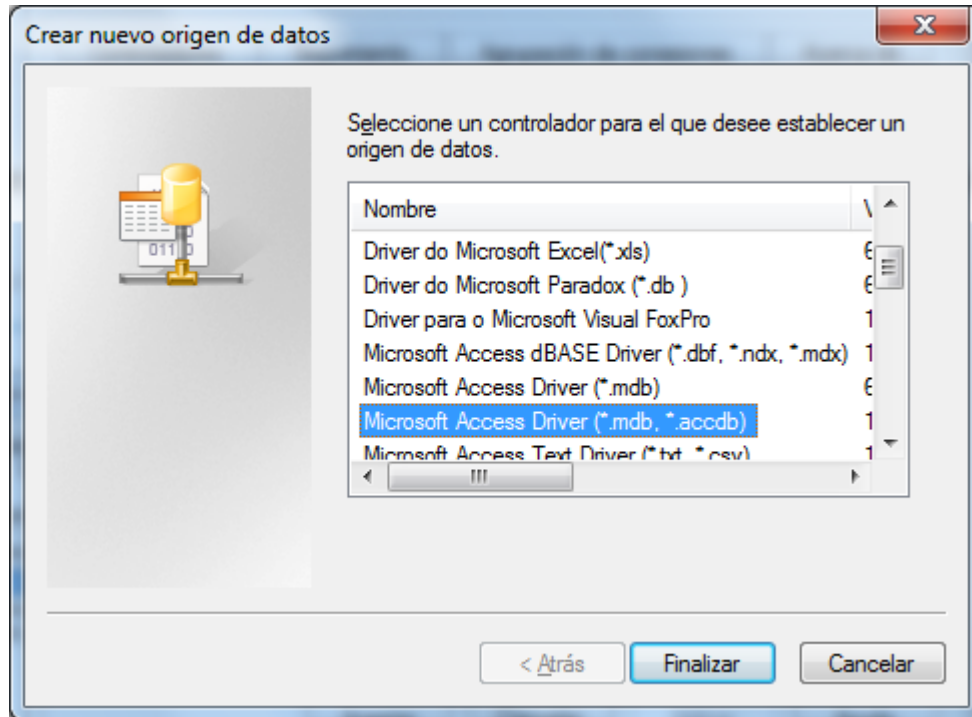


Gráfico Número3: seleccionar el de Microsoft Access Driver Mdb accdb

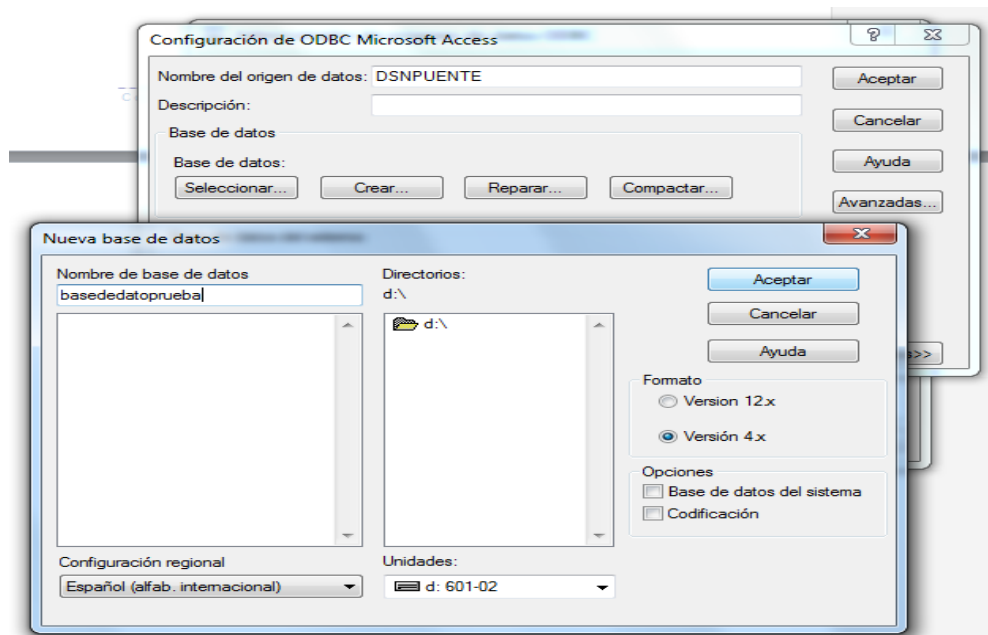


Gráfico Numero4: seleccionar o crear la base de datos

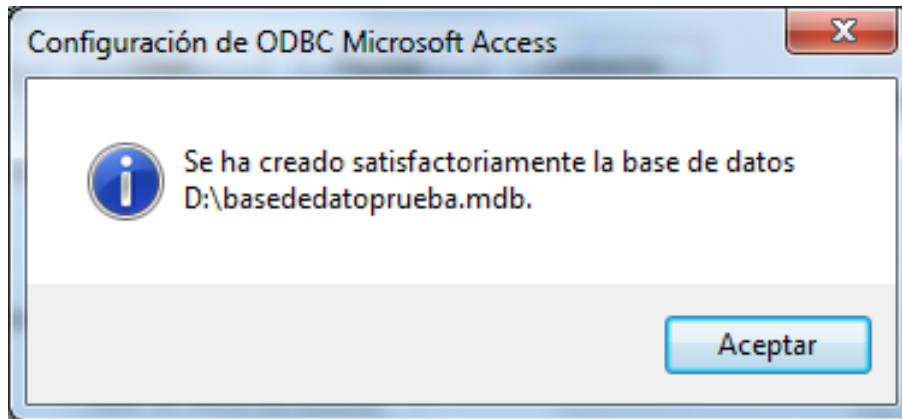


Gráfico Número5: Aceptar la selección de la base de datos

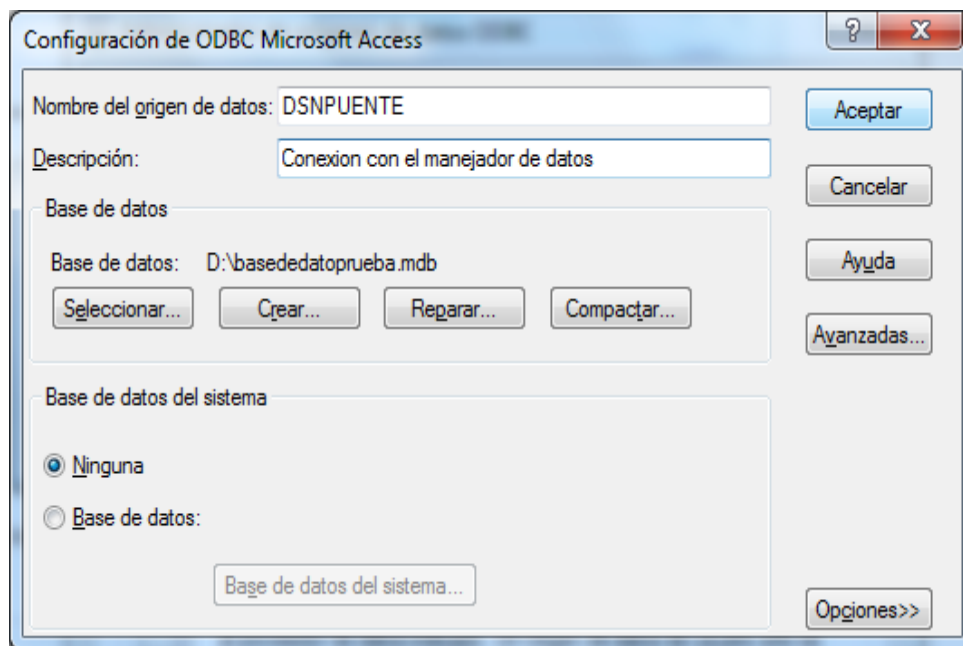


Gráfico Número6: Comprobar la conexión y lavase correspondiente

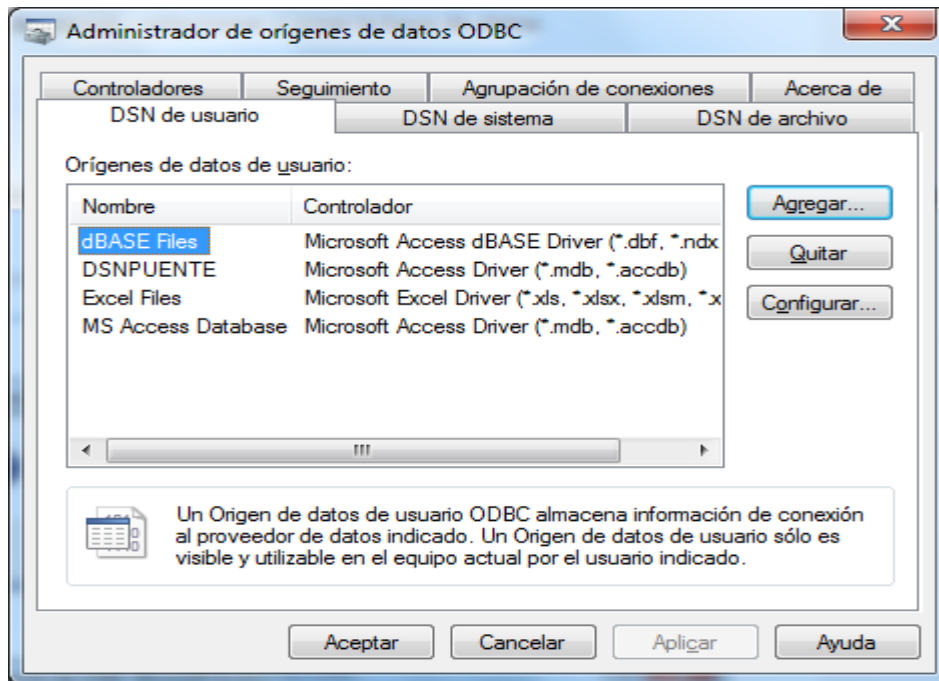


Gráfico Número7: Se encuentra registrado el nombre del origen de datos para la conexión entre el JDBC y el ODBC

Desde Mysql

```
mysql> create database prueba;
```

```
mysql> use prueba;
```

```
mysql> create table persona (id smallint auto_increment, nombre varchar(60), nacimiento date,
primary key(id));
```

```
mysql> insert persona values (NULL, 'Pedro', '1995-09-12');
```

Una vez instalada la base de datos, MySQL en nuestro ejemplo, vamos a ver de forma rápida como podemos conectarnos con ella y ejecutar los comandos SQL básicos: SELECT, INSERT, UPDATE y DELETE.

El driver con la base de datos

Para el caso de MySQL, podemos descargarlo de

<http://dev.mysql.com/downloads/connector/j/5.0.html>.

Nos bajamos el mysql-connector-java-5.0.5.zip, lo desempaquetamos en algún sitio y nos quedamos con el mysql-connector-java-5.0.5-bin.jar que viene dentro. En ese jar está la clase Driver que nos interesa.

Tendremos que poner ese jar accesible en nuestro proyecto. Dependiendo de qué utilicemos para programar en java hay muchas opciones.

Guardarlo en la carpeta <DIRECTORIO_DE_JAVA>/jre/lib/ext. Si has hecho una instalación por defecto en Windows, estará en C:\Archivos de Programa\Java\jdk1.5.0_05\jre\lib\ext.

Esta es la carpeta en la que java nos deja poner jar adicionales para que se encuentren por defecto. Es buena idea ponerlo aquí si vamos a hacer muchos programas con base de

```
C:\> set CLASSPATH=<PATH_DEL_JAR>\mysql-connector-java-5.0.5-bin.jar
```

Conectarnos con la base de datos

Una vez que java tiene el jar accesible y sabe dónde encontrarlo, ya podemos empezar con el código. Lo primero es conectarse con la base de datos

El código puede ser como este

```
import java.sql.Connection;  
import java.sql.DriverManager;
```

```
...
```

```
try
```

```
{
```

```
    Class.forName("com.mysql.jdbc.Driver");
```

```
    Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/agenda", "root",  
"LA_PASSWORD");
```

Asegurarse de que se inicializa el Driver y se registra. Para ello hay dos opciones. Hacer un new de él o bien instanciarlo con Class.forName("nombre de clase"), que es como hacer el new, pero de una forma rara.

La ventaja de hacerlo con Class.forName() es que sólo necesitaremos el Driver de una base de datos si lo usamos. Me explico. Imagina que haces un programa que permite conectarse con varias bases de datos distintas: MySQL, PostGres, Access

```
/* Todos estos import son necesarios para que compilen los news correspondientes */
```

```
import driver_de_mysql;
```

```
if (ha_elegido_mysql)
```

```
    new driver_de_mysql();
```

Sin embargo, con Class.forName() no necesitamos el import. Haremos los mismos if de antes, pero usaremos simplemente String distintos para llamar a Class.forName

```
if (ha_elegido_mysql)
```

```
    Class.forName("driver_de_mysql");
```

Una vez que nos hemos asegurado que java tiene el Driver cargado, simplemente pediremos conexión con la base de datos a la clase DriverManager.

```
Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/agenda", "root",  
"LA_PASSWORD");
```

DriverManager tiene muchos métodos getConnection() con parámetros variados. Todos son variantes de lo mismo y la información que suministramos es la misma. Aquí hemos utilizado uno con tres parámetros String, que vamos a explicar.

url: Es una cadena que nos permite localizar la base de datos. Para mysql, el formato es

"jdbc:mysql://ordenador_donde_corre_la_base_de_datos/nombre_base_datos". Donde se pone el nombre o IP del ordenador en el que se encuentra nuestro servidor de base de datos y el nombre de la base de datos. En nuestro ejemplo, tenemos el servidor de base de datos corriendo en el mismo ordenador que el programa java, por lo que ponemos localhost. La base de datos la he llamado agenda. El comando SQL para crear la base de datos agenda sería

```
mysql> CREATE DATABASE agenda;
```

user: Un usuario válido para la base de datos.

password: La clave del usuario.

Si todo va bien, tendremos nuestra conexión a la base de datos. Si va mal, saltará una excepción. Por eso es importante (y obligatorio para que compile) meter todo esto en un try-catch.

Esta forma de obtener una conexión está bien para aplicaciones sencillas, en el que únicamente se establece una conexión con la base de datos, no hay muchos hilos trabajando, etc. Si nuestra aplicación es algo más compleja/seria, en el que varios hilos pueden trabajar simultáneamente, en vez de obtener las conexiones directamente con DriverManager.getConnection(), es mejor obtenerlas a través de un Pool de Conexiones.

Creamos la tabla en Base de Datos... y la borramos.

Para enviar comandos SQL a la base de datos, se usa la clase Statement de java. Esta clase se obtiene a partir de la conexión, de esta forma:

```
Statement st = conexion.createStatement();
```

por supuesto, dentro de un try-catch.

Statement tiene muchos métodos, pero hay dos interesantes: executeUpdate() y executeQuery(). El primero se usa para sentencias SQL que impliquen modificaciones en la base de datos (INSERT, UPDATE, DELETE, etc). El segundo sólo para consultas (SELECT y similares).

Nuestra creación de tabla afecta a la base de datos, así que con executeUpdate().

```
st.executeUpdate("CREATE TABLE contacto (id INT AUTO_INCREMENT, PRIMARY KEY(id), nombre VARCHAR(20), apellidos VARCHAR(20), telefono VARCHAR(20))");
```

Esto crea una tabla contacto con cuatro campos: id, nombre, apellidos y telefono.

Para borrar la tablita esta, lo mismo, pero con DROP en vez de CREATE.

```
st.executeUpdate("DROP TABLE contacto");
```

Insertar datos en la base de datos

Vamos a hacer un esfuerzo de imaginación y supongamos que no hemos borrado la tabla contacto, que sigue existiendo en base de datos. Vamos a meterle datos.

```
String nombres[]={"Juan","Pedro","Antonio"};
String apellidos[]={"Gomez","Lopez","Alvarez"};
String telefonos[]={"123111","456222","789333"};
for (int i=0;i<nombres.length;i++)
    st.executeUpdate("INSERT INTO contacto (nombre, apellidos, telefono) VALUES ('"+nombres[i]+"','"+apellidos[i]+"','"+telefonos[i]+"')");
```

también en un try-catch.

Consultar datos de la base de datos

Vamos a hacer ahora una consulta de los datos que acabamos de insertar en la base de datos. Las consultas se hacen con `executeQuery()` y nos devolverán un `ResultSet`.

El `ResultSet` de alguna forma representa una conexión hacia los datos. En el `ResultSet` NO están todavía los datos. Según se los vayamos pidiendo, los irá trayendo de base de datos. Esto quiere decir que si una consulta devuelve muchos resultados, no se nos va a llenar la memoria por el hecho de hacer la consulta.

Para traer el primer resultado, debemos llamar el método `next()` del `ResultSet`. Para el siguiente otro `next()` y así sucesivamente hasta que `next()` devuelva `false`, indicando que ya no quedaban datos. El código puede ser así

```
ResultSet rs = st.executeQuery("SELECT * FROM contacto");
while (rs.next())
{
    System.out.println("nombre="+rs.getObject("nombre")+
        ", apellidos="+rs.getObject("apellidos")+
        ", telefono="+rs.getObject("telefono"));
}
```

`rs.close();`

Hemos hecho un `SELECT` para obtener los datos. Luego un bucle `while(rs.next())`, es decir, mientras `next()` vaya trayendo resultados.

Después de `next()`, el resultado recién traído está disponible en el `ResultSet`. La forma de recoger los campos es pedirlos con algún método `get()`. Si sabemos de qué tipo es el dato, podemos

pedirlo con `getInt()`, `getString()`, etc. Si no lo sabemos o nos da igual (como en este caso), bastará con un `getObject()`, capaz de traer cualquier tipo de dato.

En estos métodos `get()` podemos pasar como parámetro un entero, empezando en 1, que es el número del campo en el `SELECT`. Es decir, si hacemos `SELECT campo1, campo2, campo3...`, si pedimos `getObject(1)` obtenemos el valor de `campo1`, para `getObject(2)` el de `campo2`, etc.

Otra opción que a mi me gusta más, es pasar el nombre del campo, como se muestra en el código. Así, en el ejemplo de antes, `getObject("campo1")` nos devuelve el valor del `campo1`.

Update en la base de datos

Para `UPDATE` usamos `executeUpdate()`. Vamos a cambiar el número de teléfono de Juan. Ya que en la tabla hemos puesto un campo `id`, vamos primero a obtener el `id` de Juan para luego usarlo en el `UPDATE`. Esto no tiene por qué ser así, es sólo el ejemplo.

Para obtener el `id`:

```
rs = st.executeQuery("SELECT id FROM contacto WHERE nombre='Juan'");  
rs.next();
```

```
int id = rs.getInt("id");
```

Es un `SELECT` para obtener el campo `id` de Juan. Con `rs.next()` hacemos que venga el primer resultado de la consulta (y único en nuestro ejemplo). Como sabemos que es un `INT`, lo recogemos con `getInt()` y lo guardamos.

Para hacerlo bien, deberíamos ver si `rs.next()` devuelve o no resultado (quizás no haya nadie que se llame Juan en la base de datos) o si devuelve más de uno (más de un Juan en la base de datos), pero no vamos a complicarnos la vida en este ejemplo sencillo.

Ahora, sabiendo el `id`, podemos hacer el `UPDATE`.

```
st.executeUpdate("UPDATE contacto SET telefono='111' WHERE id="+id);
```

Esta vez, como el campo `id` es numérico, no hemos puesto las comillas simples.

Borrar datos de la base de datos

El borrado se hace con `DELETE` y `executeUpdate()`. Vamos a borrar a Pedro, que nos hemos peleado con él. Obtenemos primero su `id`, igual que antes.

```
rs = st.executeQuery("SELECT id FROM contacto WHERE nombre='Pedro'");  
rs.next();
```

```
id = rs.getInt("id");
```

y ahora lo borramos

```
st.executeUpdate("DELETE FROM contacto WHERE id="+id);
```

Cerramos conexiones

Los Connection, Statement y ResultSet con conexiones abiertas con base de datos. Debemos cerrarlas.

ResultSet se cierra solo cuando hacemos otra llamada execute() al Statement del que obtuvimos este ResultSet o bien cuando el recolector de basura "recolecta" al ResultSet. No nos preocupa en principio que se quede abierto, porque se acabará cerrando solo.

Eso sí, no podemos hacer nada con el Statement hasta que hayamos terminado con el ResultSet o se nos cerrará. Si necesitamos realizar otra cosa con base de datos, debemos crear otro Statement

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de SGBD (5 VALOR 7.7%)

Describir conceptos obtenidos en SGBD.

Se validan los conceptos SGBD.

Escriba la respuesta correcta a las siguientes preguntas según su conocimiento:

1. Desde la visión del programador en el SGBD existe un solo manejador de base de datos si o no y porque .
2. Defina que es una base de datos y por que lo diferencia de un archivo?
3. Es posible ingresar datos a una tabla (clase) de dos formas diferentes minimamente si o no y describa cuales?

Ejercicio de aprendizaje

Nombre del ejercicio de aprendizaje: SGBD	Datos del autor del taller: César Augusto Jaramillo Henao																																										
Escriba o plantee el caso, problema o pregunta: El Sistema General de Bases de Datos rige todo el contorno de almacenamiento, dentro de este encontramos las normas, las características, los tipos de datos, la normalización, las reglas para crear un modelo entidad relación, este tema es fundamental para la buena creación de una aplicación con altos estándares de calidad. Cree un tabla de estudiantes con los campos Carnet, nombre, fecha_matrícula, total asignaturas, promedio, mostrar su estructura con la descripción de cada campo.																																											
Solución del taller: Dentro de este tipo de estructuras encontramos: Describe estudiantes y obtenemos																																											
<table><tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr><tr><td>carnet</td><td>char(12)</td><td>NO</td><td>PRI</td><td>NULL</td><td></td></tr><tr><td>nombre</td><td>char(50)</td><td>NO</td><td></td><td>NULL</td><td></td></tr><tr><td>fecha_matricula</td><td>date</td><td>NO</td><td></td><td>NULL</td><td></td></tr><tr><td>totalasignaturas</td><td>int(3)</td><td>NO</td><td></td><td>NULL</td><td></td></tr><tr><td>promedio</td><td>float</td><td>NO</td><td></td><td>NULL</td><td></td></tr><tr><td>carnet_antiguo</td><td>char(10)</td><td>NO</td><td></td><td>NULL</td><td></td></tr></table>		Field	Type	Null	Key	Default	Extra	carnet	char(12)	NO	PRI	NULL		nombre	char(50)	NO		NULL		fecha_matricula	date	NO		NULL		totalasignaturas	int(3)	NO		NULL		promedio	float	NO		NULL		carnet_antiguo	char(10)	NO		NULL	
Field	Type	Null	Key	Default	Extra																																						
carnet	char(12)	NO	PRI	NULL																																							
nombre	char(50)	NO		NULL																																							
fecha_matricula	date	NO		NULL																																							
totalasignaturas	int(3)	NO		NULL																																							
promedio	float	NO		NULL																																							
carnet_antiguo	char(10)	NO		NULL																																							
Se observan los campos, los tipos, el tamaño de cada campo, si es o no requerido, la clave primaria, el valor por defecto y si tiene claves de autoincremento. A esto se refiere el SGBD, que se enfoca en establecer las características del manejo de la herramienta que almacena los datos.																																											

Pista de aprendizaje:

Tener en cuenta: de una buena construcción de la BD habrá un buen aplicativo.

Tenga presente: si el tiempo invertido para realizar las características de la BD lo tendrá que repetir n veces.

Traer a la memoria: cuando tenga muchas tablas es muy importante el análisis de cada una de ellas para que se establezca una relación acorde a las necesidades.

3.5. La implementación RMI

Se gestiona la atención al usuario (persona o empresa) explorando y describiendo sus actividades para el desarrollo del ciclo de vida del software, enfocando su intercomunicación según la estrategia requerida y proponiendo la tecnología además de la plataforma de soporte. Con ello se evalúa su creatividad, el aprovechamiento de la herramienta y sus conocimientos como un valor agregado para instalar e implementar el software e instanciarlo de una u otra manera según el entorno. Para este se estructuran los componentes que invocan a otros procesos no solo dentro de un único entorno sino dentro de diferentes de manera remota esto lo hace RMI (Java Remote Method Invocation) invocación de métodos de manera remota.

Definición APLICACIÓN

RMI (Java Remote Method Invocation) Es un mecanismo que permite invocar a un método en un objeto que existe en otro espacio de direcciones. El espacio de direcciones de otros podría ser en el mismo equipo o uno diferente definido por (eg.buckernell, 2007) al igual que CORBA, siendo mecanismo RPC (Remote Procedure Call) Llamada a Procedimiento Remoto otro tipo diferenciado por su estándar. Definido como un protocolo encargado de ejecutar una aplicación en otro entorno dentro de la misma o en diferente máquina eso si independiente de la comunicación así para (Berger, 2004) RPC, “es la transferencia sincrónica de datos y control entre dos partes de un programa distribuido a través de espacios de direcciones disjuntas”.

Método de proyectos

Existen tres elementos que deben ser tenidos en cuenta al invocar métodos remotos.

1. El cliente es el proceso que hace uso de un método en un objeto remoto.
2. El servidor es el proceso que posee el objeto remoto.
3. El Registro de objetos es un servidor de nombres que se relaciona con los nombres de los objetos. Aquí se registran para accederlo en cualquier entorno.

También se debe tener en cuenta la Serialización es una clase que al instanciarse se puede copiar en otro entorno a esto se le llama estado de operación remoto.

Esta es una clase la cual se importa en el paquete o se aplica básicamente cuando las aplicaciones cliente servidor se encuentran en diferentes entornos y comienza la otra etapa en este proceso remoto

La seguridad involucra el paquete lang con su clase System quien a su vez incluye SecurityManager y se activa por a setSecurityManager a las diferentes políticas estas emplean el metodo checkConnect en sus diferentes formas y relaciones

ACTIVIDAD ESTUDIO DE CASO

Primero se establece el servidor

```
import javax.swing.*;
public class GUIServidor extends JFrame
{
    private JTextArea areaTexto;

    public GUIServidor() {
        super("Servidor RMI");
        areaTexto = new JTextArea();
        areaTexto.setEditable(false);
        getContentPane().add(new JScrollPane(areaTexto));

        setSize(600, 400);
        setVisible(true);
    }

    public void anadirEntradas(String texto) {
        areaTexto.append(texto + "\n");
    }
}
```

Segundo el administrador del servidor RmiServidor.java

```
import java.net.InetAddress;
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.*;
import javax.swing.JFrame;
public class RmiServidor extends UnicastRemoteObject implements InterfazReceptorMensajes
{
    private static GUIServidor ventana;
    private int estePuerto;
```

```
private String estaIP;
private Registry registro;

public RmiServidor() throws RemoteException {
    try {
        // obtener la direccion de este host.
        estaIP = (InetAddress.getLocalHost()).toString();
        // asignar el puerto que se registra
    } catch (Exception e) {

        throw new RemoteException("No se puede obtener la direccion IP.");
    }

    try {
        // crear el registro y ligar el nombre y objeto.
        registro = LocateRegistry.createRegistry(estePuerto);
        registro.rebind("rmiServidor", this);
    } catch (RemoteException e) {
        throw e;
    }
}

public static void main(String[] args) {
    JFrame.setDefaultLookAndFeelDecorated(true);
    ventana = new GUIServidor();
    ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    try {
        new RmiServidor();
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

Desde consola se aplica la siguiente orden para obtener la interfaz inicial :

```
javac *.java rmic RmiServidor
```

Segundo desde el cliente


```
import java.rmi.*;
import java.rmi.registry.*;
import javax.swing.*;
import java.awt.*;import java.awt.event.*;
public class RmiCliente extends JFrame implements ActionListener {
    private JTextField cajaEnviar;
    private JButton botonEnviar;
    private JLabel estado;
    private static InterfazReceptorMensajes rmiServidor;
    private static Registry registro;
    private static String direccionServidor = "127.0.0.1";
    private static String puertoServidor = "3232";

    public RmiCliente() {
        super("Cliente RMI");
        getContentPane().setLayout(new BorderLayout());
        cajaEnviar = new JTextField();
        cajaEnviar.addActionListener(this);
        botonEnviar = new JButton("Enviar");
        botonEnviar.addActionListener(this);
        estado = new JLabel("Estado...");

        getContentPane().add(cajaEnviar);
        getContentPane().add(botonEnviar, BorderLayout.EAST);
        getContentPane().add(estado, BorderLayout.SOUTH);

        setSize(300, 100);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (!cajaEnviar.getText().equals("")) {
            enviarMensaje(cajaEnviar.getText());
            cajaEnviar.setText("");
        }
    }

    private static void conectarseAlServidor() {
        try {
            // obtener el registro
            registro = LocateRegistry.getRegistry(direccionServidor,
```

```
(new Integer(puertoServidor)).intValue());
// creando el objeto remoto
rmiServidor = (InterfazReceptorMensajes) (registro
.lookup("rmiServidor"));
} catch (RemoteException e) {
e.printStackTrace();
} catch (NotBoundException e) {
e.printStackTrace();
}
}

private void enviarMensaje(String mensaje) {
estado.setText("Enviando " + mensaje + " a " + direccionServidor + ":"
+ puertoServidor);
try {
// llamando el metodo remoto
rmiServidor.recibirMensaje(mensaje);
estado.setText("El mensaje se ha enviado!!!");
} catch (RemoteException re) {
re.printStackTrace();
}
}

Public static void main(String args[]) {
JFrame.setDefaultLookAndFeelDecorated(true);
conectarseAlServidor();
RmiCliente ventana = new RmiCliente();
ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
Este ejemplo fue tomado de (Berger, 2004) .
```

EJECICIO DE AUTOEVALUACIÓN

La evaluación de Implementacion de RMI (6 7.7%)

Describir conceptos obtenidos en RMI.

Se validan los conceptos de RMI.

Describe la respuesta correcta las siguientes preguntas según su conocimiento:

1. Desde la visión del programador en que consiste RMI?
2. Describa la diferencia entre RMI y RPC?
3. Es posible trabajar RMI en un solo entorno (una sola maquina) de manera remota si o no y porque?

Nombre del ejercicio de aprendizaje: RMI	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Java Remoto Method Invocation o invocación de métodos remotos de java, es una técnica avanzada y de mucha utilidad en el desarrollo de software de mediana y gran escala, que permite que se pueda acceder a un aplicativo de forma remota, situación que agiliza y permite que el aplicativo no esté limitado a un entorno de unos cuantos equipos. ¿Cuáles son las etapas de recorrido de este tipo de aplicativo?	
Solución del taller: Primera capa La primera capa es la de aplicación, la interfaz se usa básicamente para "marcar" un objeto como remotamente accesible. Segunda capa La capa dos es la capa proxy, o capa stub-skeleton. Tercera capa La capa tres es la de referencia remota. Cuarta Capa La capa cuatro es la de transporte	

Pista de aprendizaje:

Tener en cuenta: debe clasificar los aplicativos según su alcance.

Tenga presente: no todos los desarrollos llegan a un nivel de compartición remoto, ni a un nivel web.

Traer a la memoria: son procesos muy claros y específicos.

Taller de entrenamiento:

Nombre del taller: Aplicativo	Datos del autor del taller: César Augusto Jaramillo Henao
Actividad previa: Lenguaje de Programación III – Unidad 3.1 Bases de Datos JDBC.	
Describe la actividad: Crear un proyecto que contenga el manejo apropiado de capas, un sistema de por lo menos 5 tablas (tema Libre) debidamente relacionado y validado, implemente este desde un equipo principal y cree la estructura para accederlo desde otro pc de manera remoto con RMI.	

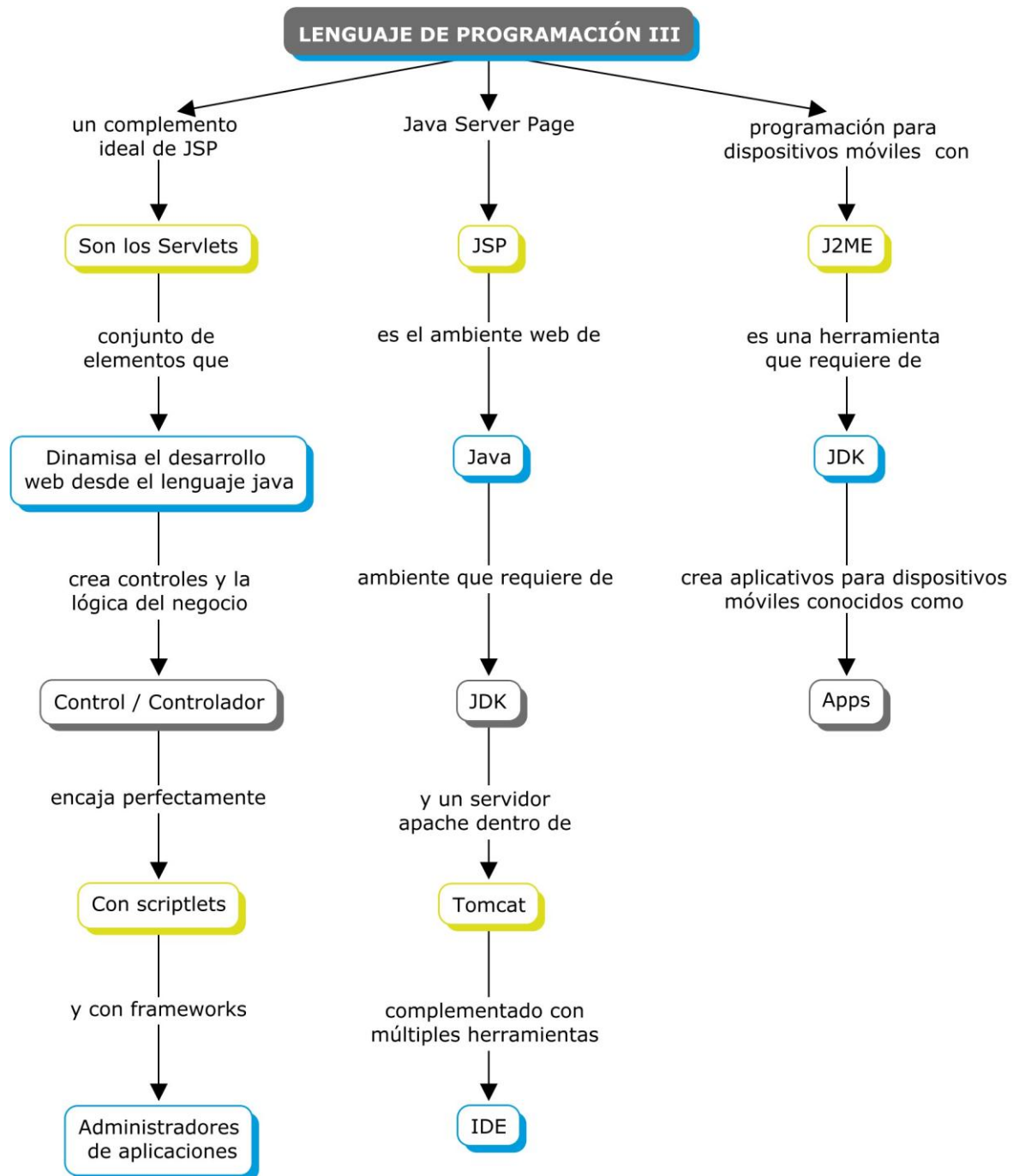
4. SERVLET

<http://www.youtube.com/watch?v=8pAvROEc-DU>



Imagen relacionada del video de youtube

4.1. Relacion de conceptos



OBJETIVO GENERAL

Proporcionar los conceptos básicos para la aplicación y comunicación de los Servlet.

OBJETIVOS ESPECÍFICOS

- Aplicar los procesos de comunicación (Gamma, 2005) por medio de los Servlet.
- Describir los componentes y aplicación de los Jsp.
- Explorar la plataforma para dispositivos inalámbricos electrónicos.

4.2. Prueba Inicial

Explorar los conceptos para el funcionamiento de los servicios Web Implementar los conocimientos alcanzados en recopilación de datos y los requerimientos de cliente - servidor

Se validan preconceptos de servicios con el explorador o Internet empleando o no el servidor.

Seleccione la respuesta correcta a las siguientes preguntas sobre Servlet según su conocimiento previo:

1. Un desarrollador de software emplearía Servlet para:
 - a) Ejecutar aplicaciones en el servidor.
 - b) Ejecutar aplicaciones en el manejador de base de datos.
 - c) Ejecutar aplicaciones en el cliente.
 - d) Ejecutar aplicaciones en el cliente -servidor
 - e) Ejecutar aplicaciones en el explorador
2. Un cookie lo emplearía para:
 - a) Recopilar información en el disco servidor a petición del visitante.
 - b) Recopilar información en el disco visitante a petición del visitante.
 - c) Recopilar información en el disco servidor a petición del servidor
 - d) Recopilar información en el disco visitante a petición del servidor
 - e) Recopilar información en el disco servidor a petición del sistema.
3. Un Java Server Pages JSP es una tecnología del java
 - a) Empleada para la creación de aplicaciones básicas.

- b) Empleada para la creación de páginas Web.
 - c) Empleada para la creación de páginas de servidor java
 - d) Empleada para la creación de páginas cliente.
 - e) Empleada para la creación de páginas guías.
4. JavaBeans es una tecnología del java
- a) Empleada para modelar objetos con la capacidad de reutilizarse.
 - b) Empleada para modelar diferentes componentes sin la capacidad de reutilizarse.
 - c) Empleada para modelar diferentes componentes.
 - d) Empleada para modelar diferentes componentes con la capacidad de reutilizarse.
 - e) Empleada para modelar diferentes componentes para elaborar una aplicación con la capacidad de reutilizarse.
5. Los Servlet son empleados
- a) Ejecutar aplicaciones en el servidor según requerimientos y respuestas.
 - b) Ejecutar aplicaciones en el cliente según requerimientos y respuestas
 - c) Ejecutar aplicaciones en el servidor y el cliente según requerimientos y respuestas.
 - d) Ejecutar aplicaciones en el cliente y servidor solo para respuestas.
 - e) Ejecutar aplicaciones en el servidor según las peticiones.
6. Un Servicio Web tiene como objeto:
- a) Realizar una aplicación en el computador sin emplear el explorador
 - b) Realizar una aplicación en el computador empleando el explorador
 - c) Realizar una aplicación en el computador sin emplear el explorador ni empleando interfaz HTML.
 - d) Realizar una aplicación en el computador empleando el explorador y emplea interfaz HTML o XML.
 - e) Realizar una aplicación en el computador

4.3. Servlet

Esta estructura esta conformada por módulos, se utilizan para ampliara la capacidad del despliegue (los servidores). Permitir la visualización de la página en forma dinámica, el usuario por intermedio de la Web (javax.servlet.). Realiza peticiones usuario y se le proporcionan sus respuestas

Un Servlet es un programa que se ejecuta en un servidor Web y que es utilizado para generar contenidos dinámicos en los sitios Web, se encarga de recibir una petición de un usuario y retornar un resultado de una búsqueda en una base de datos.

Pueden existir distintos tipos de Servlet, no sólo limitados a servidores Web y al protocolo HTTP(HyperText Transfer Protocol) entre otros

Introducción a los Servlet

Cada aplicación tiene que tener una estructura de directorios predeterminada. Empieza por un directorio raíz del contexto de aplicación, por ejemplo mi aplicación _html, cuya URL sería `http://www.host.com/public_html`.

Procesamiento de peticiones

Suele ser habitual tener el archivo `index.jsp` o `index.html` en este directorio (pero no obligatorio). Los subdirectorios que parten de este directorio raíz serán:
Directorios para vistas (JSP, HTML, imágenes).

Estos directorios no son imprescindibles y pueden llamarse como quiera.

WEB-INF. Este directorio es imprescindible y debe llamarse exactamente como aparece aquí. Contiene el archivo `web.xml`, es decir, el descriptor de despliegue de la aplicación, principalmente indica los servlets que utilizará la aplicación.

Cabeceras y códigos desde Servlet

Subdirectorios:

classes. Este subdirectorio es imprescindible si usamos archivos `.class`. Debe llamarse exactamente como aparece aquí. Los subdirectorios de "classes" serán los correspondientes a los paquetes de nuestras clases Java, reproduciendo la estructura de paquetes y subpaquetes.

lib. Necesario si se quiere incluir librerías, normalmente archivos jar. Debe llamarse exactamente como aparece aquí.

En WEB-INF se encuentra el archivo web.xml, donde se indican los Servlets contenidos en la aplicación. Hablaremos de este archivo cuando tengamos que referirnos a la instalación de los servlets. Por cierto, para hablar de forma estricta se dice "despliegue" (deployment) y no "instalación".

Otro aspecto interesante, además del archivo web.xml, es que los servlets están físicamente en WEB-INF/classes (si son .class) o en WEB-INF/lib (si son .jar). Ver como ejemplo examples/WEB-INF/classes, que viene por defecto en nuestra versión de Tomcat; donde no hay librerías y por tanto no hay directorio lib. Existe un directorio WEB-INF, ya que resulta imprescindible como receptáculo de web.xml y del directorio classes:

En realidad de WEB-INF puede colgar cualquier subdirectorio (hay quien pone el código fuente en un subdirectorio src).

Notas dignas de tener en cuenta:

WEB-INF y todo lo que cuelga de él es privado, es oculto al usuario. Intente en una instalación estándar de Tomcat acceder a web.xml, verá que no puede.

No caer en la falacia de que lo único que hay que poner en classes o lib son los servlets. Una clase cualquiera, que no sea un servlet puede colgar de classes (respetando la estructura de paquetes y subpaquetes). En suma, en los directorios de WEB-INF tendremos cualquier clase que utilice nuestra aplicación, sea servlet o no. Según (Lago, 2007)

Manejo de cookies y sesiones

Comunicación de Servlet

La estructura de directorios de nuestra aplicación no tiene que colgar necesariamente del directorio webapps, que viene por defecto en Tomcat. En el siguiente ejemplo la aplicación public_html está en un directorio independiente de los directorios de instalación de nuestro Tomcat. De hecho, es aconsejable por razones de mantenimiento que los directorios de nuestras aplicaciones estén fuera del espacio de directorios de Tomcat. (Lago, 2007)

Un ejemplo de Servlet

Un ejemplo de servlet nos permite entrar en el contexto de los servlet continuando con el ejemplo en el que se describe una página de "Bienvenido al mundo de J2ee":

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Saludo extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Bienvenido al primer servlet</title></head>");
        out.println("<body          bgcolor=\"#FFFF9D\"><FONT          color=\"#000080\"
FACE=\"Arial,Helvetica,Times\" SIZE=2>"+
                    "<CENTER><H3>Servlets</H3></CENTER>");
        Date d = new Date();
        out.println("<HR><p>i Bienvenido al mundo de J2ee!. Fecha y hora: " + d.toString() + ". Esto es
una aplicación Java.</p>");
        out.println("</body></font></html>");
    }
}
```

La mayoría de servlets heredan de HttpServlet.

La JVM del servidor de aplicaciones recibe una solicitud, para dar la respuesta realiza una serie de tareas:

Un hilo para cada solicitud. Normalmente se cumple que una solicitud corresponde con un hilo y todos los hilos comparten una única instancia del servlet (a menos que nuestra clase servlet herede de SingleThreadModel, que no es lo habitual). La JVM traduce la solicitud en la creación de un hilo, sobre el que se realiza una llamada a:

Método service(), heredado de HttpServlet, service(), si no lo sobrescribimos, invocará al método correspondiente (si el método de invocación al servlet es GET, entonces la JVM llamará al método doGet() de nuestro servlet, si el método de invocación es POST la JVM llamará a doPost()).

Ya hemos visto que la JVM invoca al método de entrada (doGet(), doPost(), etc). Pero además debe transferir al método de entrada dos objetos: uno es de la clase HttpServletRequest, que encapsula información diversa de la petición Http (cabecera, parámetros) y otro encapsula el flujo Http de respuesta, de la clase HttpServletResponse.

Esto sólo es un pequeño anticipo de los que es un Servlet o un JSP. Los detalles del ciclo de vida de estas clase/páginas vendrán más adelante.

En el siguiente ejemplo de formulario en HTML podemos ver como se invoca al servlet anterior:

```
<form action="../../servlet/hola_mundo" method="get" target=_blank >
<p><input type="submit" name="Submit" value="Pulse aqui para llamar al servlet que
saluda"></p>
</form>
```

El form en "real":

En la parte 'action' del formulario indicamos la URL o URI del recurso (en nuestro ejemplo es el servlet de saludo). Con el atributo 'method' indicamos el método de invocación. Con este método los parámetros de la solicitud aparecen explícitos, es decir, se incluyen en la URL de solicitud. Sin embargo con el método POST los parámetros quedan ocultos al cliente, ya que se transmiten en el cuerpo de la solicitud. El botón es el tipo 'submit', lo que indica que su pulsación disparará la acción (solicitud de recurso) (Microsystem, 2007).

Actividad estudio de caso

Los servlets hacen en init() la carga del driver y en doPost()/doGet se conectan a la base de datos y ejecutan una consulta que se escribe sobre la página.

Sólo los servlets (FormClientes y FormVentas) deben estar indicados en web.xml. según tomcat

Los servlets en web.xml:

```
<servlet>
  <servlet-name>FormClientes</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormClientes</servlet-class>
</servlet>
<servlet>
  <servlet-name>FormVentas</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormVentas</servlet-class>
</servlet>
```

```
<servlet-mapping>
  <servlet-name>FormClientes</servlet-name>
  <url-pattern>/servlet/FormClientes</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>FormVentas</servlet-name>
  <url-pattern>/servlet/FormVentas</url-pattern>
</servlet-mapping>
```

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de Servlet (7 VALOR 7.7.%)

Describir los conceptos obtenidos en Servlet.

Se validan los conceptos de los Servlet.

Describe la respuesta correcta según su conocimiento:

1. Desde la visión del programador que entiende por Servlet?
2. De cual clase reciben o heredan los servlet ?
3. Los servlet se cargan en el init() si o no y porque?

Nombre del ejercicio de aprendizaje: Servlet	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Los Servlets son pequeños programas (similares a los applets), que se ejecutan en un ambiente web. Dentro de esta plataforma pueden ir archivos de varias extensiones como los html, js, css, jsp, que son los archivos del lado del servidor. Pueden existir archivos .java en los que su pueden ubicar los archivos servlets. Cree un Servlet que imprima Hola Mundo.	
Solución del taller: Veamos un pequeño ejemplo del manejo de los servets. <pre>import java.io.IOException; import java.io.PrintWriter; import javax.servlet.ServletException; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest;</pre>	

```
import javax.servlet.http.HttpServletResponse

public class HolaMundo extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">");
        out.println("<html>");
        out.println("<head><title>Hola Mundo</title></head>");
        out.println("<body>");
        out.println("<h1>Hola Mundo</h1>");
        out.println("</body></html>");
    }
}
```

Obsérvese que es una mezcla de ambos mundos, contiene los paquetes como en java SE, se tiene una clase que hereda de HttpServlet y contiene una serie de etiquetas HTML que permite la visualización de la información que deseamos.

Pista de aprendizaje:

Tener en cuenta: los Servlets son complementos de un sitio web.

Tenga en cuenta: un Servlet es la mezcla de 2 mundos, el HTML, el java.

Traiga a la memoria: la mezcla de estas herramientas aprovechan un gran poder que separada no tendrían.

4.4. Java Server Pages

Su función básica (Allamaraju & al, 2000) es similar a los Servlet la independencia de la plataforma tiene dos capítulos, en algunas situaciones pueden representarse con modelos lineales, se encuentra con que sólo tienen sentido aquellas soluciones de la región factible en las que toda o algunas de las variables de decisión sean números enteros, además de su plataforma java2.

Definición JSP

Java Server Page, es una tecnología del modelo cliente-servidor que hace más dinámica el servicio web, permite el código del lenguaje java y administra las páginas de manera remota según los requerimientos.

Etiquetas o delimitadores en JSP

Para insertar código java dentro de una página html se deberán usar una serie de tags o delimitadores (por ejemplo se está usando `<% una serie de instrucciones de java %>` en los talleres ejemplo (http://www.programacionfacil.com/java_jsp/java_server_page, 2006)) donde cada uno de ellos tiene un propósito definido.

Comentarios `<%- comentario -%>` Ignorados cuando jsp es convertida a servlet y muy útiles para documentar nuestros programas jsp.

Declaración `<%! Variables, métodos, etc %>` Recordar que todo buen programa, empieza declarando variables.

Instrucción `<%= instrucción %>` Para poner una y solo una instrucción de java, además recordar que ya existen aparte ciertas instrucciones o variables predefinidas tales como request, response, out, session, application, config, and pageContext(también disponibles en scriptlets). Recordar además que cuando se use `<%= una sola instrucción %>`, la instrucción no debe terminar con punto y coma.

Scriptlet `<% todo un programa completo %>` Un scriptlet es un grupo de instrucciones de java, como se deduce de esta definición, se usará muchos scriptlets en nuestros jsp. Aquí sí, las instrucciones deben terminar con punto y coma

Un bloque de instrucciones `<% bloque java %>`, puede empezar (`<%`) en un scriptlet y terminar en otro scriptlet, pero asegurarse de que todos los scriptlets se abran y se cierren. Include Directive `<%@ include file="url" %>` Se usa para incluir archivos en la pc que compila la jsp, esto se realiza al tiempo que la jsp es convertida en servlet, el url debe ser relativo.

jsp:include action para incluir el archivo al tiempo de request por parte de un usuario remoto
Jsp:forward Action `<jsp:forward page="URL relativo"/>` utilizado para llamar una página.

Ciclo de vida de un JSP

Al crear objetos HTML se declaran en mayúsculas y cuando creamos variables java se declaran en minúsculas, no es una regla pero si es una buena practica de programación (<http://www.abcdatos.com/tutoriales/tutorial/z4303.html>, 2006).

Se esta usando un objeto HTML submit del cual se usan las propiedades NAME y VALUE, este objeto tiene como proposito principal activar la acción de la forma (llamarse a si mismo el programa.jsp) y ademas mandar los datos que proporciono el usuario hacia el servidor.

La ultima parte de la pagina es FORM, todos los objetos html deberan estar encerrados entre esta forma o ventana, form tiene dos propiedades la primera de ellas es una acción y en este caso la acción es pedirle al servidor que vuelva a ejecutarse el propio programa1 un Los metodo, es este caso es el metodo POST tambien se puede poner el metodo GET, capturan la información correspondiente desde el usuario hacia el servidor, pero POST lo hace en forma invisible y GET lo hace en forma publica.

Desarrollo de un JSP

Al principio del programa se deberan declarar e inicializar a 0 o "" todas las variables del programa. Se esta usando una instrucción IF para revisar el VALUE de SUBMIT, recordar que cuando el usuario pide por primera vez el programa al servidor , el VALUE del objeto SUBMIT es "null", ya que el usuario carga datos en los objetos TEXT y cuando realiza un click en SUBMIT, el value de SUBMIT pasa a valor generado en "evento1".

El objeto REQUEST.GETPARAMETER (VALUE OBJETO HTML) permite leer la propiedad VALUE de los objetos HTML asignandoselo a las variables java apropiadas.

Todos los datos que entran o salen de un objeto html seran de tipo string. Por tanto se esta usando el metodo parseInt (string) para convertir la cadena a tipo numérica entero.

Se usa una clase llamada FileOutputStream (<http://casidiablo.net/uso-del-objectoutputstream-java/>, 2008), especializada en archivos específicamente de escritura (salida) con muchos metodos y constructores para crear, manipular y procesar archivos el constructor usado solo lleva dos parametros, el primero todo la ruta o path a donde quedara el archivo(cuidado con no poner la doble diagonal \\) y el segundo parametro es la palabra "true", esto es para que el archivo se abra en modo llamado "APPEND", es decir que cada nuevo registro se vaya escribiendo al final del archivo, si no se pone este parametro(true), un nuevo registro se sobrescribiria sobre el registro anterior.

Sin embargo en el programa no se uso solo FILEOUTPUTSTREAM (solo para crear el archivo), tambien se usa DATAOUTPUTSTREAM
(http://www.programacionfacil.com/java_jsp/grabacion_archivos_disco, 2006).

Método de proyectos

“void” flush()” flujo de datos de entrada o salida

“int” size()” define el tamaño de la cadena o del flujo de datos o archivo.

write”(byte[] b, int off, int len)”

“void” Writes “len” bytes define el tamaño de la cadena , donde inicia lo define “off” para lo que se va escribir .

“void” write”(int b)” Escribir un character .

“void” writeBoolean”(boolean v)” Escribir un “boolean”

“void” writeByte”(int v)” Escribir un 1-byte value.

“void” WriteBytes (String s) Escribir un cadena.

“void” writeLong”(long v)” Escribir un “long”

“void” writeUTF (String str) Escribir un UTF-8 .

Observar que la grabación lleva un try-catch FileNotFoundException y IOException, que son obligatorios para el acceso a datos o no compila el programa.

No olvidar cerrar el archivo, con la instruccion archivo.close

Talleres experimentales 1

Empezamos con un pequeño ejemplo: prog1.jsp

```
<% // declarando
```

```
int base=0,altura=0; double area=0;
```

```
if(request.getParameter("OK") != null)
```

```
{ base = Integer.parseInt(request.getParameter(" Digite la BASE"));
```

```
altura = Integer.parseInt(request.getParameter("Digite la ALTURA"));
```

```
area= base * altura / 2.0 ; };
```

```
// Construyendo forma dinámica
```

```
out.println("<FORM ACTION=prog1.jsp METHOD=post>");
out.println("DAME LA BASE:<INPUT TYPE=TEXT NAME=BASE value="+base+"><BR>");
out.println("DAME LA ALTURA:<INPUT TYPE=TEXT NAME=ALTURA value="+altura+"><BR>");
out.println("AREA:<INPUT TYPE=TEXT NAME=AREA value="+area+"><BR>");
out.println("<INPUT TYPE=SUBMIT NAME=OK VALUE=evento1 ><BR>");
out.println("</FORM>");
%>
```

Para ejecutar : se realiza desde el localhost definido para tomcat

Talleres experimentales 2

Crear una aplicaion llamada programa.jsf para manejar almacenamiento temporal con parámetros <%!

```
//recordar que r es global
```

```
int r;
```

```
void decl1(int alfa[])
```

```
{
```

```
for(r=0;r<=2;r++)
```

```
alfa[r]=alfa[r]+10;
```

```
};
```

```
%>
```

```
<%
```

```
// no usar objetos request y out fuera de scriptlet
```

```
// porque no estan creados por java todavia
```

```
if(request.getParameter("OK") != null)
```

```
{
```

```
int eta[]={3,4,5};

decl1(eta);

for(r=0;r<=2;r++)

out.println("eta="+eta[r]+"<br>");

};

// construyendo forma dinamica
out.println("<FORM ACTION=programa.jsp METHOD=post>");

out.println("<INPUT TYPE=SUBMIT NAME=OK VALUE=evento1 ><BR>");

out.println("</FORM>");

%>
```

Trabajo colaborativos

Grabación y lectura son los dos procesos mas comunes con archivos donde se realiza almacenamiento en disco sea la aplicación llamada ProgramaArchivo.jsp

```
<%@ page import="java.io.*" %>

<%
// declarando la estructura

int clave=0;String nombre=""; float estatura=0;

// creando un objeto de tipo archivo

DataOutputStream archivo = null;

if(request.getParameter("INSERTAR") != null)

{

// capturando datos
```

```
clave=Integer.parseInt(request.getParameter("CLAVE"));

nombre=request.getParameter("NOMBRE");

estatura=Float.parseFloat(request.getParameter("ESTATURA"));

try {

    // creando archivo en append

    archivo = new DataOutputStream(new FileOutputStream("Ruta : \\archivoJsp.dat",true));

    // grabando al archivo

    archivo.writeInt(clave);

    archivo.writeUTF(nombre);

    archivo.writeFloat(estatura);

    out.println(clave+": registro grabado");

}

catch(FileNotFoundException fnfe) {}

catch (IOException ioe) {};

// cerrando el archivo

archivo.close();

};

// construyendo de forma dinamica

out.println("<FORM ACTION=prog17.jsp METHOD=post>");

out.println("CLAVE :<INPUT TYPE=TEXT NAME=CLAVE><BR>");
```

```
out.println("NOMBRE :<INPUT TYPE=TEXT NAME=NOMBRE><BR>");

out.println("ESTATURA :<INPUT TYPE=TEXT NAME=ESTATURA><BR>");

out.println("<INPUT TYPE=SUBMIT NAME=INSERTAR VALUE=GRABAR ><BR>");

out.println("</FORM>");

%>
```

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de JSP (8 VALOR 7.7 %)

Describir conceptos obtenidos en Jsp

Se validan los conceptos de JSP.

Describe la respuesta correcta a las siguientes preguntas según su conocimiento:

1. Desde la visión del programador que es JSP?.
2. Defina las funciones básicas de JSP ?
3. Es importante la aplicación de JSP en la web si o no y porque?

Nombre del ejercicio de aprendizaje: JSP	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: JSP, es el ambiente del lado del servidor utilizado por java, es una herramienta que entra directamente en competencia con ASP y PHP, con acceso a todas las características del java SE.	
Solución del taller: <pre><% double valor=Double.parseDouble(request.getParameter("valor")); int unidadini=Integer.parseInt(request.getParameter("unidadini")); int unidadfin=Integer.parseInt(request.getParameter("unidadfin")); if(unidadini<unidadfin) { int resul=unidadfin-unidadini; double total=Math.pow(1024,resul); double resultado=(valor/total); }</pre>	

```
        out.print("\n El resultado es "+resultado);
    }if(unidadini==unidadfin){
        out.print("\n El resultado es "+valor);
    }if(unidadini>unidadfin){
        int res=unidadini-unidadfin;
        double total=Math.pow(1024,res);
        double resultado=(valor*total);
        out.print("\n El resultado es "+resultado);
    }
    %>
```

Concentramos en este ejemplo un caso típico de jsp, se ve que la mayor parte del código es netamente java aunque su propicito sea distinto.

En él encontramos tipos de datos double, int, tipos referenciales como Double y el Integer. Cambia en su codificación la recepción de los valores extornos como el request.getParameter. Las condiciones tienen la misma estructura de java, en la impresión cambia al no tener que utilizar System.out.print y se utiliza out.printl, los cálculos matemáticos son los mismos de java.

Pista de aprendizaje:

Tener en cuenta: JSP es una herramienta web que requiere de un servidor que lo interprete, por ejemplo el TomCat.

Tenga presente: la mayor parte de la codificación es java.

Traiga a la memoria: que el alcance en java que es distinto en jsp, la que tiene como finalidad la Web.

4.5. Móvil J2ME

Estructuras para manipular información grafica o no de manera dinámica por medio de una plataforma apropiada para dispositivos inalámbricos electrónicos como los teléfonos móviles llamados celulares, PDAs, cámara, GPS o similares con una baja capacidad de byte requerida para su utilización . Siendo una componente de la plataforma java para comunicación.

Definición J2ME

El teléfono móvil es un dispositivo inalámbrico electrónico que permite tener acceso a la red de telefonía celular o móvil. Se denomina celular en la mayoría de países latinoamericanos debido a las antenas repetidoras que conforman la red, cada una de las cuales es una célula, si bien existen redes telefónicas móviles satelitales (Perez, 2011). Su principal característica es su portabilidad, que permite comunicarse desde casi cualquier lugar. Aunque su principal función es la comunicación de voz, como el teléfono convencional.

J2ME por lo tanto, se divide en configuraciones, perfiles, y paquetes opcionales.

A partir del siglo XXI, los teléfonos móviles han adquirido funcionalidades que van mucho más allá que limitarse a llamar o enviar mensajes de texto, se podría decir que se ha unificado (que no sustituido) con distintos dispositivos tales como PDA, cámara de fotos, agenda electrónica, reloj despertador, calculadora, microproyector, GPS o reproductor multimedia, así como poder realizar multitud de acciones en un dispositivo pequeño y portátil que lleva prácticamente todo el mundo de países desarrollados. A este tipo de evolución del teléfono móvil se le conoce como smartphone.

La configuración J2ME

Sun Microsystem establece dos entornos normalizados para la implementación de las aplicaciones java además de J2SE (Java 2 Standard Edition) (Gálvez Rojas & Ortega Diaz, 2004) para las aplicaciones independientes, que es la base de esta tecnología con sus aplicaciones básico java y applet , la primera SE VISUALIZA en la pantalla en ambiente dos o grafica y la segunda desde el explorador incluyendo a html o xml, los otros son J2EE y J2ME.

J2EE (Java 2 Enterprise Edition) adiciona los elementos de comunicación para desarrollo empresarial (flujo de datos almacenamiento permanente, redes entre otros) esto es justificado por ser cada elemento independiente en su implementacion y requeridas solo al momento de usarlas encajando en su estructura objetual.

J2ME (Java 2 Micro Edition) (desarrollomovil, 2006) específicamente configura los mínimos requisitos de hardware y software, para aplicaciones de dispositivos de capacidad más reducidas del requerimiento normal pero con colecciones de tecnología haciéndolo más portable y permitiendo más amplitud en su comunicación empleando redes intermitentes para obtener una máquina virtual Java de menor kilate binario KVM para solucionar las peticiones del mercado estas APIs.

Es conocida como APIs CLDC (connected Limited Device Configuration) enfocada a dispositivos con restricciones de procesamiento y memoria, básicamente proporciona las clases de J2SE, además las MID contienen las clases para crear interfaz y Connected Device Configuration APIs (CDC) enfocada a dispositivos con más recursos.

LA CONFIGURACIÓN CLDC define:

Las características del lenguaje Java incluidas.

La funcionalidad será incluida en la máquina virtual Java.

Incluir las APIs necesarias para el desarrollo de aplicaciones en móviles.

Los requerimientos Hardware de los dispositivos

Excluye los soportes de operadores matemáticas en punto flotante

NO PERMITE método de FINALIZACIÓN

DISMINUYE EL NÚMERO DE EXCEPCIONES.

Requerimientos

Corre sobre la KVM

Bajo consumo de energía

Soporta conectividad a la red

❏ Memoria disponible para entorno Java: 150 a 500 Kb.

❏ LA CONFIGURACIÓN CDC define:

❏ Corre sobre una máquina virtual C: CVM

❏ Los dispositivos de esta categoría tienen importante capacidad de procesamiento

❏ Soporta conectividad a la red

❏ Memoria disponible para entorno Java: 500 Kb. a 2Mb.

El perfil J2ME

Las estructuras se caracterizan por ser una configuración pero agrega un API más específico para hacer un ambiente completo para su uso. Mientras que una configuración describe un JVM y un

sistema básico de APIs, por sí mismo no especifica bastante detalle para permitirle construir usos completos. Los perfiles que son unas bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos; incluyen generalmente APIs para el ciclo vital de uso, el interfaz, y el almacenaje persistente.

La función de los Perfiles en un entorno de ejecución determinado de J2ME se compone de una selección de:

Máquina virtual
Configuración
Perfil.

Paquetes Opcionales J2ME

Mobile Information Device Profile

Este perfil está construido sobre la configuración CLDC. Al igual que CLDC fue la primera configuración definida para J2ME, MIDP fue el primer perfil definido para esta plataforma.

Este perfil está orientado para dispositivos con las siguientes características:

Reducida capacidad computacional y de memoria.

Conectividad limitada (en torno a 9600 bps).

Capacidad gráfica muy reducida (mínimo un display de 96x54 pixeles monocromo).

Entrada de datos alfanumérica reducida.

128 Kb de memoria no volátil para componentes MIDP.

8 Kb de memoria no volátil para datos persistentes de aplicaciones.

32 Kb de memoria volátil en tiempo de ejecución para la pila Java.

Los tipos de dispositivos que se adaptan a estas características son: teléfonos móviles, pagers o PDAs de gama baja con conectividad.

El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con: La aplicación (semántica y control de la aplicación MIDP).

Las características MIDP 2.0 incluyen:

Seguridad, usando HTTPS

Parser XML

API para sonido

Inclusión OTA (Over the Air)

Mejoras en las capacidades de interfaz de usuario

Requerimientos para MIDP

32 Kb. de memoria volátil para correr Java

Un tipo de entrada para usuario

8 Kb. de memoria no volátil para almacenar datos persistentes de las aplicaciones

128 Kb. de memoria no volátil para correr componentes MID

Conectividad inalámbrica a redes

Pantalla 96 x 54 px.

Los paquetes ocasionales J2ME

Un paquete opcional proporciona la funcionalidad que no se puede asociar a una configuración o a un perfil específica. Un ejemplo de un paquete opcional es el Bluetooth API (JSR 82), que proporciona un API estandarizado para usar el establecimiento de una red de Bluetooth. Este paquete opcional se podía poner en ejecución junto a virtualmente cualquier combinación de configuraciones y de perfiles.

Las configuraciones son las especificaciones que detallan una máquina virtual y un sistema bajo de APIs que se pueda utilizar con cierta clase del dispositivo. Una configuración, por ejemplo, se pudo diseñar para los dispositivos que tienen menos de 512 KB de memoria y de una conexión de red intermitente.

Existen 2 configuraciones definidas en J2ME: Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria, y Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

Elemento Relacionales

Jambi es una implementación en Java de Qt y por tanto, hereda toda la "filosofía" (por llamarlo de alguna manera) de este. Es por esto que debemos conocer algunas cosas respecto a la forma de "ver las cosas" que tiene Qt antes de ponernos a programar.

En la implementación de Qt se basa en componentes y contenedores llamados widget (Cancela, 2007), desde un botón, hasta una ventana, e incluso podemos agrupar widgets y crear uno propio. Sin más, podemos decir que una aplicación Qt es un widget que a su vez contiene un conjunto de widgets en su interior que pueden estar formados por otro conjunto de widgets.

Estos widgets, se comunican unos a otros mediante lo que Qt llama "Signals and slots", esto es, los widgets emiten señales (signal) que son tomados por un slot específico de otro widget lo que provoca alguna acción.

Un midlet en J2ME

Un midlet es una aplicación desarrollada en J2ME para descargar en un teléfono móvil. Con un tamaño medio de 50 KB Tipos de midlets: (García Serrano, 2006)

- ❑ Juegos, que pueden necesitar de conexión o no (ejemplo, juegos en red o juegos monopuesto)
- ❑ Personalización del móvil, para las descargas de logos, melodías, salvapantallas, etc.
- ❑ Aplicaciones generales, como servicios de localización, mapas, información meteorológica, guía de establecimientos, etc.
- ❑ Aplicaciones específicas para cada empresa, y desarrolladas según sus propias necesidades (por ej. listado de productos en stock)

Actividad estudio de caso

Primero instalar el entorno de programación de J2SE (JDK). Puedes descargar la última versión de JDK desde la URL <http://java.sun.com/j2se/downloads.html>. Una vez descargado e instalado, estaremos en condiciones de descargar e instalar J2ME desde la URL <http://java.sun.com/j2me/download.html>. El entorno de desarrollo que nos provee el Wireless Toolkit se llama KToolBar.

Compilando el primer MIDlet.-

Vamos a construir uno paso a paso nuestro primer MIDlet usando esta herramienta. Tras la instalación del wireless toolkit, tendremos un nuevo submenú en el menú inicio.

Selecciona la aplicación KToolBar e inicializa el entorno.

Crear un nuevo proyecto, así que pulsamos el botón New Project. Nos solicitará un nombre para el proyecto y otro para la clase principal de la aplicación.

Tanto el proyecto como la clase principal se llamarán HelloWorld, así que introducimos este nombre en ambos cuadros de texto y pulsamos el botón Create Project. En este momento KToolBar crea la estructura de directorios necesaria para albergar el proyecto.

Cada una de las carpetas creadas tiene una misión concreta. Por ahora nos bastará saber que nuestros archivos fuente irán emplazados en el directorio src, y los recursos necesarios como gráficos, sonidos, etc... se alojarán en el directorio res

A diferencia de otros entornos de programación, KToolBar no cuenta con un editor integrado para editar los programas, por lo tanto vamos a utilizar uno externo. Puedes utilizar el bloc de notas de Windows o jcreator.

Otro sugerido es Crimson Editor (<http://www.crimsoneditor.com/>), también soporta Javaejemplo tomado de (pusivus, 2005)

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloWorld extends MIDlet implements CommandListener {
    private Command exitCommand;
    private Display display;
    private Form screen;

    public HelloWorld() {
        // Obtenemos el objeto Display del midlet.
        display = Display.getDisplay(this);

        // Creamos el comando Salir.
        exitCommand = new Command("Salir", Command.EXIT,2);

        // Creamos la pantalla principal (un formulario)
        screen = new Form("HelloWorld");

        // Creamos y añadimos la cadena de texto a la pantalla
        StringItem saludo = new StringItem("", "Hola Mundo...");
        screen.append(saludo);

        // Añadimos el comando Salir e indicamos que clase lo manejará
        screen.addCommand(exitCommand);
        screen.setCommandListener(this);
    }
    public void startApp() throws MIDletStateChangeException {
        // Seleccionamos la pantalla a mostrar
        display.setCurrent(screen);
    }

    public void pauseApp() {
    }
}
```

```
public void destroyApp(boolean incondicional) {  
}  
  
public void commandAction(Command c, Displayable s) {  
    // Salir  
    if (c == exitCommand) {  
        destroyApp(false);  
        notifyDestroyed();  
    }  
}  
}
```

En el desplegable Device puedes seleccionar el emulador que quieres utilizar para seleccionar el grafico o la imagen correspondiente al celular.

El DefaultColorPhone tiene soporte de color, así que te resultará más atractivo. Pulsa el botón Run. Verás aparecer un emulador con forma de teléfono móvil. En la pantalla del móvil aparece un menú con un sólo programa llamado HelloWorld. Pulsa select para ejecutarlo.

En KToolBar despliega el menú project, y selecciona create package del submenú package. KToolBar nos informa de que ha creado los archivos HelloWorld.jar y HelloWorld.jad dentro del directorio bin. Estos son los archivos que habremos de transferir al teléfono móvil.

java.lang
java.util
java.io

Además cuenta con el "Generic Connection Framework" que ofrece posibilidades de conexión y comunicación.

MIDP contiene los siguientes paquetes:

javax.microedition.midlet
javax.microedition.lcdui
javax.microedition.io
javax.microedition.rms

El paquete javax.microedition.midlet, es importante contiene la clase MIDlet, que ejecuta aplicaciones de dispositivos móviles.

El paquete javax.microedition.lcdui crea interfaces de usuario es un AWT básico.

Trabajar con 'Screens' sobre las que podremos colocar elementos de la interfaz de usuario, como textos, menus, etc., por otro, podremos basar nuestras aplicaciones en 'Canvas' sobre las que podemos trabajar a nivel gráfico, es decir, a más bajo nivel. Tanto Screen como Canvas son objetos que heredan de la clase 'Displayable'.

Método de proyectos

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de J2ME (9 VALOR 7.7%)

Describir conceptos obtenidos en J2ME.

Se validan los conceptos de las Apis al terminar la unidad de J2ME.

Seleccione la respuesta correcta a la primera pregunta según su conocimiento previo:

Describa su respuesta a las preguntas 2 y 3 según su conocimiento:

1. Cuáles son las características del lenguaje Java excluidas en la configuración de J2ME?
 - a) Las APIs necesarias
 - b) El método finalizador (eliminar los objetos)
 - c) Ninguna de las dos.
2. Defina las APIs que utiliza?
3. La orden **import javax.microedition.midlet.*;** se puede aplicar en j2me si o no y porque?

Nombre del ejercicio de aprendizaje: J2ME	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: <p>J2ME, es la plataforma de Oracle para desarrollo de aplicativos en dispositivos móviles (ME Micro Edition), aquí la codificación se conserva en un porcentaje muy alto, esto es un beneficio muy importante porque desde el mismo ambiente Java se podrán desarrollar aplicaciones para JSP y ME.</p> <p>Cree un método en J2ME que determine si se almaceno un registro</p>	
Solución del taller: <p>Un pequeño método del J2ME</p> <pre>private void writeData(String data) { byte[] rec = data.getBytes(); try { recordID = miRS.addRecord(rec, 0, rec.length); } catch (Exception e) { append("Error al guardar los datos"); } }</pre> <p>Obsérvese que los métodos tienen la misma estructura, igual sucede con la creación del arreglo, control de los datos con try y catch.</p> <p>Tiene particularidades como los MIDLET's, que se denominan aplicativos pequeños para dispositivos con recursos limitados, esto fue muy común en los últimos 10 años, hoy en día esas limitantes no son tan marcadas, hoy se tienen dispositivos de mayor almacenamiento y mayor procesamiento de la información.</p>	

Pista de aprendizaje:

Tener en cuenta: crea aplicaciones más pequeñas porque es para un dispositivo como un teléfono o una Tablet.

Tenga presente: que todo se rige por la normatividad de java

Traer a la memoria: que la codificación es idéntica a java aunque con un propósito distinto

Taller de entrenamiento:

Nombre del taller: Micro Edition	Datos del autor del taller: César Augusto Jaramillo Henao
Actividad previa: Lenguaje de Programación III – Unidad 4.3 Movil J2ME.	
Describe la actividad: Cree un aplicativo para J2ME que represente una calculadora simple con sus funciones tradicionales como suma, resta, multiplicación, división, potenciación y raíz cuadrada.	

5. PROGRAMACIÓN Y TECNOLOGÍA

http://www.youtube.com/watch?v=Kkn_8BcmopA&playnext=1&list=PLB7BB551126EDD5E0&feature=results_main

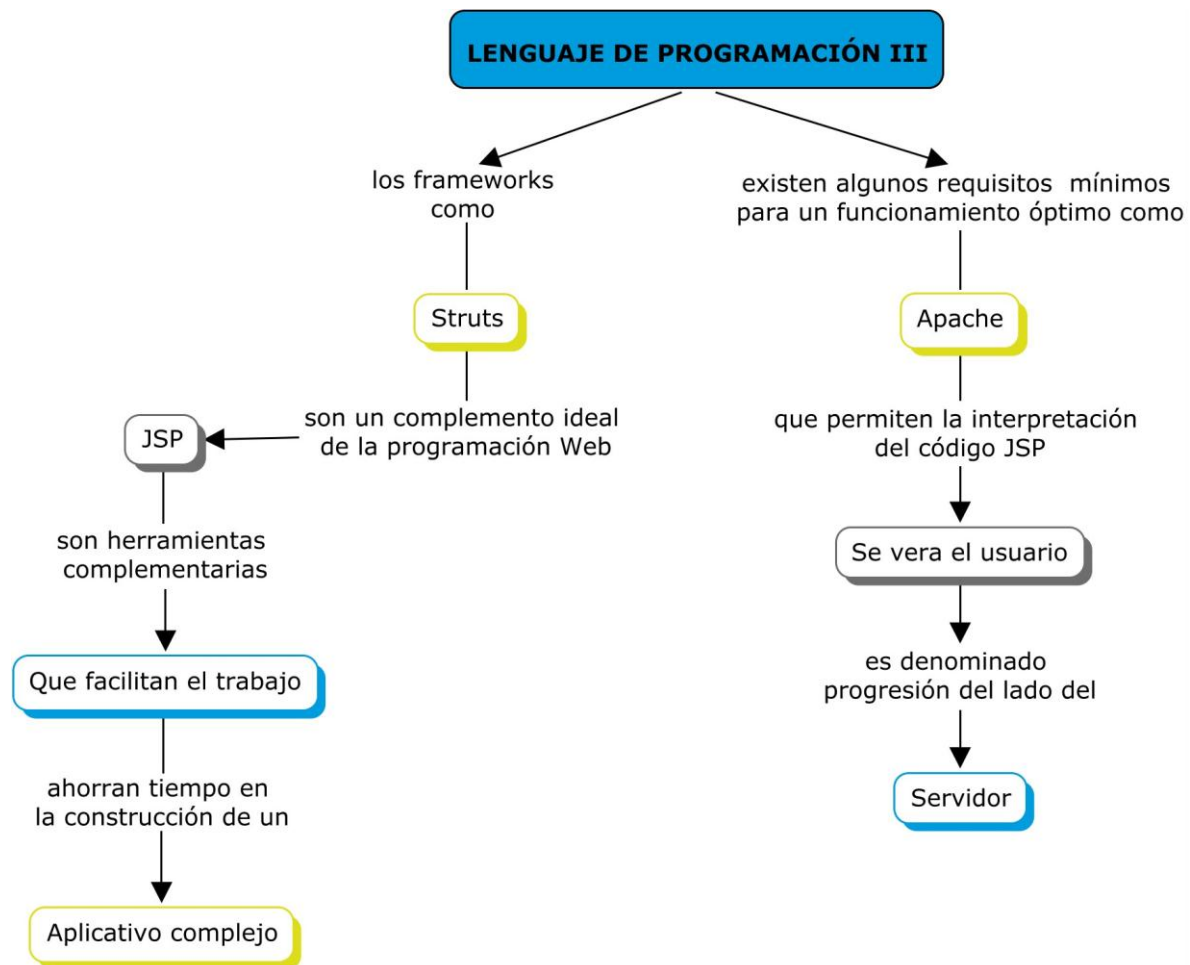
```

121 </result-types>
122
123 <interceptors>
124   <interceptor name="alias" class="com.opensymphony.xwork2.interceptor.AliasInterceptor"/>
125   <interceptor name="autowiring" class="com.opensymphony.xwork2.spring.interceptor.ActionAutowiringInterceptor"/>
126   <interceptor name="chain" class="com.opensymphony.xwork2.interceptor.ChainingInterceptor"/>
127   <interceptor name="conversionError" class="org.apache.struts2.interceptor.StrutsConversionErrorInterceptor"/>
128   <interceptor name="cookie" class="org.apache.struts2.interceptor.CookieInterceptor"/>
129   <interceptor name="clearSession" class="org.apache.struts2.interceptor.ClearSessionInterceptor"/>
130   <interceptor name="createSession" class="org.apache.struts2.interceptor.CreateSessionInterceptor"/>
131   <interceptor name="debugging" class="org.apache.struts2.interceptor.debugging.DebuggingInterceptor"/>
132   <interceptor name="executeAndWait" class="org.apache.struts2.interceptor.ExecuteAndWaitInterceptor"/>
133   <interceptor name="exception" class="com.opensymphony.xwork2.interceptor.ExceptionMappingInterceptor"/>
134   <interceptor name="fileUpload" class="org.apache.struts2.interceptor.FileUploadInterceptor"/>
135   <interceptor name="i18n" class="com.opensymphony.xwork2.interceptor.I18nInterceptor"/>
136   <interceptor name="logger" class="com.opensymphony.xwork2.interceptor.LoggingInterceptor"/>
137   <interceptor name="modelDriven" class="com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor"/>
138   <interceptor name="scopedModelDriven" class="com.opensymphony.xwork2.interceptor.ScopedModelDrivenInterceptor"/>
139   <interceptor name="params" class="com.opensymphony.xwork2.interceptor.ParametersInterceptor"/>
140   <interceptor name="actionMappingParams" class="org.apache.struts2.interceptor.ActionMappingParametersInterceptor"/>
141   <interceptor name="prepare" class="com.opensymphony.xwork2.interceptor.PrepareInterceptor"/>
142   <interceptor name="staticParams" class="com.opensymphony.xwork2.interceptor.StaticParametersInterceptor"/>
143   <interceptor name="scope" class="org.apache.struts2.interceptor.ScopeInterceptor"/>
144   <interceptor name="servletConfig" class="org.apache.struts2.interceptor.ServletConfigInterceptor"/>
145   <interceptor name="timer" class="com.opensymphony.xwork2.interceptor.TimerInterceptor"/>
146   <interceptor name="token" class="org.apache.struts2.interceptor.TokenInterceptor"/>
147   <interceptor name="tokenSession" class="org.apache.struts2.interceptor.TokenSessionStoreInterceptor"/>
148   <interceptor name="validation" class="org.apache.struts2.interceptor.validation.AnnotationValidationInterceptor"/>
149   <interceptor name="workflow" class="com.opensymphony.xwork2.interceptor.DefaultWorkflowInterceptor"/>
150   <interceptor name="store" class="org.apache.struts2.interceptor.MessageStoreInterceptor"/>
151   <interceptor name="checkbox" class="org.apache.struts2.interceptor.CheckboxInterceptor"/>
152   <interceptor name="profiling" class="org.apache.struts2.interceptor.ProfilingActivationInterceptor"/>
153   <interceptor name="roles" class="org.apache.struts2.interceptor.RolesInterceptor"/>
154   <interceptor name="annotationWorkflow" class="com.opensymphony.xwork2.interceptor.annotations.AnnotationWorkflowInterceptor"/>
155   <interceptor name="multiselect" class="org.apache.struts2.interceptor.MultiselectInterceptor"/>

```

Imagen relacionada del video de youtube

5.1. Relacion de conceptos



OBJETIVO GENERAL

Desarrollar la arquitectura de alojamiento Tomcat, Tomcat –apache, en los servicios Web.

OBJETIVOS ESPECÍFICOS

- ❑ Aplicar la arquitectura para la elaboración de los Struts.
- ❑ Describir las interfaces (marcos) aplicadas en Jsf.
- ❑ Diferenciar las estructuras para la dinámica del servicio Tomcat
- ❑ Aplicar los conceptos en el despliegue con la Web

5.2. Prueba Inicial

1. Programación y tecnología consiste en.
 - a) Traducir las reglas de un lenguaje dado a otro
 - b) Revisar y corregir errores de escritura de un lenguaje
 - c) Realizar las fases completas de análisis y síntesis a un lenguaje dado, para convertir un código para que sea entendido por la máquina.
 - d) Emplear componentes y contenedores con interfaz
 - e) Todas las anteriores.
2. La portabilidad de esta tecnología java se refiere a :
 - a) Disponer de hardware, software y un sistema operativo dependiente.
 - b) Disponer software y un sistema operativo independiente
 - c) Disponer de hardware y un sistema operativo independiente
 - d) Disponer de hardware, software y un sistema manejador de datos
 - e) Disponer de hardware, software y un sistema operativo independiente
3. Un desarrollador de software emplearía el Framework Struts para:
 - a) Facilitar la solución de aplicaciones compleja.
 - b) Facilitar la solución de aplicaciones compleja que no requieren explorador.
 - c) Facilitar la solución de aplicaciones que requieren Internet.
 - d) Facilitar la solución de aplicaciones que requieren Internet complejas o no.
 - e) Facilitar la solución de aplicaciones que requieren Internet complejas.
4. La programación se ha elaborado progresivamente de la siguiente manera
 1. Secuencialmente por líneas de código en un solo proceso poco flexible
 2. Proceso por bloques (módulos) que tienen mas flexibilidad en su uso
 3. Complementa los procesos con sus requerimientos para solucionar empleando clases.
 4. Maneja métodos pequeñas unidades lógicas de código se les llama objeto tiende a los aspectos
5. Todas las anteriores.

6. El aumento de la complejidad de los sistemas nos induce a :
- a) buscar nuevas técnicas de desarrollo incluida plataformas
 - b) buscar nuevas técnicas de desarrollo incluida servidores
 - c) buscar nuevas técnicas de desarrollo incluida las bases de datos
 - d) descartar técnicas de desarrollo incluidas sus plataformas
 - e) Generar nuevas técnicas de desarrollo y descartar plataformas

5.3. Struts

Definición de Struts

Struts se define como un Framework Java MVC2 para desarrollar aplicaciones Web basadas en las tecnologías JSP (http://java.ciberaula.com/articulo/tecnologia_java/, 2005) y Servlet sobre la plataforma J2EE.

Framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

La definición para Framework desde el punto de vista de (Soaagenda, 2007) “ambiente de trabajo y ejecución por java o .net, proporcionando librerías (paquetes) para la implementación o conexión empleando componentes “.

Arquitectura modelo-vista-controlador. Implementación en Struts. Acciones.
Struts (II): ActionForms. Taglibs propias.

Software necesario

j2eesdk-1_4_02_2005Q2-windows.exe. JDK 5.0 + J2EE 1.4 + Sun AppServer.

apache-tomcat-5.5.12.exe. Apache Tomcat 5.5.

Eclipse-wtp-all-in-one-0.7.1-win32.zip. Eclipse completo, con herramientas de desarrollo web.

apache-ant-1.6.5-bin.zip: ANT 1.6.5 para construir el proyecto sin necesidad de Eclipse.

Struts-1.2.8-bin.zip. Distribución de Apache Struts, que contiene binarios (en formato JAR), archivos auxiliares y WAR de documentación y ejemplos.

Datos del repositorio. Para conectarse con el repositorio CVS hace falta el protocolo, servidor, usuario, nombre del repositorio, puerto y módulo. Si no disponemos de un repositorio CVS, el tutorial de “Instalación y administración de un repositorio CVS en Windows” nos ayudará.

Definición Arquitectura

Entorno (framework) normalizado de presentación. Dicho así puede quedar un poco confuso, pero veamos como encajándolo dentro de la arquitectura de desarrollo MVC (Modelo-Vista-Controlador) seguro que descubrimos cuánto nos puede ayudar Struts.

Struts se encarga de normalizar (o al menos intentarlo) el desarrollo de la capa Vista dentro de la arquitectura MVC. Sin embargo, es necesario que para tal fin, también proporcione mecanismos para trabajar con la capa C. Pero nunca Struts se mezclará con la capa del Modelo. Esta característica principal de Struts permite separar la lógica de presentación de la lógica del negocio, con las ya consabidas ventajas que este tipo de desarrollo supone.





La primera vez que te enfrentas al framework de Struts (Gonzalez Almiron, 2003) para realizar aplicaciones Web, aparece todo un conjunto de nuevos conceptos ligados a este framework que te pueden hacer perder la visión de conjunto.

Actividad estudio de caso

- a) Se desarrollará una aplicación Web que consistirá en un sencillo sistema de envío y visualización de mensajes a través de la Web, cuyas páginas se mostrarán en un navegador Web. Cada mensaje estará formado por un destinatario, un remitente y un texto.
- b) La página de inicio muestra dos enlaces con las opciones del usuario, la de visualización de mensajes le llevará a otra página ("mostrar.htm"), donde se le solicitará el nombre del destinatario cuyos mensajes quiere visualizar. En caso de tener mensajes asociados se le enviará a una página donde se le mostrará una tabla con todos sus mensajes, indicando para cada uno de ellos el remitente y el contenido del mensaje. Por otro lado, la opción de envío de mensajes le llevará a una página en la que se le solicitarán los datos del mensaje que quiere enviar, devolviéndolo después a la página de inicio una vez que el mensaje ha sido almacenado.

DESARROLLO

- c) Los mensajes manejados por la aplicación serán almacenados en una tabla cuya estructura se indica.

	Nombre del campo	Tipo de datos
	remitente	cadena de texto
	destinatario	cadena de texto
	texto	cadena de texto
- d) El desarrollo de esta aplicación se realizará siguiendo el patrón MVC, donde tendremos un servlet llamado "controlador" en el que se centralizarán todas las peticiones procedentes desde el cliente.
- e) El Modelo estará implementado mediante una clase a la que llamaremos Operaciones que dispondrá de dos métodos: grabarMensaje(), encargado de almacenar en la base de datos los datos de un mensaje, y obtenerMensajes(), cuya función será la de recuperar la lista de mensajes asociados al destinatario que se proporciona como parámetro.
- f) Los mensajes serán manipulados mediante una clase JavaBean llamada Mensaje, que encapsulará los tres datos asociados a un determinado mensaje.

- g) En cuanto a la vistas, serán implementadas mediante cinco páginas, dos XHTML(inicio.htm y mostrar.htm), y tres JSP(envio.jsp, ver.jsp, nomensajes.jsp). Utilizando el parámetro "operación" insertando en la URL, la página inicio.htm, mostrar.htm y envio.jsp indicarán al servlet controlador el tipo de acción que se debe llevar a cabo en cada petición.
- h) Listados - A continuación se presenta el código de cada uno de los elementos de la aplicación.

Clase Mensaje

```
package javabeans;

public class Mensaje {
    private String remite;
    private String destino;
    private String texto;
    public Mensaje(){}
    //constructor que permite crear un objeto
    //Mensaje a partir de los datos del mismo
    public Mensaje(String remite, String destino, String texto){
        this.remite=remite;
        this.destino=destino;
        this.texto=texto;
    }
    public void setRemite(String remite){
        this.remite=remite;
    }
    public String getRemite(){
        return this.remite;
    }
    public void setDestino(String destino){
        this.destino=destino;
    }
    public String getDestino(){
        return this.destino;
    }
    public void setTexto(String texto){
        this.texto=texto;
    }
    public String getTexto(){
        return this.texto;
    }
}
```

```
}  
}
```

Clase controlador

```
package servlets;
```

```
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;  
import java.util.*;  
import javax beans.*;  
import modelo.*;  
public class Controlador extends HttpServlet {  
    public void service(HttpServletRequest request, HttpServletResponse response) throws  
        ServletException, IOException {  
        String op=request.getParameter("operacion");  
        //acceso a la página de envío de mensajes  
        if(op.equals("envio"))  
            response.sendRedirect("envio.jsp");  
        //grabación de un mensaje  
        if(op.equals("grabar")){  
            Mensaje men=(Mensaje)request.getAttribute("mensa");  
            Operaciones oper=new Operaciones();  
            oper.grabaMensaje(men);  
            response.sendRedirect("inicio.htm");  
        }  
        //acceso a la página de solicitud de mensajes  
        if(op.equals("muestra"))  
            response.sendRedirect("mostrar.htm");  
        //acceso a la lista de mensajes del usuario  
        if(op.equals("ver")){  
            Operaciones oper=new Operaciones();  
            ArrayList mensajes=oper.obtenerMensajes(request.getParameter("nombre"));  
            request.setAttribute("mensajes",mensajes);  
            RequestDispatcher rd=request.getRequestDispatcher("/ver.jsp");  
            rd.forward(request,response);  
        }  
    }  
}
```


Clase Operaciones

```
package modelo;
import java.sql.*;
import javax.swing.*;
import java.util.*;

public class Operaciones {
    //método común para la obtención
    //de conexiones
    public Connection getConnection(){
        Connection cn=null;
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            cn=DriverManager.getConnection("jdbc:odbc:mensajes");
        }
        catch(Exception e){e.printStackTrace();}
        return cn;
    }
    public ArrayList obtenerMensajes(String destino){
        Connection cn=null;
        ArrayList mensajes=null;
        Statement st;
        ResultSet rs;
        try{
            cn=getConnection();
            st=cn.createStatement();
            String tsq;
            tsq="select * from mensajes where destinatario='"+destino+"'";
            rs=st.executeQuery(tsq);
            mensajes=new ArrayList();
            //para cada mensaje encontrado crea un objeto
            //Mensaje y lo añade a la colección ArrayList
            while(rs.next()){
                Mensaje m=new
                Mensaje(rs.getString("remite"),rs.getString("destinatario"),rs.getString("texto"));
                mensajes.add(m);
            }
            cn.close();
        }
    }
}
```

```
catch(Exception e){e.printStackTrace();}
return(mensajes);
}
public void grabaMensaje(Mensaje m){
Connection cn;
Statement st;
ResultSet rs;
try{
cn=getConnection();
st=cn.createStatement();
String tsq;
//a partir de los datos del mensaje construye
//la cadena SQL para realizar su inserción
tsq="Insert into mensajes values('";
tsq+=m.getDestino()+"', '"+m.getRemite()+"', '"+m.getTexto()+"')";
st.execute(tsq);
cn.close();
}
catch(Exception e){e.printStackTrace();}
}
}
```

VISTAS DE LA APLICACIÓN

inicio.htm

```
<html>
<body>
<center>
<br/><br/>
<a href="controlador?operacion=envio">
Enviar mensaje
</a><br/><br/>
<a href="controlador?operacion=muestra">
Leer mensajes
</a>
</center>
</body>
</html>
```

mostrar.htm

```
<html>
<body>
<center>
<br/><br/>
<form action="controlador?operacion=ver" method="post">
Introduzca su nombre:<input type="text" name="nombre"><br><br>
<input type="submit">
</form>
</center>
</body>
</html>
```

envio.jsp

```
<html>
<head>
<title>envio</title>
</head>
<!--captura de datos e inserción en el Javabeans-->
<jsp:useBean id="mensa" scope="request" class="javabeans.Mensaje" />
<jsp:setProperty name="mensa" property="*" />
<%if(request.getParameter("texto")!=null){%>
<jsp:forward page="controlador?operacion=grabar"/>
<%}%>

<body>
<center>
<h1>Generación de mensajes</h1>
<form method="post">
<br/><br/>
<b>Datos del mensaje:</b><br/><br/>
Introduzca destinatario: <input type="text" name="destino"><br/>
<br/>
Introduzca remitente : <input type="text" name="remite"><br/>
<br/>
Introduzca texto : <br/>
<textarea name="texto">
</textarea>
```

```
<hr/><br/>
<input type="submit" name="Submit" value="Enviar">
<input type="reset" value="Reset">
</form>
</center>
</body>
</html>
```

ver.jsp

```
<%@ page import="javabeans.*,java.util.*"%>
<html>
<head>
<title>ver</title>
</head>
<body>
<center>
<%String nombre=request.getParameter("nombre");%>
<h1>
Mensajes para <%=nombre%>
</h1>
<table border=1>
<tr><th>Remitente</th><th>Mensaje</th></tr>
<%boolean men=false;
ArrayList mensajes=(ArrayList)request.getAttribute("mensajes");
if(mensajes!=null)
//si existen mensajes para ese destinatario
//se generará una tabla con los mismos
for(int i=0;i<mensajes.size();i++){
Mensaje m=(Mensaje)mensajes.get(i);
if((m.getDestino()).equalsIgnoreCase(nombre)){
men=true;%>
<tr><td><%=m.getRemite()%></td><td><%=m.getTexto()%></td></tr>
<%}
}
if(!men){%>
<!--si no hay mensajes se envía al usuario
a la página nomensajes.jsp-->
<jsp:forward page="nomensajes.jsp"/>
<%}%>
```

```
</table>
<br/><br/>
<a href="inicio.htm">Inicio</a>
</center>
</body>
</html>
```

nomensajes.jsp

```
<html>
<head>
<title>nomensajes</title>
</head>
<body>
<center>
<h2>
Lo siento, <%=request.getParameter("nombre")%> no tiene mensajes
</h2>
<br/><br/><br/><br/>
<a href="inicio.htm">Inicio</a>
</center>
</body>
</html>
```

Por último pongo el web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app          version="2.4"          xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee    http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">
<servlet>
<servlet-name>Controlador</servlet-name>
<servlet-class>servlets.Controlador</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Controlador</servlet-name>
<url-pattern>/controlador</url-pattern>
</servlet-mapping>
<session-config>
```

```
<session-timeout>
30
</session-timeout>
</session-config>
<welcome-file-list>
<welcome-file>
index.jsp
</welcome-file>
</welcome-file-list>
</web-app>
```

BIBLIOGRAFÍA

1. STRUTS, Antonio J. Martín Sierra, Editorial Alfaomega RA-MA, Año 2008.
2. Manual de Referencia JSP, Phil Hanna, Mc Graw Hill, Osborne Media.

Escrito por guillermoarreola el 27/07/2009 23:10 | Comentarios

(3)Resumen

Si hemos conseguido desplegar la aplicación, podremos comenzar a utilizar nuestra aplicación del carrito de la compra. Algunas pantallas son: capturas 1...4

Espero que este tutorial os sirva de ayuda, ya que su intención no es ser un manual de Struts, sino iniciar al programador en este entorno de presentación, del que seguro obtendrá muchas ventajas. Si veo que tiene mucha aceptación, os mostraré como añadir EJBs de forma inmediata.

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de Struts (10 7.7%)

Describir conceptos obtenidos en Struts.

Se validan los conceptos de los Struts.

Seleccione la respuesta correcta a las siguientes preguntas según su conocimiento previo:

1. Desde la visión del programador que es Struts?
2. Describa cuales son las funciones básicas?
3. Un Struts se considera un patrón (framework) si o no y porque?

Nombre del ejercicio de aprendizaje: STRUTS	Datos del autor del taller: César Augusto Jaramillo Henao
<p>Escriba o plantee el caso, problema o pregunta:</p> <p>Struts es una herramienta de desarrollo de aplicaciones Web bajo el modelo MVC bajo la plataforma Java EE.</p> <p>Represente Gráficamente como realizar un Struts dentro de JSP.</p>	
<p>Solución del Taller:</p> <p>Esta plataforma permite el uso de recursos framework para el desarrollo de aplicaciones web.</p> <pre><%@ taglib uri="/WEB-INF/struts-template.tld" prefix="template"%></pre> <pre><template:insert template="/portal/templates/template.jsp"> <template:put name="title" content="Templates" direct="true"/> <template:put name="keywords" content="aplicacion portal struts" direct="true"/> <template:put name="description" content="Utilizacion de Templates en Struts" direct="true"/> <template:put name="contenido" content="/portal/info/introduccion.html"/> <template:put name="pie" content="/portal/info/footer.html"/> <template:put name="cabeza" content="/portal/info/header.jsp"/> <template:put name="costado_i" content="/portal/info/sidebar_l.jsp"/> <template:put name="costado_d" content="/portal/info/sidebar_r.jsp"/> </template:insert></pre> <p>La primera declaración de este JSP indica que será utilizada la librería struts-template.tld y maneja una palabra particular que es el template.</p> <p>La primera etiqueta hace referencia a un archivo jsp indicando que es un complemento del lenguaje.</p>	

Pista de aprendizaje:

Tener en cuenta: struts es un complemento de SSP.

Tenga presente: un framework es una herramienta de gran poder y complementario para un desarrollo rápido y de poderoso.

Traer a la memoria: que aunque sea un archivo JSP tiene complementos que lo hacen más estructural y completo.

5.4. JSF

La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java”.

(Sun Microsystems) –Estándar de Java (JSR-127)

JSF: arquitectura y componentes GUI

Es un framework agrupado en estándar java J2EE se engarga de realizar peticiones al servidor por medio de componentes para realizar aplicaciones bajo el MVC (Modelo Vista Controlador) (González Almirón, 2003) .

Definición JSF

Este patrón nos permite/obliga a separar la lógica de control (sabe que cosas hay que hacer pero no como), la lógica de negocio (sabe como se hacen las cosas) y la lógica de presentación (sabe como interactuar con el usuario) empleando jsp de respaldo en la visualización.

Una aplicación JSF se ejecutan en un contenedor web estándar, por ejemplo, Tomcat.

Facilita el uso de componentes para su presentacion en aplicaciones empresariales.

A continuación presento algunos de los puntos por los que JSF me parece una tecnología muy interesante (incluso mas que Struts ;)

- ❏ JSF trata la vista (el interfaz de usuario) de una forma algo diferente a lo que estamos acostumbrados en aplicaciones web. Sería más similar al estilo de Swing, Visual Basic o Delphi, donde la programación del interfaz se hace a través de componentes y basada en eventos (se pulsa un botón, cambia el valor de un campo.).
- ❏ JSF es muy flexible. Por ejemplo nos permite crear nuestros propios componentes, o crear nuestros propios “render” para pintar los componentes según nos convenga.
- ❏ Es más sencillo.

JSF comparado con Struts según

(www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350 de Google, 2004) no puede competir en madurez con Struts (en este punto Struts es claro ganador), pero si puede ser una opción muy recomendable para nuevos desarrollos, sobre todo si todavía no tenemos experiencia con Struts.

Actividad estudio de caso

Vamos a ver un ejemplo donde tendremos un formulario para registrar personas y meterlos en un array. Luego tendremos otro botón que nos permitirá sacar por pantalla los datos introducidos.

Lo primero que haremos será crear la clase JavaBean referente al formulario del cliente. El código es el siguiente:

```
package javabeans;
import java.util.*;
import javax.faces.context.*;
import javax.faces.event.*;
import javax.faces.component.*;
import javax.servlet.http.*;
```

```
public class PersonaBean {
    private String nombre;
    private long telefono;
    private int edad;

    public PersonaBean() {
    }

    public String getNombre(){
        return this.nombre;
    }
}
```

```
public void setNombre(String nombre){
    this.nombre=nombre;
}

public long getTelefono(){
    return this.telefono;
}

public void setTelefono(long telefono){
    this.telefono=telefono;
}

public int getEdad(){
    return this.edad;
}

public void setEdad(int edad){
    this.edad=edad;
}

public String doGuardar(){
    FacesContext context = FacesContext.getCurrentInstance();
    HttpSession sesion = (HttpSession)context.getExternalContext().getSession(true);
    ArrayList listapersonas = (ArrayList)sesion.getAttribute("listapersonas");

    //comprueba si ya existe la coleccion de personas en la sesion y si no es así la crea
    if(listapersonas==null){
        listapersonas = new ArrayList();
        sesion.setAttribute("listapersonas", listapersonas);
    }
    listapersonas.add(this);
    return null;
}
```

Como vemos es una clase JavaBean normal con sus métodos SET y GET para las propiedades y un método doGuardar que se ejecutará cuando se pulse el botón de Guardar del formulario.

En este método doGuardar, lo que hacemos es obtener del ámbito sesión la lista de personas. Si esta lista es nula, entonces la creamos nueva, si no es nula, lo que hacemos es añadir a la lista el nuevo objeto persona.

El archivo faces-config.xml será el siguiente que os ponemos a continuación:

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ===== FULL CONFIGURATION FILE ===== -->

<faces-config version="1.2"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">
  <managed-bean>
    <managed-bean-name>PersonaBean</managed-bean-name>
    <managed-bean-class>javabeans.PersonaBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>
  <navigation-rule>
    <from-view-id>/grabadatos.jsp</from-view-id>
    <navigation-case>
      <from-outcome>ver</from-outcome>
      <to-view-id>/ver.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>
</faces-config>
```

En este archivo, lo que hacemos es declarar el JavaBean utilizado y las reglas de navegación. En estas reglas, hemos indicado que cuando se dé de alta un usuario, la aplicación vuelva al formulario y cuando se pulse el botón de ver, este llamará a una url ver, que nos llevará al archivo ver.jsp.

Pasemos ahora a la vista del formulario de alta de personas.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Insertar personas</title>
</head>
<body>
  <center>
    <h1>Formulario de datos</h1>
  </center>
  <f:view>
    <h:form>
      <table width="50%" align="center" border="0">
        <tr>
          <td>Nombre:</td>
          <td>
            <h:inputText value="#{PersonaBean.nombre}"/>
          </td>
        </tr>
        <tr>
          <td>Telefono:</td>
          <td>
            <h:inputText value="#{PersonaBean.telefono}"/>
          </td>
        </tr>
        <tr>
          <td>Edad:</td>
          <td>
            <h:inputText value="#{PersonaBean.edad}"/>
          </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <h:commandButton value="Guardar" action="#{PersonaBean.doGuardar}"/>
          </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <h:commandButton id="vertodos" rendered="false" value="Ver Todos"
action="ver"/>
          </td>
        </tr>
      </table>
    </h:form>
```

```
</f:view>
</body>
</html>
```

De este código debemos destacar la utilización de las librería core y html comentadas anteriormente, y como se le asigna una propiedad del JavaBean a los distintos campos. Por ejemplo al campo edad, el código es:

```
<h:inputText value="#{PersonaBean.edad}"/>
```

InputText es el control referente al tipo text de html y le asociamos la propiedad edad del javabean.

En los botones le indicamos las acciones que deben de realizar, así cuando se pulsa en guardar ejecutará el método doGuardar del javabean. Mientras que cuando se pulse al botón de ver, lo que hace es mandarlo a la url ver, que corresponde a la vista ver.jsp, tal y como le hemos indicado en el archivo faces-config.xml.

Por último, la vista ver.jsp, lo que hará será recorrer la lista de clientes y mostrarla en una tabla. Para ello utilizaremos la librería core de JSTL (librería utilizada para crear páginas en JSP para no utilizar código java en los archivos)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ver registros</title>
</head>
<body>
    <center>
        <h1>Listado de personas</h1>
        <table border="1" align="center">
            <tr>
                <th>Nombre</th>
                <th>Teléfono</th>
                <th>Edad</th>
            </tr>
```

```
<c:forEach var="per" items="${listapersonas}">
  <tr>
    <td><c:out value="${per.nombre}"/></td>
    <td><c:out value="${per.telefono}"/></td>
    <td><c:out value="${per.edad}"/></td>
  </tr>
</c:forEach>
</table>
<br/><br/>
<a href="grabadatos.jsp">Volver</a>
</center>
</body>
</html>
```

Se puede ver como el servlet de JSF (www.desarrolloweb.com/articulos/2380.php, 2003) recogerá todas las peticiones que acaben en “.jsf”.

Talleres experimentales

```
import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

//Con esto obtenemos el contexto
FacesContext facesContext= FacesContext.getCurrentInstance();

//Con esto obtenemos la request
HttpServletRequest request =
(FacesContext) facesContext;
request.getExternalContext().getRequest();

//Con esto obtenemos la lista de nombres de los parámetros enviados
// en la request
Enumeration params = request.getParameterNames();

//Acceso a la sesión http
HttpSession sessio = request.getSession();
```

Método de componentes

JSF está orientado a prestar el soporte necesario para la construcción de las capas controlador y vista, dejando para las clases normales la capa del modelo.

Toda aplicación JSF tendrá un archivo de configuración Faces-config.xml, donde definiremos varios elementos claves para el funcionamiento de la aplicación. Es aquí donde definiremos las reglas de navegación por la web, según el resultado que obtengamos en cada operación.

Otro de los componentes que utilizaremos en las aplicaciones serán los Beans Gestionados, que no son más que simples JavaBeans, que además de ser utilizados para encapsular los datos de la capa cliente, disponen de métodos que responden a los eventos producidos en la capa cliente.

Todo Beans utilizado en la aplicación deberá de ser registrado en el archivo faces-config.xml. La apariencia que tendrás será la siguiente:

```
<managed-bean>
  <managed-bean-name>credencialesBean</managed-bean-name>
  <managed-bean-class>javabeans.CredencialesBean</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Explicuemos a continuación cada campo lo que significa:

managed-bean. Es el elemento principal en el que se incluyen los datos de registro del bean. Será necesario añadir un bloque `<managed-bean>` por cada bean que se quiera sea gestionado por el framework.

managed-bean-name. Es el nombre que permite referirse a la instancia del bean. Este nombre es utilizado mediante el lenguaje EL desde los componentes de la interfaz para acceder a las propiedades y métodos del objeto.

managed-bean-class. Nombre cualificado de la clase a la que pertenece el bean.

managed-bean.scope. Ámbito en el que será mantenida la instancia, siendo sus posibles valores: request, session y application.

Por último tendremos las vistas, que será la parte visible a los usuarios y que podrá estar creadas utilizando JSP. Si es así, deberemos de utilizar una serie de librerías de acciones que pasamos a detallar a continuación:

Librería html: proporciona los controles básicos html y una serie de componentes gráficos complejos. Todos los controles permiten integrar el modelo JavaBean, vinculando de manera

automática las propiedades de un objeto a las de un control. Para su uso hay que poner la siguiente línea al principio del archivo JSP:

```
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
```

Librería core: Incluyen elementos básicos para la generación de una vista JSF. También incluyen elementos especiales conocidos como validadores y conversores que permiten realizar validaciones y conversiones de datos. Para poder utilizar esta librería hay que incluir la siguiente línea:

```
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/core" %>
```

El entorno: para la aplicación es :

Hardware: Portátil Ahtex Signal X-9500M (Centrino 1.6 GHz, 1024 MB RAM, 60 GB HD).

Sistema Operativo: GNU / Linux, Debian Sid (unstable), Kernel 2.6.12, KDE 3.4

Máquina Virtual Java: JDK 1.5.0_03 de Sun Microsystems

Eclipse 3.1

Tomcat 5.5.9

MyFaces 1.0.9

Instalación de MyFaces

Una de las ventajas de que JSF sea una especificación es que podemos encontrar implementaciones de distintos fabricantes. Esto nos permite no atarnos con un proveedor y poder seleccionar el que más nos interesa según: número de componentes que nos proporciona, rendimiento, soporte, precio, política,.

En nuestro caso vamos a usar el proyecto de Apache: MyFaces. Esta es una implementación de JSF de Software Libre que, además de cumplir con el estándar, también nos proporciona algunos componentes adicionales.

Descargamos myfaces-*.tgz de <http://myfaces.apache.org/binary.cgi>

Una vez descargado lo descomprimos en un directorio.

Creamos un proyecto web en nuestro Eclipse: File --> New --> Project... --> Dynamic Web Project

Copiamos las librerías necesarias para usar JSF

```
$ cp <MYFACES_HOME>/lib/* <MYPROJECT_HOME>/WebContent/WEB-INF/lib
```

donde <MYFACES_HOME> es el directorio donde hemos descomprimido myfaces-*.tgz y <MYPROJECT_HOME> es el directorio de nuestro proyecto de Eclipse.

EJERCICIO DE AUTOEVAÑUACIÓN

La evaluación de JSF (11)

Describir conceptos obtenidos en JSF.

Se validan los conceptos de JSF.

Describe la respuesta correcta a las siguientes preguntas según su conocimiento:

1. Desde la visión del programador que es JSF?
2. Describa algunos puntos llamativos de la tecnología JSF
3. Es posible comparar un JSF con struts si o no y por que?

Nombre del ejercicio de aprendizaje: JSF	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Un FrameWork es utilizado para herramientas de J2EE, los frameWork son complementos para agilizar las tareas cotidianas y darle una presentación más llamativa y atractiva para el usuario final. Cree las estructuras básicas para crear un JSF.	
Solución del Taller: <pre>import javax.faces.bean.ManagedBean; import javax.faces.bean.RequestScoped; @ManagedBean(name="Beans") @RequestScoped public class Beans { private String mensaje="hola mundo"; public Beans() { } public String getMensaje() { return mensaje; } public void setMensaje(String mensaje) { his.mensaje = mensaje; }</pre>	

```
}  
este framework contiene un conjunto de APIs para representar componentes de una interfaz
```

Pista de aprendizaje:

Tener en cuenta: es JSF son aplicativos o metodologías complementarias dependientes de la tecnología JSP

Tenga presente: que se conserva la codificación de java

Traer a la memoria: que no siempre es necesario trabajar con estos frames pero los procesos se vuelven más extensos.

5.5. Tomcat

Integra dos servidores que soportan tecnologías de servidores y contenedores WEB, Apache es el servidor WEB más usado a nivel mundial en Internet dada su "simplicidad" y su alta eficiencia. Tomcat (Tomcat, 2006) es un contenedor WEB para la implementación de sitios dinámicos mediante JSPs

Definición

Es el servidor web y de aplicaciones del proyecto Yakarta, decimos que es servidor web ya que gestiona solicitudes y respuestas Http (incluye el servidor Apache) gracias a sus conectores Http; además es servidor de aplicaciones o contenedor de Servlets/JSP (Catalina).segun (Lago, 2007)

La Arquitectura de Tomcat

Lo primero es la descarga en El sitio de la Apache Software Foundation. El ejecutable incluye un wizard que facilita bastante la instalación:

<http://www.proactiva-calidad.com/java/herramientas/tomcat/index.html>

El archivo .class del servlet se colocan en un subdirectorio de WEB-INF que se llama classes/docen_servlet01, ya que éste es el paquete de la clase. Con lo cual tendremos la clase en c:/web/public_html/WEB-INF/classes/docen_servlet01.

Aplicando todo lo indicado escribimos en server.xml:

Los contenedores

Contenedor web:

El contenedor necesita conectores que sirven de intermediarios para comunicarse con elementos externos. Los conectores capacitan al AS para acceder a sistemas empresariales (backends

El Java Message Service ofrece conectividad con sistemas de mensajería como MQSeries.

El API JDBC da la capacidad de gestionar bases de datos internas al AS, pero además permite ofrecer servicios como un pool de conexiones.

Es necesaria una gestión de hilos, ya que será necesario controlar la situación en la que tenemos una instancia de un componente (por ejemplo, un servlet) que da respuesta a varias peticiones, donde cada petición se resuelve en un hilo.

Conceptos básicos Método de proyectos

Directorio conf

Scripts de inicialización

Variables de ambiente

Actividad estudio de caso

Pero conviene empezar por el principio, es decir, el lenguaje básico de interconexión: el protocolo HTTP. Es un protocolo de aplicación, generalmente implementado sobre TCP/IP. Es un protocolo sin estado basado en solicitudes (request) y respuestas (response), que usa por defecto el puerto 8080:

"Basado en peticiones y respuestas": significa que el cliente (por ejemplo un navegador) inicia siempre la conexión (por ejemplo, para pedir una página). No hay posibilidad de que el servidor realice una llamada de respuesta al cliente (retrollamada). El servidor ofrece la respuesta (la página) y cierra la conexión. En la siguiente petición del cliente se abre una conexión y el ciclo vuelve a empezar: el servidor devuelve el recurso y cierra conexión.

"Sin estado": el servidor cierra la conexión una vez realizada la respuesta. No se mantienen los datos asociados a la conexión. Más adelante veremos que hay una forma de persistencia de datos asociada a la "sesión".

¿Qué ocurre cuando un navegador invoca una aplicación? El cliente (el navegador) no invoca directamente el contenedor de aplicaciones, sino que llama al servidor web por medio de HTTP. El

servidor web se interpone en la solicitud o invocación; siendo el servidor web el responsable de trasladar la solicitud al contenedor de aplicaciones.

Aspectos a considerar de forma síncrona:

El cliente (normalmente por medio de un navegador, aunque podría ser una aplicación Swing) solicita un recurso por medio de HTTP. Para localizar el recurso al cliente especifica una URL (Uniform Resource Locator), como por ejemplo `http://www.host.es/aplicacion/recurso.html`. El URI (Uniform Resource Identifier) es el URL excluyendo protocolo y host. Existen diversos métodos de invocación, aunque los más comunes son POST y GET. Los veremos más adelante.

Sobre una misma máquina podemos tener diversas instancias de un AS (Application Server), procurando que trabajen sobre puertos diferentes, para que no se produzcan colisiones (por defecto HTTP trabaja con 8080).

Un servicio crucial es la capacidad de recibir peticiones HTTP, para lo cual tenemos un HTTP Listener (aunque puede tener listeners para otros protocolos como IIOP).

La solicitud llega al servidor de páginas web, que tiene que descifrar si el recurso solicitado es un recurso estático o una aplicación. Si es una aplicación delega la solicitud en el contenedor web (contenedor Servlet/JSP). El contenedor web gestiona la localización y ejecución de Servlets y JSP, que no son más que pequeños programas. El contenedor web o contenedor Servlet/JSP recibe la solicitud. Su máquina Java (JVM) invoca al objeto Servlet/JSP, por tanto nos encontramos ante un tipo de aplicaciones que se ejecutan en el servidor, no en el cliente. No conviene olvidar que un Servlet o un JSP no es más que una clase Java. Lo más interesante en este sentido es que: La JVM (generalmente) no crea una instancia de la clase por cada solicitud, sino que con una única instancia de un Servlet/JSP se da servicio a múltiples solicitudes HTTP. Esto hace que el consumo de recursos sea pequeño en comparación con otras opciones, como el uso de CGIs, en donde cada solicitud se resuelve en un proceso.

Para cada solicitud se genera un hilo (thread) para resolverla (pero con una única instancia de la clase, como hemos dicho).

Un Application Server tendrá un servidor de administración (y normalmente un manager de las aplicaciones).

EJERCICIO DE AUTOEVALUACIÓN

La evaluación de Tomcat (12 valor 7.7 %)

Describir conceptos obtenidos en Tomcat

Se validan los conceptos de Tomcat.

Seleccione la respuesta correcta a las siguientes preguntas según su conocimiento:

1. Desde la visión del programador como define Tomcat?
2. Describa alguna de las funciones de Tomcat?
3. Tomcat permite integrar servidor y contenedor en una aplicación si o no y porque?

Nombre del ejercicio de aprendizaje: Tomcat	Datos del autor del taller: César Augusto Jaramillo Henao
---	---

Escriba o plantee el caso, problema o pregunta:

Dentro los muchos sistemas intérpretes de sitios web encontramos algunos particulares como el Tomcat, este servicios permite ejecutar Apache, que es un intérprete para otros ambientes. El tomcat integra una serie de procesos complementarios, tiene su propia carpeta de trabajo y se puede integrar a su vez con otras plataformas, ideal para trabajar JSP.

Solución del taller:

Se crea el siguiente formulario en jsp

Ingreso de Carrera

[Retornar](#)

Código

Nombre

Este archivo tiene una codificación netamente HTML, en el formulario tendría este código

```
<form id="form1" name="form1" method="post" action="Procesos.jsp?opc=1">
```

Y se crea el archivo correspondiente de la siguiente manera

```
<%@ page language = "java" import = "java.sql.*" %>
```

```
<%@ include file = "/Conexion.jsp" %>
<%
    Statement con = null;
    int resultado = 0;

    con = cnn.createStatement ();

    int opc = Integer.parseInt (request.getParameter("opc"));

    switch (opc){

        case 1:
            resultado = con.executeUpdate("call consultarCarrera
('"+request.getParameter("codigoCarrera")+"'");
            if (resultado == 0){
                resultado = con.executeUpdate("call insertarCarrera
('"+request.getParameter("codigoCarrera")+"' , '"+request.getParameter("nombreCarrera")+"' ,
'Activo')");
            }
            <script type = "text/javascript">
                alert ("Registro Almacenado");
                history.back ();
            </script>
        }
        else {

            <script type = "text/javascript">
                alert ("Registro ya Existe");
                history.back ();
            </script>
        }
        break;
    }
}%>
```

Este proceso conecta, valida y ejecuta la sentencia correspondiente para almacenar la información.

Pista de aprendizaje:

Tener en cuenta: JSF son aplicativos o metodologías complementarias dependientes de la tecnología JSP.

Tenga presente: que se conserva la codificación de java.

Traiga a la memoria: que no siempre es requerido trabajar con estos frames pero los procesos se vuelven más extensos.

5.6. Integrando Tomcat y Apache

Estructura Apache+Tomcat se utiliza para aplicaciones Web. En dicha estructura apache es un "escuchador" que cuando se le solicita una página que no entiende por si mismo la redirige a otros sistemas que si lo entienden.

Definición t4

Talleres experimentales

Comprobando

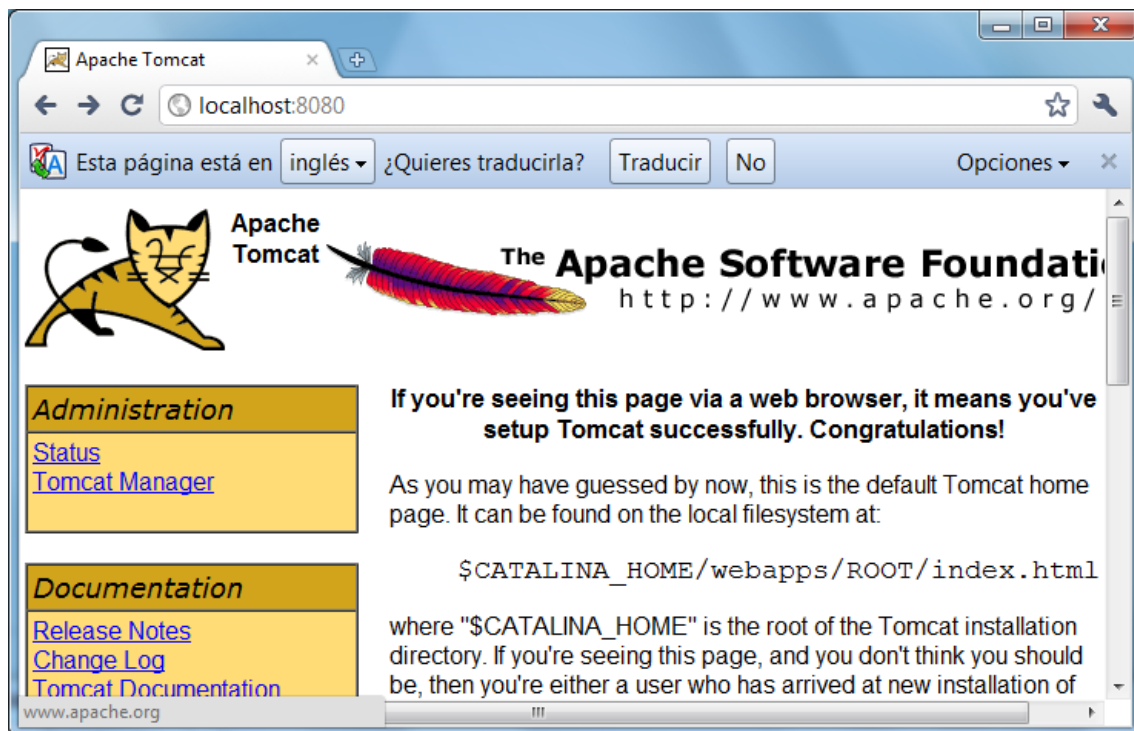


Gráfico Numero8 Se comprueba la existencia de Apache tomcat

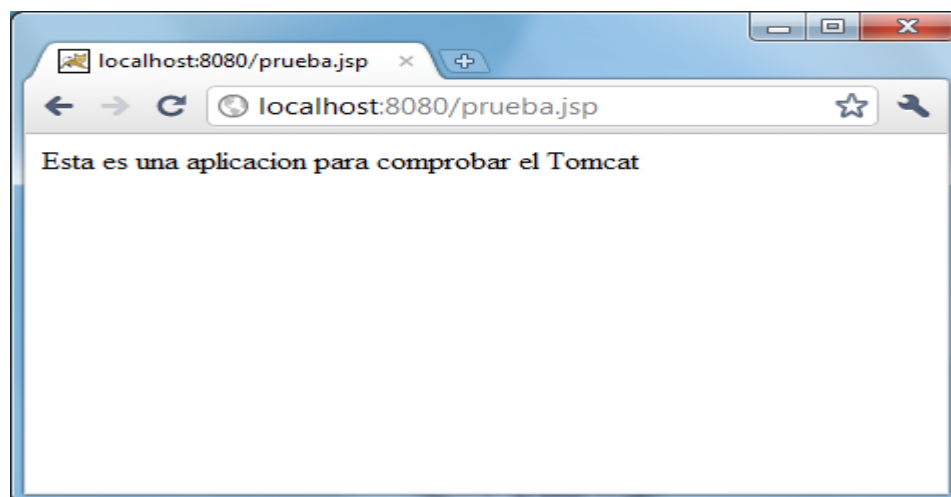


Gráfico Numero9 Se comprueba la existencia de Apache tomcat

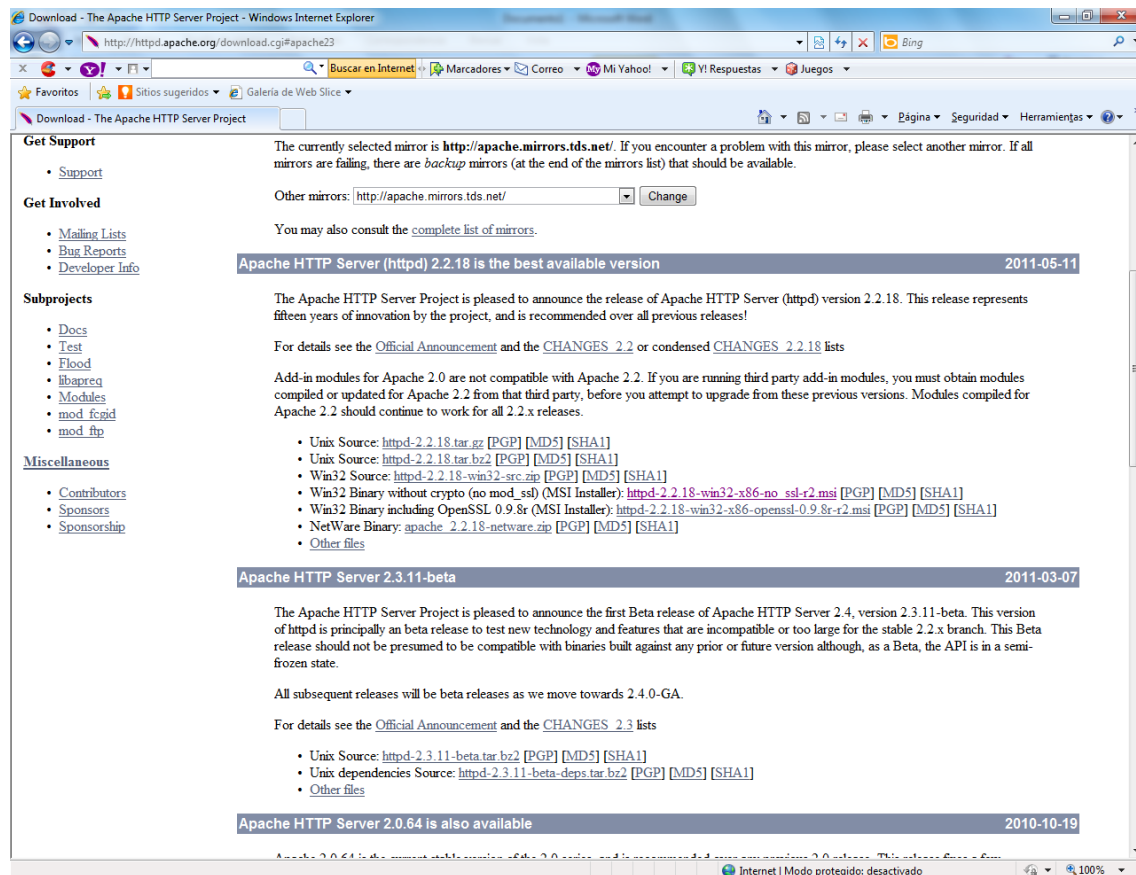


Gráfico Numero10 Se comprueba la existencia de Apache tomcat

Tomado de <http://httpd.apache.org/download.cgi#apache23>

Para Apache

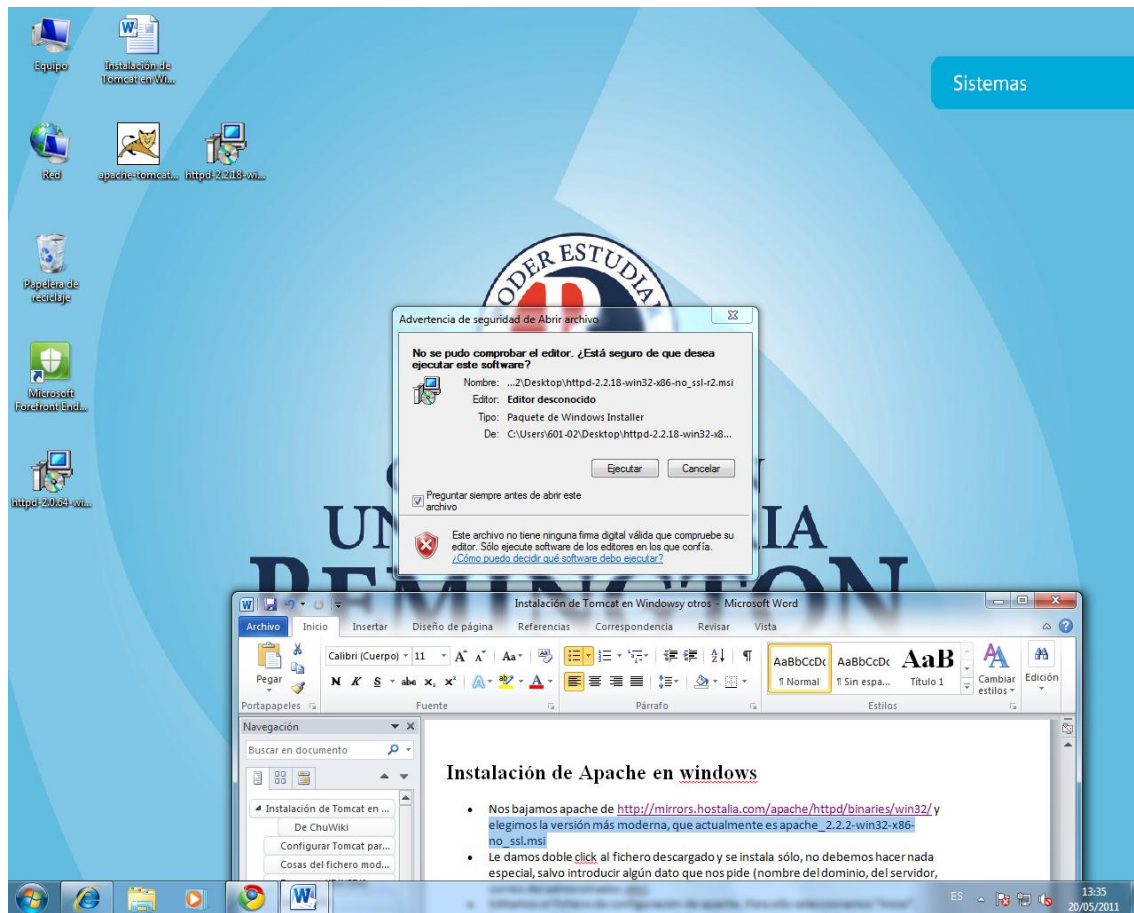


Gráfico Numero11 Se comprueba la existencia de Apache tomcat para Windows.

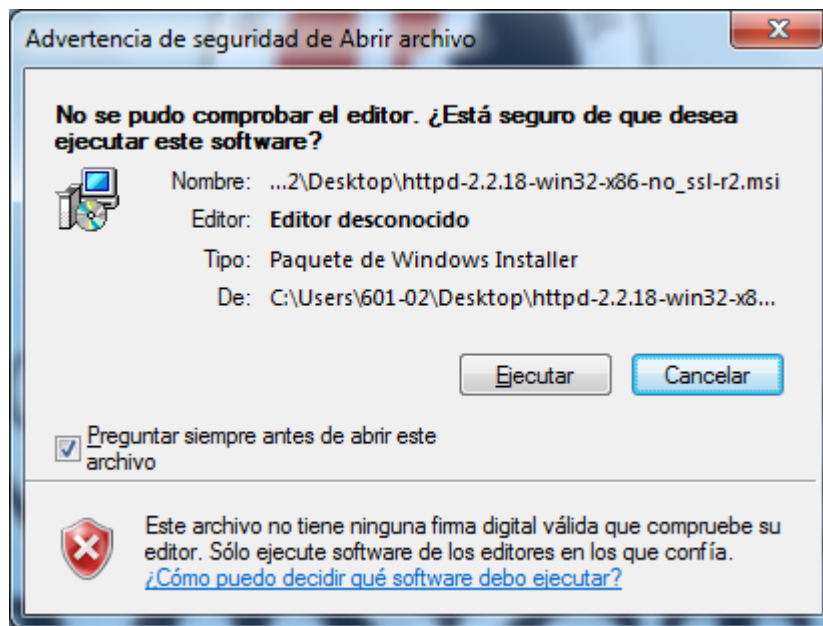


Gráfico Numero12 Se Inicia el proceso de instalación Apache tomcat

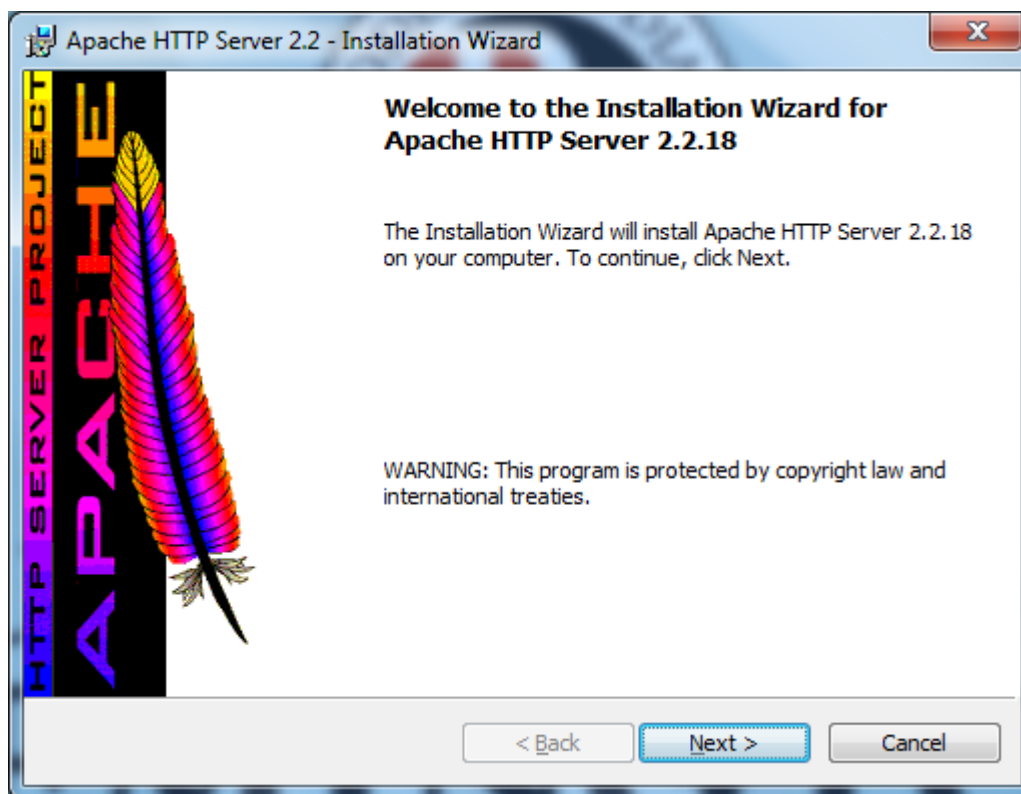


Gráfico Numero13 Se realizan cada uno de los pasos de Apache tomcat

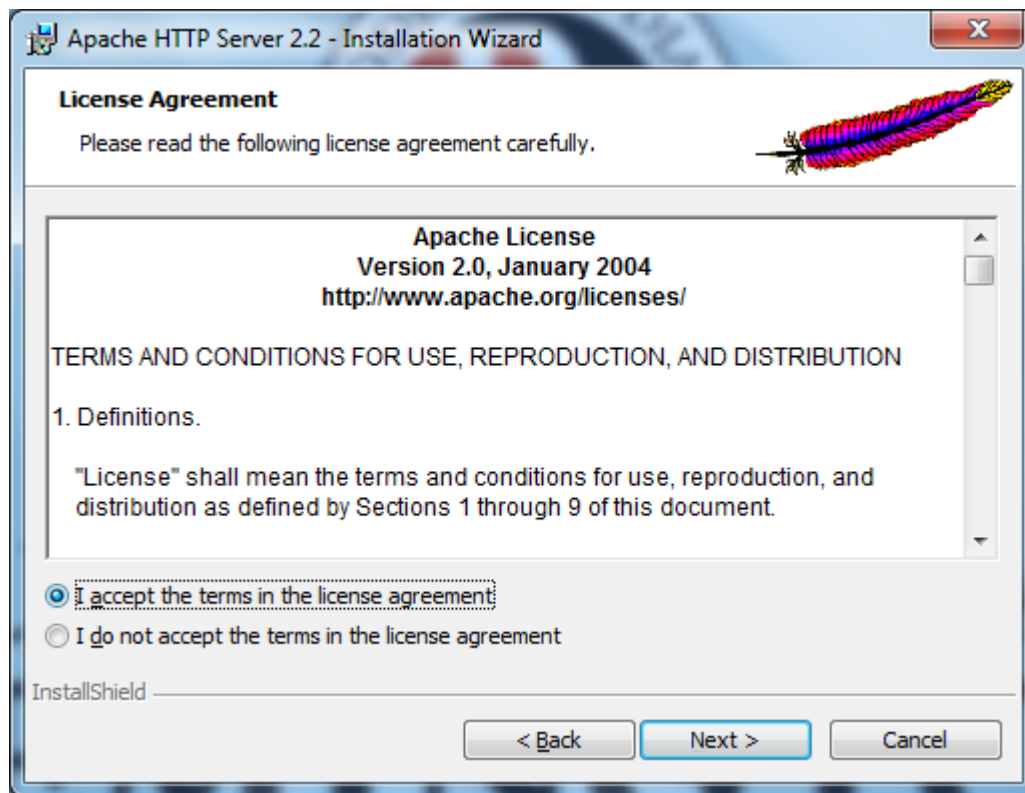


Gráfico Numero14 Se comprueba la existencia de Apache tomcat

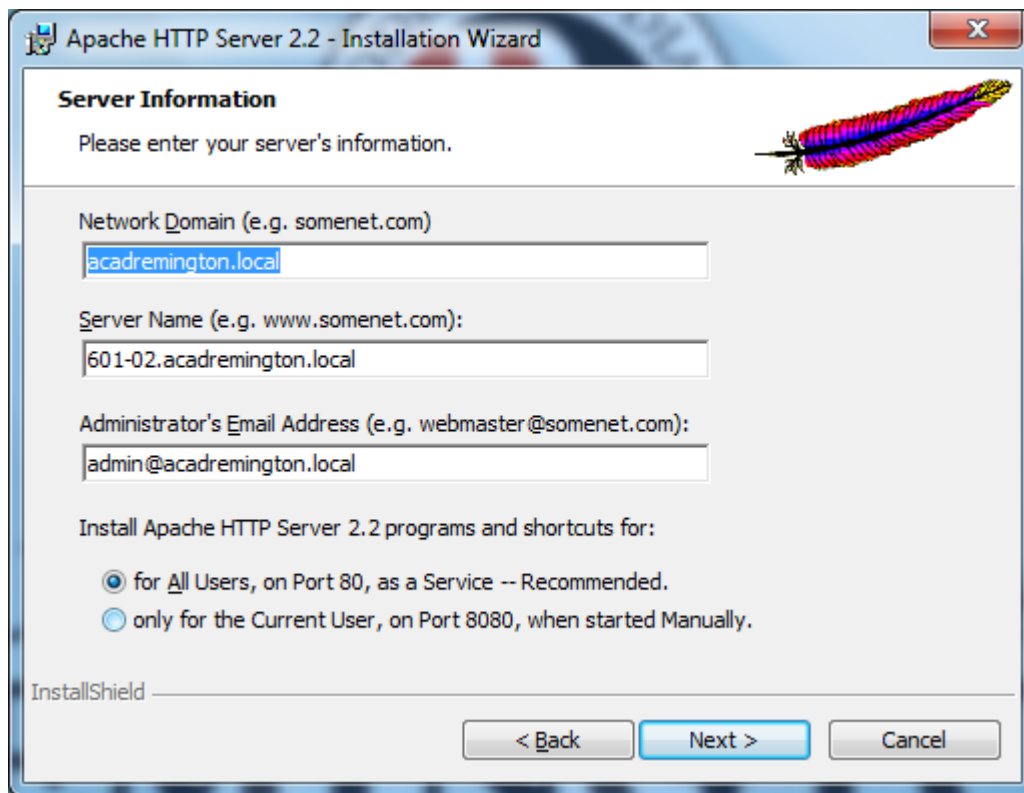


Gráfico Numero15 Se comprueba la autorización correspondiente Apache tomcat

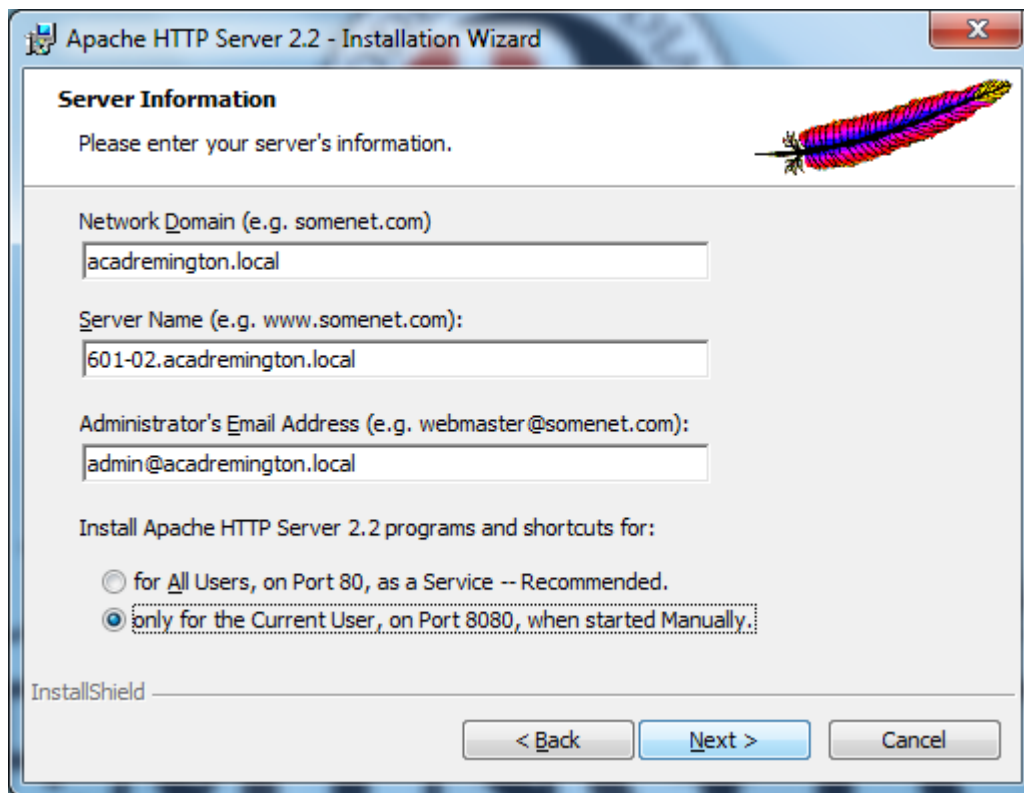


Gráfico Numero16 Se comprueba la existencia de Apache tomcat

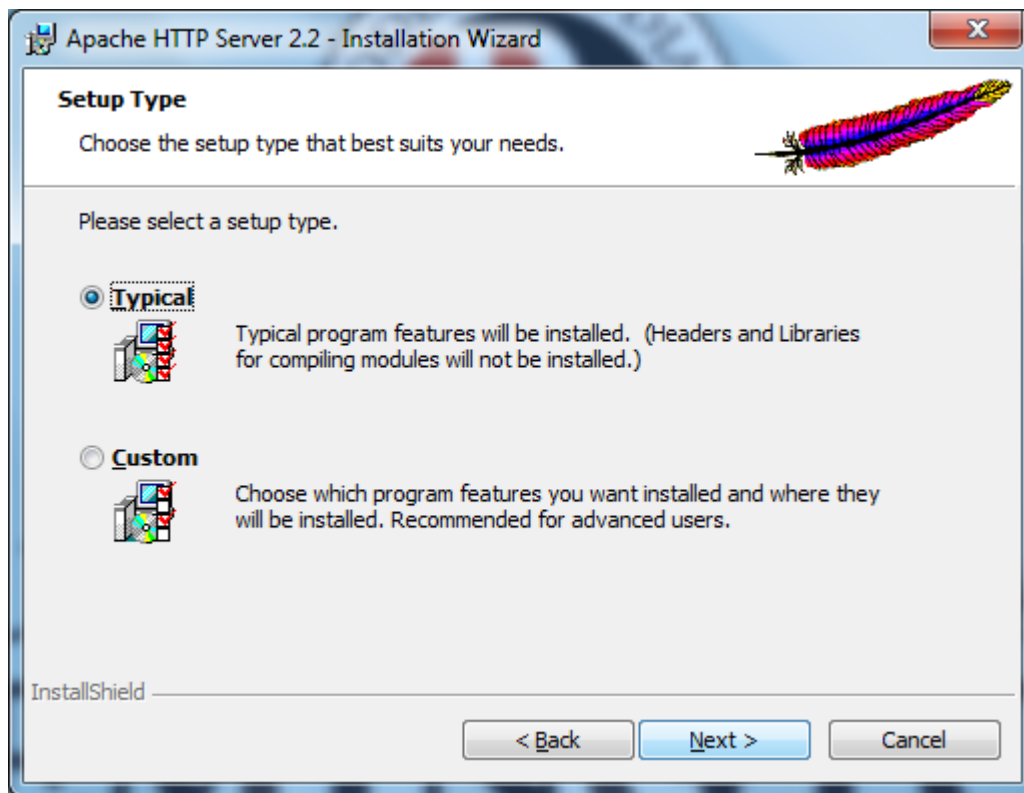


Gráfico Numero17 Se selecciona el tipo de instalacion de Apache tomcat

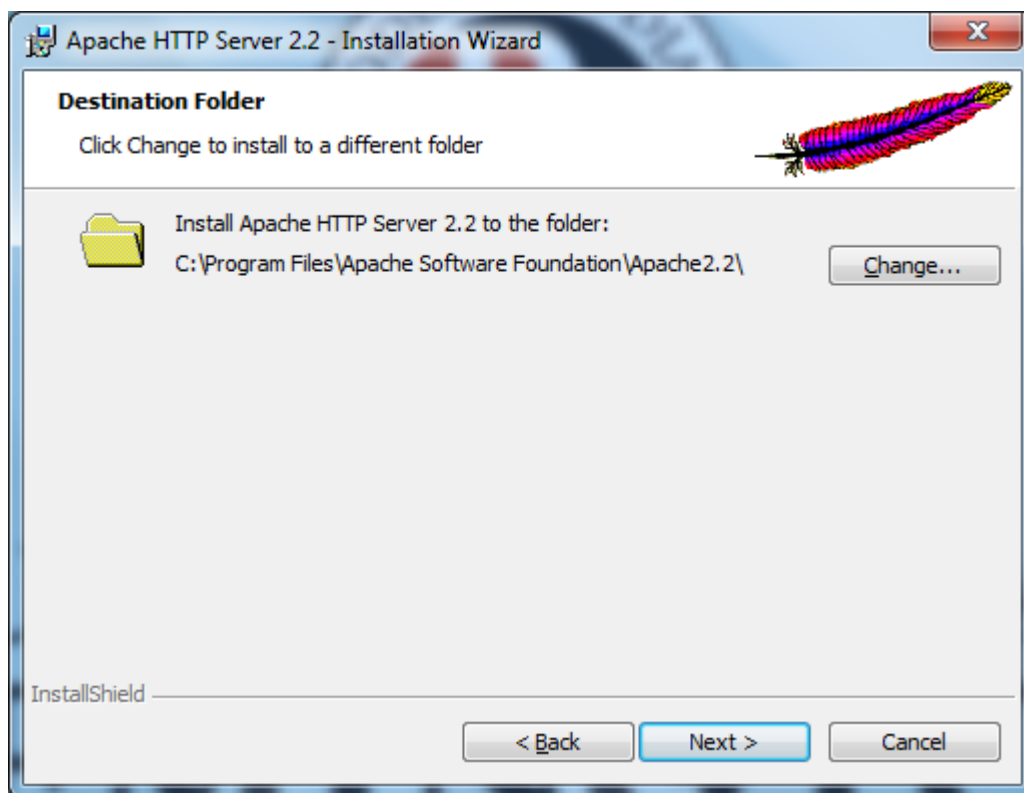


Gráfico Numero18 Se comprueba la existencia de Apache tomcat

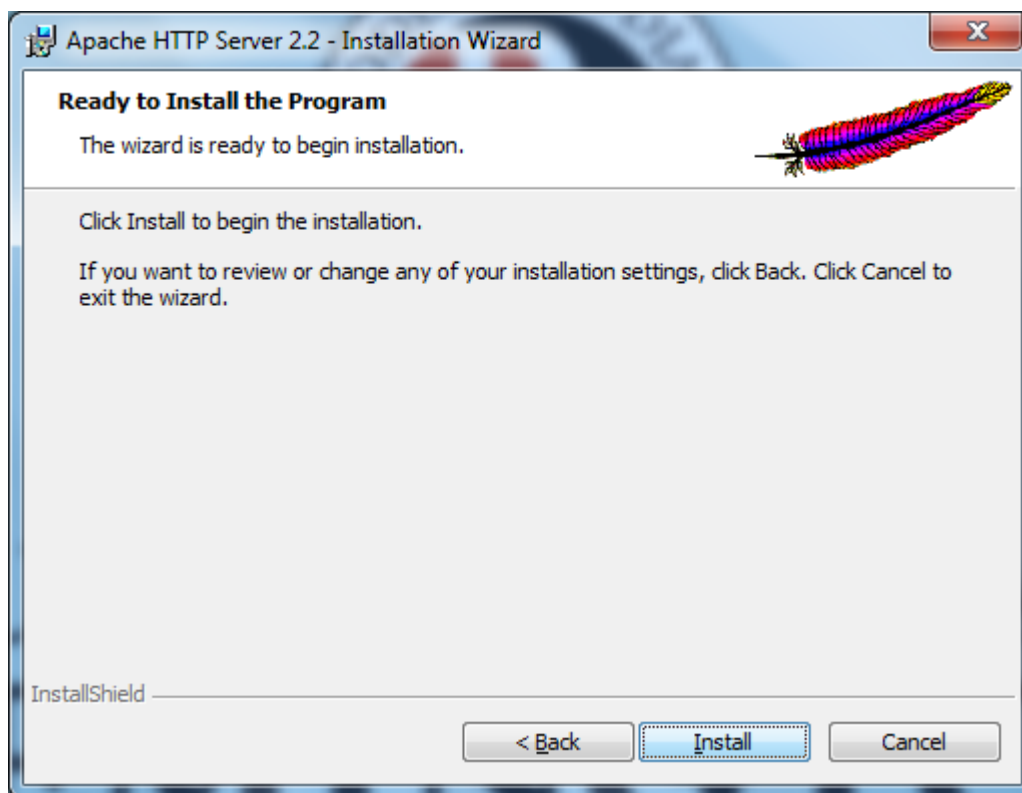


Gráfico Numero19 Se comprueba la existencia de Apache tomcat

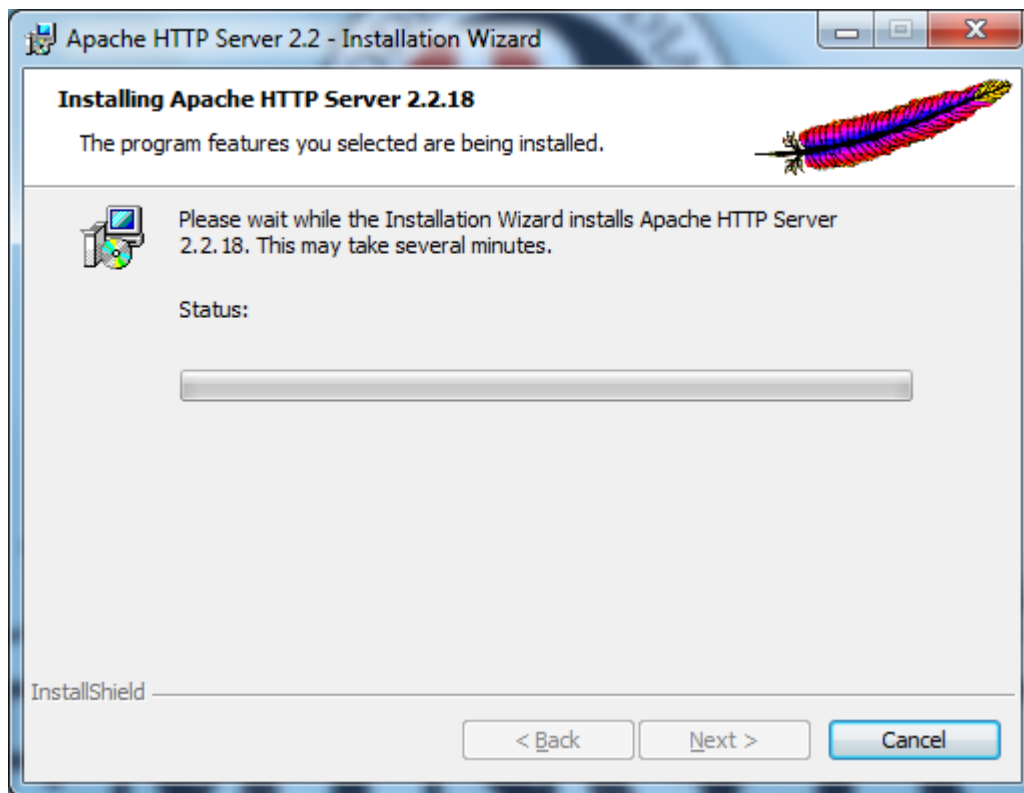


Gráfico Numero20 Se comprueba la existencia de Apache tomcat

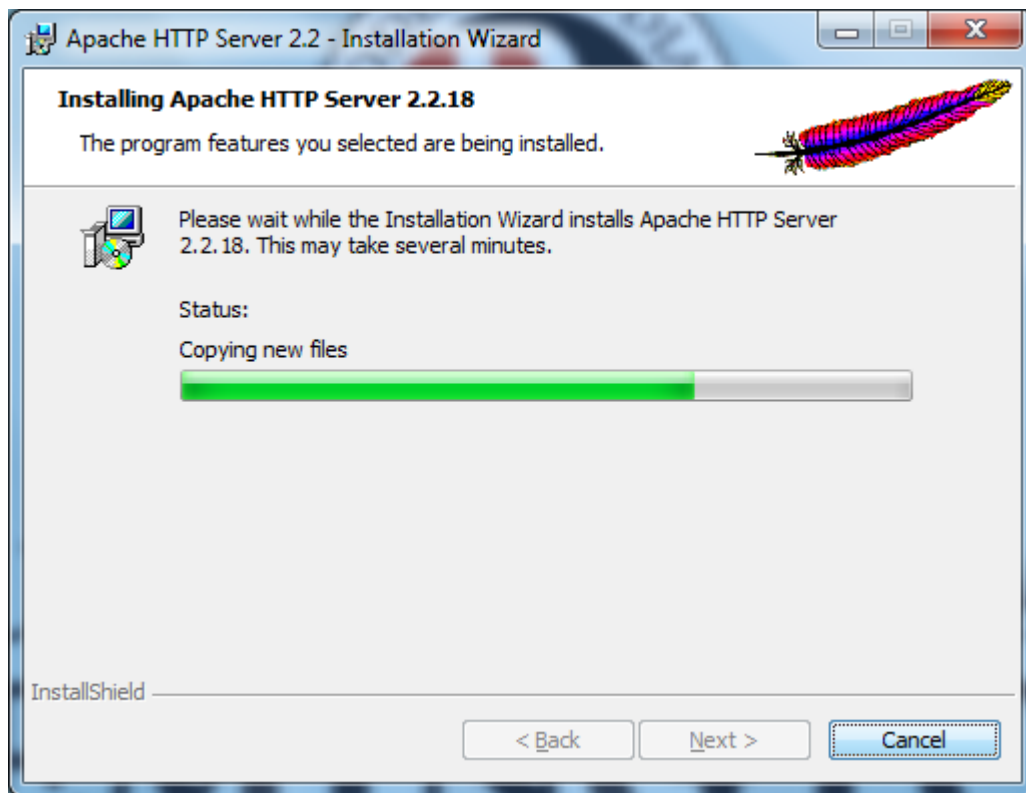


Gráfico Numero21 Se comprueba la existencia de Apache tomcat

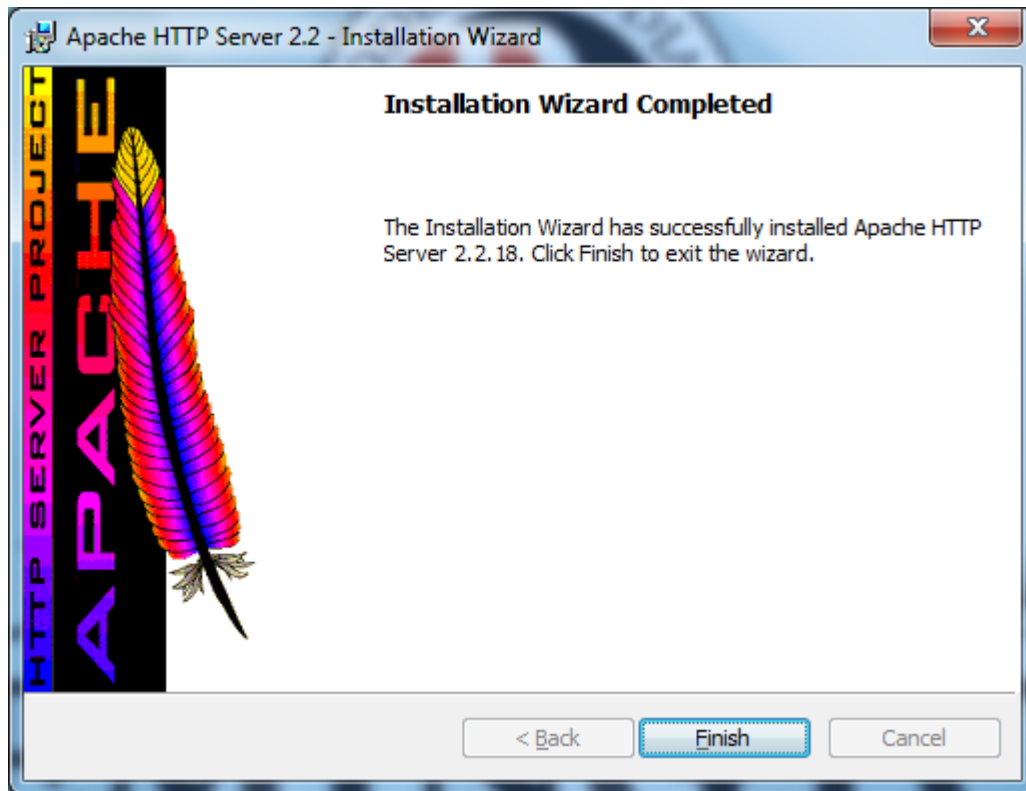


Gráfico Numero 22 Se comprueba la existencia de Apache tomcat

Actividad ejemplo

Realizar la instalación correspondiente a este servidor.

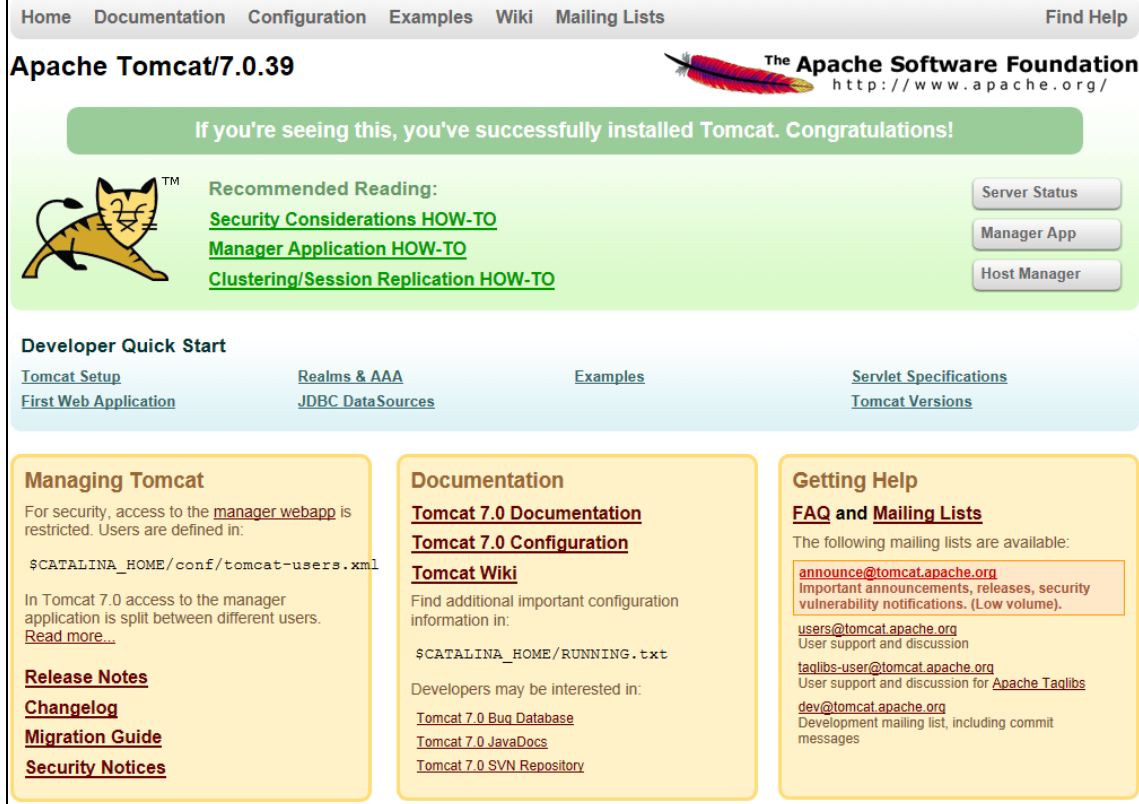
EJERCICIO DE AUTOEVALUACIÓN

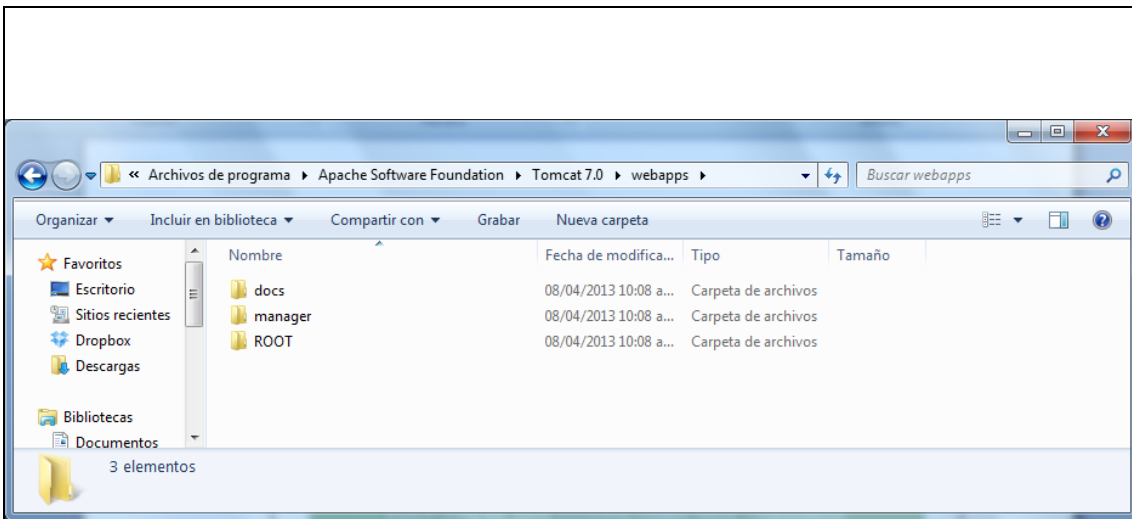
La evaluación Integrando Tomcat y Apache (13 Valor 7.7 %)

Describir conceptos obtenidos Tomcat Apache

Se validan los conceptos de Tomcat Apache y otros.

Verificar las instalaciones correspondientes a los requerimientos establecidos de Tomcat Apache Jdk Editores de las diferentes etapas realizando registro en vista previa:

Nombre del ejercicio de aprendizaje: TOMCAT y APACHE	Datos del autor del taller: César Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Los aplicativos que son del lado del servidor tienen la particularidad de ser muy seguros aunque requieren de un espacio para ejecutar dicha herramienta. La codificación JSP no permite que se muestre directamente con un browser, para ello se requiere un servicio conocido como el tomcat, una herramienta que interpreta al lado del apache que también es utilizada por PHP, para mostrar y procesar los datos.	
 <p>The screenshot shows the Apache Tomcat 7.0.39 homepage. At the top, there's a navigation bar with links: Home, Documentation, Configuration, Examples, Wiki, Mailing Lists, and Find Help. Below this, the title 'Apache Tomcat/7.0.39' is displayed next to the Apache Software Foundation logo and URL. A green banner reads: 'If you're seeing this, you've successfully installed Tomcat. Congratulations!'. Below the banner, there's a section for 'Recommended Reading' with links to 'Security Considerations HOW-TO', 'Manager Application HOW-TO', and 'Clustering/Session Replication HOW-TO'. To the right of these links are buttons for 'Server Status', 'Manager App', and 'Host Manager'. A 'Developer Quick Start' section follows with links for 'Tomcat Setup', 'Realms & AAA', 'Examples', and 'Servlet Specifications'. Below this, there are three main content boxes: 'Managing Tomcat' (with links for Release Notes, Changelog, Migration Guide, Security Notices), 'Documentation' (with links for Tomcat 7.0 Documentation, Tomcat 7.0 Configuration, Tomcat Wiki), and 'Getting Help' (with links for FAQ and Mailing Lists). The 'Managing Tomcat' box also includes a code snippet for setting the SCATALINA_HOME and a link to 'Read more...'. The 'Documentation' box includes a link to 'Find additional important configuration information in:' and a code snippet for SCATALINA_HOME/RUNNING.txt. The 'Getting Help' box includes a list of mailing lists and their purposes.</p>	
<p>Para la ejecución de prueba se utiliza:</p> <p>http://localhost:8080</p> <p>Esta ruta nos mostrará este pantallazo indicándonos que está configurado y listo para utilizar. Los aplicativos jsp se ubican por defecto en esta ruta.</p>	



Ya con esto se puede ejecutar el aplicativo, todo esto está interpretado por Apache.

Pista de aprendizaje:

Tener en cuenta: de la configuración apropiada podremos obtener resultados adecuados.

Tenga presente: si no tenemos Apache o Tomcat configurados no habrá acceso o perderíamos la seguridad en nuestro sitio.

Traer a la memoria: que este aplicativo tiene un compilador interno que nos guiará en la solución de errores.

Taller de entrenamiento:

Nombre del taller: JSP - TomCat	Datos del autor del taller: César Augusto Jaramillo Henao
Actividad previa: Lenguaje de Programación III – Unidad 5.3 Integrando Tomcat y Apache	
Describe la actividad: Crear una BD de estudiante con la tabla Carrera. Diseñar un formulario con los campos que considere necesarios. Configure apropiadamente el tomcat y el eclipse. Ejecute el aplicativo mostrando como reporte los datos almacenados en la tabla.	

6. PISTAS DE APRENDIZAJE

A continuación se presentan los recordatorios respecto a cada uno de los temas tratados en este modulo.

Tener en cuenta

Java 2 Enterprise Edition es la arquitectura creada por Sun para el desarrollo de todo tipo de aplicaciones para empresas y usuarios en general. Sun lo define como un estándar para el desarrollo de aplicaciones empresariales multicapa.

Tenga presente:

Las aplicaciones realizadas en J2EE se pueden dividir desde 2, 3 o más capas:

En la primera capa es donde se encuentran las interfaces como paginas Jsp, Servlet, HTML, Applet, aplicaciones Java.

Traer a la memoria:

La segunda capa contiene subcapas (el contenedor web y el contenedor EJB) encontramos componentes EJB, los Web Services y toda la lógica de negocio.

La última capa es para acceder a Bases de Datos, ERP, EIS.

Tener en cuenta:

Al implementar servicios web J2EE y .NET y consumirlos desarrollando clientes en ambas plataformas, se comprobó en efecto que los servicios web permiten la interoperabilidad.

Pista tema Arquitectura de tres capas 1

Tenga presente:

Estilos de programación que en esencia requieren separar las funcionalidades y lógicas más comunes dependiendo de los objetivos.

Como definir el dominio, los requerimientos (normas del negocio), la trazabilidad, los informes, formularios entre otros además permite la escalabilidad, la integración, la disponibilidad y la seguridad de los datos.

Traer a la memoria

Es un patrón de diseño en el cual se describe la estructura de una aplicación, para ello emplea la arquitectura de capas que van a componer la misma y la funcionalidad de cada una.

El modelo incluye la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación

En una aplicación J2EE el modelo puede ser implementado mediante clases estándar Java. La vista son los resultados generados dinámicamente y proporcionados para dar respuesta al cliente, para ello se debe emplear cualquiera de los servicios web, entre otros los servlets o los jsp.

Tener en cuenta:

Es la propiedad de un objeto por la que su existencia trasciende el tiempo es decir, el objeto continúa existiendo después de que su creador deja de existir; el espacio es decir la posición del objeto varía con respecto al espacio de direcciones en el que fue creado.

La persistencia abarca algo más que la mera duración de los datos en las bases de datos orientada a objetos, no solo persiste el estado de un objeto; si no que su clase debe trascender a cualquier programa individual, de forma que todos los programas interpreten de la misma manera el estado almacenamiento.

Tenga presente

La conectividad de la base de datos de Java (JDBC, Java Database Connectivity) es un marco de programación para los desarrolladores de Java que escriben los programas que tienen acceso a la información guardada en bases de datos específicamente. JDBC se utiliza comúnmente para conectar un programa del usuario con una base de datos por “detrás de la escena”, sin importar qué software de administración o manejo de base de datos se utilice para controlarlo. De esta manera, JDBC es una plataforma-cruzada.

Traer a la memoria

Es importante recordar que `init ()` sólo se ejecuta en la primera invocación al servlet. Por tanto cualquier modificación al archivo `.properties` surtirá efecto una vez que recarguemos el servlet (el manager de Tomcat viene con una opción de 'reload')

En `doPost ()` respondemos a cada invocación al servlet desde el formulario HTML.

La tarea más importante que se realiza aquí es la apertura de una conexión a la base de datos. ¿Dónde poner la conexión a la base de datos? ¿En `init ()` o en `doXX ()`? Dicho de otra forma, ¿nos conectamos en `init ()` o cada vez que se hace una solicitud (un "request" o invocación al servlet)? Lo normal es que queramos que cada usuario abra una conexión a la base de datos, por tanto intentaremos la conexión en respuesta a una solicitud `get` o `post`. Si pusieramos la conexión en el `init ()`, dicha conexión sólo se realizaría la primera vez que se invocase el servlet.

Tener en cuenta:

En nuestro programa java, todos los import que necesitamos para manejar la base de datos están en `java.sql.*`. Puesto que casi todos los métodos relativos a base de datos pueden lanzar la excepción `SQLException`, meteremos todo nuestro programa en un `try-catch`.

Para el caso de no trabajar con Access debemos tener el servidor de MySQL (Tomcat) arrancado. Si hemos instalado y dejado esa opción como estaba, cada vez que encendamos el ordenador, se arrancará el servidor de MySQL, por lo que no tenemos que preocuparnos por ello.

Tenga presente

El externo es el nivel más cercano al usuario, describe la parte que interesa al usuario.

El conceptual describe cuales son los datos reales de la base y que relaciones existen entre los datos. Este nivel contiene la base de datos en términos de unas relaciones sencillas. Estas simples estructuras del nivel conceptual pueden estar reflejadas en complicadas estructuras físicas. Este es el nivel empleado por el administrador de la base de datos

El interno o físico es diferente de uno lógico. La operación de transformar registros lógicos en físicos y viceversa se llama transformación de datos o mapeo

Traer a la memoria

La estructura lógica, en el ámbito conceptual o externo, es la base para la clasificación de los DBMS en las cuatro categorías siguientes: jerárquica, red, relacional y orientada a objetos.

Cualquier categoría debe permitir un acceso aleatorio a los datos requeridos, utilizando para tal fin una estructura de datos: redes, árboles, tablas o listas enlazadas.

Pista tema RMI 1

Con ello se evalúa su creatividad, el aprovechamiento de la herramienta y sus conocimientos como un valor agregado para instalar e implementar el software e instanciarlo de uno u otra manera según el entorno. Para este se estructuran los componentes que invocan a otros procesos no solo dentro de un único entorno sino dentro de diferentes de manera remota esto lo hace RMI (Java Remote Method Invocation) invocación de métodos de manera remota.

Tener en cuenta

RPC o Llamada a Procedimiento Remoto otro tipo de estándar. Definido como un protocolo encargado de ejecutar una aplicación en otro entorno dentro de la misma o en diferente máquina eso si independiente de la comunicación así para (Berger, 2004) RPC, "es la transferencia sincrónica de datos y control entre dos partes de un programa distribuido a través de espacios de direcciones disjuntas".

Tenga presente

También se debe tener en cuenta la Serialización es una clase que al instanciarse se puede copiar en otro entorno a esto se le llama estado de operación remoto.

La seguridad involucra el paquete lang con su clase System quien a su vez incluye SecurityManager y se activa por a setSecurityManager a las diferentes políticas estas emplean el método checkConnect en sus diferentes formas y relaciones

Traer a la memoria

Un Servlet es un programa que se ejecuta en un servidor Web y que es utilizado para generar contenidos dinámicos en los sitios Web, se encarga de recibir una petición de un usuario y retornar un resultado de una búsqueda en una base de datos.

Pueden existir distintos tipos de Servlet, no sólo limitados a servidores Web y al protocolo HTTP(HyperText Transfer Protocol) entre otros

Tener en cuenta:

La mayoría de servlets heredan de HttpServlet.

La JVM del servidor de aplicaciones recibe una solicitud, para dar la respuesta realiza una serie de tareas:

Tenga presente

Java Server Page, es una tecnología del modelo cliente-servidor que hace más dinámica el servicio web, permite el código del lenguaje java y administra las páginas de manera remota según los requerimientos.

Traer a la memoria

Se esta usando un objeto HTML submit del cual se usan las propiedades NAME y VALUE, este objeto tiene como proposito principal activar la acción de la forma (llamarse a si mismo el programa.jsp) y ademas mandar los datos que proporciono el usuario hacia el servidor.

Tener en cuenta:

El teléfono móvil es un dispositivo inalámbrico electrónico que permite tener acceso a la red de telefonía celular o móvil. Se denomina celular en la mayoría de países latinoamericanos debido a las antenas repetidoras que conforman la red, cada una de las cuales es una célula, si bien existen redes telefónicas móviles satelitales (Perez, 2011). Su principal característica es su portabilidad, que permite comunicarse desde casi cualquier lugar. Aunque su principal función es la comunicación de voz, como el teléfono convencional.

Tenga presente

Sun Microsystems establece dos entornos normalizados para la implementación de las aplicaciones java además de J2SE (Java 2 Standard Edition) (Gálvez Rojas & Ortega Diaz, 2004) para las aplicaciones independientes, que es la base de esta tecnología con sus aplicaciones básico java y applet , la primera SE VISUALIZA en la pantalla en ambiente dos o grafica y la segunda desde el explorador incluyendo a html o xml, los otros son J2EE y J2ME.

Traer a la memoria

Framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

Tener en cuenta

La definición para Framework desde el punto de vista de (Soaagenda, 2007) “ambiente de trabajo y ejecución por java o .net, proporcionando librerías (paquetes) para la implementación o conexión empleando componentes “.

Tenga presente

Struts se encarga de normalizar (o al menos intentarlo) el desarrollo de la capa Vista dentro de la arquitectura MVC. Sin embargo, es necesario que para tal fin, también proporcione mecanismos para trabajar con la capa C. Pero nunca Struts se mezclará con la capa del Modelo. Esta característica principal de Struts permite separar la lógica de presentación de la lógica del negocio, con las ya consabidas ventajas que este tipo de desarrollo supone.

Traer a la memoria

Este patrón nos permite/obliga a separar la lógica de control (sabe que cosas hay que hacer pero no como), la lógica de negocio (sabe como se hacen las cosas) y la lógica de presentación (sabe como interactuar con el usuario) empleando jsp de respaldo en la visualización.

Una aplicación JSF se ejecutan en un contenedor web estándar, por ejemplo, Tomcat.

Tener en cuenta

JSF está orientado a prestar el soporte necesario para la construcción de las capas controlador y vista, dejando para las clases normales la capa del modelo.

Tenga presente

Toda aplicación JSF tendrá un archivo de configuración Faces-config.xml, donde definiremos varios elementos claves para el funcionamiento de la aplicación. Es aquí donde definiremos las reglas de navegación por la web, según el resultado que obtengamos en cada operación

Traer a la memoria:

Integra dos servidores que soportan tecnologías de servidores y contenedores WEB, Apache es el servidor WEB más usado a nivel mundial en Internet dada su "simplicidad" y su alta eficiencia. Tomcat (Tomcat, 2006) es un contenedor WEB para la implementación de sitios dinámicos mediante JSPs

Tener en cuenta

Es necesaria una gestión de hilos, ya que será necesario controlar la situación en la que tenemos una instancia de un componente (por ejemplo, un servlet) que da respuesta a varias peticiones, donde cada petición se resuelve en un hilo.

Tenga presente

Para cada solicitud se genera un hilo (thread) para resolverla (pero con una única instancia de la clase, como hemos dicho).

Traer a la memoria:

Un Application Server tendrá un servidor de administración (y normalmente un manager de las aplicaciones).

Comprobar la intalacion respectiva

7. GLOSARIO

JCP (java community Process) Procesos comunitarios de especificaciones, dirigido por Sun Microsystems.

JAVABEAN (almacenamiento temporal)

MVC Modelo vista control

JDBC ((Java Database Connectivity)

SGDB Sistema manejador de base de datos

SERVLET Estructura conformada por módulos para la web, útil para ampliar la capacidad del despliegue del servidor y resolver requerimientos

J2ME plataforma apropiada para dispositivos inalámbricos electrónicos como los teléfonos móviles llamados celulares, PDAs o similares

Struts se define como un Framework Java MVC2 para desarrollar aplicaciones

JSF: (Java Server Faces) Es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java con modelo vista controlador

Tomcat: es un contenedor Web para la implementación de sitios dinámicos mediante JSPs

JSPs: Son estructuras etiquetadas propias para la web, bajo el lenguaje de programación java

Tipo: describe propiedades de abstracciones de aplicables el lenguaje de programación

Concurrencia: Propiedad diferenciadora de un objeto activo con un no activo.

Persistencia: propiedad de permanecer obtener espacio tiempo(memoria)

UMTS, es comúnmente conocido como red 3G o 3.5G, cuya velocidad de Internet en

http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil

CLDC (Connected Limited Device Configuration) dispositivos con restricciones de procesamiento y memoria,.

CDC Connected Device Configuration dispositivos con más recursos.

DTOs (Data Transfer Objects), también llamados Value Objects (VO).

(JSF) Java Server Faces

Http (HyperText Transfer Protocol

JDBC ((Java Database Connectivity)

DBMS Sistema de gestión de base de datos (Database management system)

SQL lenguaje de consulta estructurado (structured query language).

DDL Lenguaje de definición de datos (Data Definition Language).

BYTECODES Realiza la compilación y la interpretación de un programa fuente para luego utilizarlo en cualquier plataforma

JVM La máquina virtual de Java permite el uso de una o más aplicaciones compiladas independiente del equipo y su sistema

RTGC RadioTelefonía Grupo Cerrado (Trunking) sistemas donde un conjunto de canales de radio soporta a uno o mas móviles, por un sistema dinámico de asignación de frecuencias.

CGI Common Gateway Interface

AppServ programa agrupa aplicaciones y utilidades, de manera que al instalarlo, se configuran y predisponen para su uso

Apache es un servidor HTTP de código abierto disponible para diferentes plataformas(GNU/Linux, Windows, Macintosh).

RMI (Java Remote Method Invocation) invocación de métodos de manera remota.

8. FUENTES BIBLIOGRÁFICAS

Libros

Mario Camou Riveroll; Jomes Keogh, J2EE. MANUAL DE REFERENCIA

Adela Sancho Hernández; Jesús Bobadilla Sancho , COMUNICACIONES Y BASES DE DATOS CON JAVA A TRAVÉS DE EJEMPLOS

Art Taylor; Brian Buege; Randy Layman, HACKERS DE JAVA Y J2EE. DESARROLLE APLICACIONES JAVA SEGURAS

EErich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software.

Olav Maassen; Stephen Stelting, PATRONES DE DISEÑO APLICADOS A JAVA

Booch, Grady. Análisis y Diseño Orientado a Objeto

Cepeda, J. M.: Metodología de la enseñanza basada en competencias...

Revista Iberoamericana de Educación (ISSN: 1681-5653)

<http://www.rieoei.org/deloslectores/709Cepeda.PDF>

8.1.1. Referencias

Allamaraju, S. et al. Professional Java Server Programming J2EE Edition. Wrox Press, Inc., 2000.

BEA Systems. BEA WebLogic Server. <http://www.bea.com/products/weblogic/server/index.shtml>.

Brown, R.G. and Hahn, W. Choices in server-side programming: a comparative programming exercise. In Proceedings of the conference on On track to the 21st century. International Conference on APL, August 10 - 14, 1999, Scranton, Penn. <http://www.acm.org/pubs/articles/proceedings/apl/312627/p35-brown/p35-brown.pdf>.

iPlanet E-Commerce Solutions. Product Map. http://www.iplanet.com/products/product_map/product_name_2_0a.html.

The Jakarta Project. Jakarta Tomcat. <http://jakarta.apache.org/tomcat>

Booch, Grady. Análisis y Diseño Orientado a Objeto

The Jakarta Project. Tomcat User's Guide. <http://jakarta.apache.org/tomcat/tomcat-3.3-doc/tomcat-ug.html>.

8.1.2. Internet

Sun Microsystems. Hierarchy For Package javax.servlet.

<http://java.sun.com/j2ee/tutorial/api/javax/servlet/package-tree.html>.

Sun Microsystems. Interface javax.servlet.Servlet.

<http://java.sun.com/products/servlet/2.1/api/javax.servlet.Servlet.html>.

Sun Microsystems. The J2EE™ Tutorial. <http://java.sun.com/j2ee/tutorial/doc/HTTP.html>.

Sun Microsystems. Java™ Servlet Technology: Implementations and Specifications.

<http://java.sun.com/products/servlet/download.html>.

Sun Microsystems. JavaServer Technologies, Part I.

<http://developer.java.sun.com/developer/technicalArticles/Servlets/JavaServerTech1/index.html>.

Sun Microsystems. A New Model for Enterprise Applications.

<http://java.sun.com/j2ee/overview2.html>.

(www.geneura.ugr.es/~jmerelo/JSP/ , 2004)

<http://www.usabilidadweb.com.ar/JavaMovil/javaMovil.php>

<http://www.vsf.es/web/es/tecnologia.html>

http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil
Libro pdf

http://books.google.es/books?id=USaAQ0hHQWIC&pg=PA28&source=gb_s_selected_pages&cad=3#v=onepage&q&f=false<http://www.adictosaltrabajo.com/tutoriales/strutsb/images/captura1.JPG>

<http://www.adictosaltrabajo.com/tutoriales/strutsb/images/captura2.JPG>

<http://www.adictosaltrabajo.com/tutoriales/strutsb/images/captura3.JPG>

<http://www.adictosaltrabajo.com/tutoriales/strutsb/images/captura4.JPG>

<http://www.desarrolloweb.com/articulos/2380.php>

(www.taringa.net/posts/downloads/9273991/j2me-lo-mejor-de-java-_tecnologia-movil.html, 2003)

<http://www.java.com/es/about/>

<http://www.taringa.net/posts/downloads/9273991/j2me-lo-mejor-de-java- tecnologia-movil.html>

http://200.2.13.202/tl_files/IngenieriaInformatica/Electivas/TecnologiasJAVAyPatronesdediseno paraelWeb.pdf

<http://www.adrformacion.com/cursos/j2ee/j2ee.html>

<http://www.scribd.com/doc/13059254/jsp-Angelestebangrupoeidos>

<http://www.coreservlets.com/JSF-Tutorial/>

<http://soaagenda.com/journal/articulos/que-son-los-frameworks/>

<http://translate.google.com/translate?hl=es&sl=en&u=http://www.vogella.de/articles/JavaServerFaces/article.html&ei=rtbeTbybOM2ftgeOyoWLCg&sa=X&oi=translate&ct=result&resnum=10&ved=0CGkQ7gEwCQ&prev=/search%3Fq%3Djsf%2Btutorial%26hl%3Des%26biw%3D1259%26bih%3D816%26prmd%3Dvns>

8.1.3. Trabajos citados

<http://es.scribd.com/doc/12786595/Articulo-ANÁLISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET->. (abril de 2003). Obtenido de <http://es.scribd.com/doc/12786595/Articulo-ANÁLISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET-->

http://java.ciberaula.com/articulo/disenio_patrones_j2ee/. (2003).

<http://www.mailxmail.com/curso-procesamiento-datos-oracle/sistema-manejador-base-datos>. (12 de junio de 2003). Recuperado el mayo de 2011

www.desarrolloweb.com/articulos/2380.php. (13 de abril de 2003). Recuperado el 09 de marzo de 2011, de <http://www.desarrolloweb.com/articulos/2380.php>

www.taringa.net/posts/downloads/9273991/j2me-lo-mejor-de-java-_tecnologia-movil____.html.
(22 de marzo de 2003). Recuperado el 11 de abril de 2011

[//www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350](http://www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350) de
Google. . (10 de marzo de 2004). Recuperado el 11 de abril de 2011, de

http://www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350 de
Google.

www.geneura.ugr.es/~jmerelo/JSP/ . (2004). Obtenido de <http://www.geneura.ugr.es/~jmerelo/JSP/>
http://java.ciberaula.com/articulo/tecnologia_java/. (22 de febrero de 2005). Recuperado el 2 de
agosto de 2010

<http://www.abcdatos.com/tutoriales/tutorial/z4303.html>. (18 de marzo de 2006). Recuperado el
11 de marzo de 2011

http://www.programacionfacil.com/java_jsp/grabacion_archivos_disco. (12 de abril de 2006).
Recuperado el 01 de mayo de 2011

http://www.programacionfacil.com/java_jsp/java_server_page. (12 de abril de 2006). Recuperado
el 01 de mayo de 2011

<http://casidiablo.net/uso-del-objectoutputstream-java/>. (22 de mayo de 2008). Recuperado el 12
de abril de 2011

González Almirón, C. (10 de abril de 2003).

[htwww.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava](http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava) de Google.
Recuperado el 09 de mayo de 2011

Gonzalez Almiron, C. (20 de febrero de 2003). www.adictosaltrabajo.com/tutoriales. Recuperado
el marzo25 de 2011

Medina Montenegro, E. (11 de febrero de 2003).

www.adictosaltrabajo.com/tutotiales.php?pagina=struts. Recuperado el 22 de mayo de 2010

9. BIBLIOGRAFÍA

<http://es.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET->. (abril de 2003). Obtenido de <http://es.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET->

http://java.ciberaula.com/articulo/disenio_patrones_j2ee/. (2003).

<http://www.mailxmail.com/curso-procesamiento-datos-oracle/sistema-manejador-base-datos>. (12 de junio de 2003). Recuperado el mayo de 2011

www.desarrolloweb.com/articulos/2380.php. (13 de abril de 2003). Recuperado el 09 de marzo de 2011, de <http://www.desarrolloweb.com/articulos/2380.php>

www.taringa.net/posts/downloads/9273991/j2me-lo-mejor-de-java-_tecnologia-movil____.html. (22 de marzo de 2003). Recuperado el 11 de abril de 2011

[//www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350](http://www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350) de Google. . (10 de marzo de 2004). Recuperado el 11 de abril de 2011, de http://www.programacion.com/articulo/primera_aplicacion_con_jsf_java_server_faces_350 de Google.

www.geneura.ugr.es/~jmerelo/JSP/ . (2004). Obtenido de <http://www.geneura.ugr.es/~jmerelo/JSP/> http://java.ciberaula.com/articulo/tecnologia_java/. (22 de febrero de 2005). Recuperado el 2 de agosto de 2010

<http://www.abcdatos.com/tutoriales/tutorial/z4303.html>. (18 de marzo de 2006). Recuperado el 11 de marzo de 2011

http://www.programacionfacil.com/java_jsp/grabacion_archivos_disco. (12 de abril de 2006). Recuperado el 01 de mayo de 2011

http://www.programacionfacil.com/java_jsp/java_server_page. (12 de abril de 2006). Recuperado el 01 de mayo de 2011

<http://casidiablo.net/uso-del-objectoutputstream-java/>. (22 de mayo de 2008). Recuperado el 12 de abril de 2011

González Almirón, C. (10 de abril de 2003).

<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava> de Google.
Recuperado el 09 de mayo de 2011

Gonzalez Almiron, C. (20 de febrero de 2003). www.adictosaltrabajo.com/tutoriales. Recuperado el marzo25 de 2011

Medina Montenegro, E. (11 de febrero de 2003).
www.adictosaltrabajo.com/tutotiales.php?pagina=struts. Recuperado el 22 de mayo de 2010