UNIVERSIDAD AUTONÓMA DE NUEVO LEÓN

FCFM

UNIDAD DE APRENDIZAJE: LABORATORIO DE DISEÑO ORIENTADO A OBJETOS.

PRACTICA #9

PROFESOR. MIGUEL ÁNGEL SALAZAR S.

ALUMNA. VALERIA MARTÍNEZ DE LA ROSA

MATRICULA. 1678575

San Nicolás de los garza, nuevo león a 2 de abril de 2017

CONTENIDO:

USO DE SINGLETON.

Introducción: Se abrirá netbeans, elegirás java web y web aplicación, tecleas el nombre del proyecto (laboratorio9), clic en next, y sin elegir frameworks.

En la carpeta web pages crea dos archivos jsp, con los nombres de buscar y error.

En la carpeta source packages, crea tres folder's, llamados, controller, models y utils. En controller crea un servlet llamado ComentariosControlador. En models, crea dos archivos java, ComentariosDAO y ComentariosPojo. Finalmente en utils, crea un archivo java llamado Log. Y te quedara estructurado de la siguiente manera, tu proyecto:

Index:

Se crea un formulario, con acción al ComentariosControlador y método post. Tendrá un campo input para el nombre, de tipo text, con un nombre y sin valor. Un campo textarea para el comentario, con 5 filas y 15 columnas, y de nombre comentario. Un botón de tipo submit, con el valor de comentar. Otro botón de tipo hidden, con el nombre action y el valor comentar.

```
<body>
    <form action="ComentariosControladors" method="POST">
        <label>Nombre:</label>
            <input type="text" value="" name="nombre"><br/>
        <label>Comentario:</label>
            <textarea rows="5" cols="15" name="comentario"></textarea><br/>
        <input type="submit" value = "comentar">
        <input type="hidden" name="action" value="comentar">
    </form>
</body>
```

Buscar:

Es lo mismo que se hizo con el index, a diferencia que el botón de tipo submit su valor es buscar e igualmente el de tipo hidden.

```
<form action="ComentariosControladors" method="POST" >
    <label>Nombre: </label>
        <input type="text" value="" name="nombre"><br/>
    <label>Comentario: </label>
        <textarea rows="5" cols="15" name="comentario"></textarea><br/>
    <input type="submit" value = "buscar">
    <input type="hidden" name="action" value="buscar">
</form>
```

Con código de java, comparamos sí los datos de la sesión son diferentes de nulo, sí es así, crea una lista de arreglos llamada comentarios y obtiene el valor de comentarios. Y vuelve a comparar a este último, sí es diferente de nulo, entonces crea una tabla, con los campos nombre y comentario.

```
<%  if(session != null){
    ArrayList  comentarios = (ArrayList)session.getAttribute("comentarios");
    if(comentarios!=null){
%>
        <table border="1">
            <tr>
                <th>Nombre: </th>
                <th>Comentario: </th>
            </tr>
```

Se crea un for, dónde este crea Objets de comentarios, y los guarda en una variable comentario de tipo ComentariosPOJO. Se obtendrá el valor del nombre y comentario, y se almacenara en los campos de la tabla.

```
            <%
                for(Object o : comentarios){
                    ComentariosPOJO comentario = (ComentariosPOJO) o;
            %>
                <tr>
                    <td><%=comentario.getNombre()%></td>
                    <td><%=comentario.getComentario()%></td>
                </tr>
            <%  } %>
            </table>
        <%  }
            }%>
```

Error:

Se mostrará un mensaje de error al usuario. Opcionalmente puedes agregar dos links dónde re-direccione a la página de buscar y a la de comentar.

```
<body>
    <h1>Ups! algo salió mal</h1>
    <a href="buscar.jsp" value="buscar" >Buscar</a></br>
    <a href = "index.html" value = "comentario"> Comentar</a>
</body>
```

ComentariosPOJO:

Crea dos variables privadas de tipo String, con el nombre nombre y comentario. Y crea getters y setters para cada uno de estos.

```
public class ComentariosPOJO {
    private String nombre;
    private String comentario;


    public String getNombre() {
        return nombre;
    }

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

    public String getComentario() {
        return comentario;
    }

    public void setComentario(String comentario) {
        this.comentario = comentario;
    }
}
```

ComentariosDAO:

Se importará lo siguiente.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
```

Se crea una variable privada de tipo Connection llamada conexión.

Se crea un método llamado abrirConexion() privado y que no devuelva nada. Tendrá 3 variables, de tipo String. dbURI(con el valor de la dirección de tu base de datos), username(el valor del username de la bdd) y password(se asigna el valor de la contraseña de la bdd). Nota, esto no debería hacerse de esta manera.

```
public class ComentariosDAO {

    private Connection conexion;

    private void abrirConexion() throws SQLException{
        String dbURI = "jdbc:derby://localhost:1527/Comentarios";
        String username = "fcfm";
        String password = "lsti01";
        conexion = DriverManager.getConnection(dbURI, username, password);
    }
```

Se crea otro método, llama cerrarConexion(), privado y que no devuelva nada. Dónde cierra la conexión con el método close().

```
private void cerrarConexion() throws SQLException{
    conexion.close();
}
```

Se crea otro método, llamado insertar, con un parámetro, dónde recibe un objeto de tipo ComentariosPOJO, que sea público y booleano.

Dentro de un try, abre la conexión con el método abrirConexion(), crea una variable String, llamada insert dónde se guarda los valores de nombre y comentario de la bdd. Crea una variable de tipo Statement, llamada stmt y guarda en esta un objeto SQLServerStatement para enviar instrucciones

SQL a la base de datos. En una variable de tipo entero, el método executeUpdate(String loquesea) qué guarda lo que ejecuta el sql, y este tiene como argumento la variable insert. Invoca al método cerrarConexion(), y devuelve la cantidad de filas mayor a 0. Sino devuleve un false.

```
public boolean insertar(ComentariosPOJO cm){
    try{
        abrirConexion();
        String insert = "insert into COMENTARIOS values ('" + cm.getNombre()+ "','" + cm.getComentario() + "')";
        Statement stmt = conexion.createStatement();
        int filas = stmt.executeUpdate(insert);
        cerrarConexion();
        return filas >0 ;
    }catch(Exception ex){
     return false;
    }
}
```

Crea un método llamado buscar(), que tiene como parámetro un objeto de tipo ComentariosPOJO, público y de tipo ArrayList.

Crea un ArrayList de ComentariosPOJO, llamada beans, y guarda en este un nuevo arrayList.

```
public ArrayList buscar(ComentariosPOJO pojo){
    ArrayList<ComentariosPOJO> beans = new ArrayList();
```

Y agrega un try, y dentro de este abre la conexión, crea una variable insert de tipo String en la que guarda código de sql para obtener los valores de nombre y comentario. Crea una variable Statement llamada stmt y guarda en esta un objeto SQLServerStatement para enviar instrucciones SQL a la base de datos. Crea una variable de tipo ResultSet llamada result dónde guarde Un objeto SQLServerResultSet.

```
try{
    abrirConexion();
    String insert = "select * from COMENTARIOS where NOMBRE = '" + pojo.getNombre() + "' and COMENTARIO like '%" + pojo.getComentario() + "%'";
    Statement stmt = conexion.createStatement();
    ResultSet result = stmt.executeQuery(insert);
```

Después crea un while, crea dos variables de tipo String en dónde se obtiene el nombre, y el comentario. Crea una variable de tipo ComentariosPOJO donde guardes un nuevo objeto del mismo. Y establece el nombre y el comentario. En la variable beans agrega el objeto POJO creado.

Después cierra la conexión, y devuelve la variable beans.

```
            while(result.next()){
                String nombre = result.getString("Nombre");
                String comentario = result.getString("Comentario");
                ComentariosPOJO cmt = new ComentariosPOJO();
                cmt.setNombre(nombre);
                cmt.setComentario(comentario);
                beans.add(cmt);
            }
            cerrarConexion();
        }catch(Exception ex){
        }
        return beans;
    }
```

ComentariosControlador:

Obtendrás los valores de los campos de las paginas principales, crea una variable de tipo HttpSession y obtén el request de la sesión. Crea una variable de tipo ComentariosDao y guarda en ella un objeto del mismo, y de igual manera para ComentariosPOJO. Establece el valor de nombre, comentario.

Crea una variable de tipo Log, en dónde guardes una instancia del método getInstance() y tendrá cómo argumento el url, de tu archivo.

Hace una instancia de Log y concatena con el método write y tiene como argumento la variable nombre.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String action = request.getParameter("action");
    String nombre = request.getParameter("nombre");
    String comentario = request.getParameter("comentario");
    HttpSession session =request.getSession();
    ComentariosDAO dao = new ComentariosDAO();
    ComentariosPOJO pojo = new ComentariosPOJO();
    pojo.setNombre(nombre);
    pojo.setComentario(comentario);
    Log log = Log.getInstance("C:\\Users\\Valeria\\Downloads\\DOO\\laboratorio9\\Log.txt");
    log.write(nombre);
```

Compara sí la variable acción es igual a comentar, de ser así hace uso del método insertar y re-dirige a la pagina buscar.

Sino compara sí action es igual a buscar, de ser así, crea una lista de arreglos de ComentariosPOJO y guarda el método buscar. Y re-dirige a la

página buscar.

Sí no es ninguna de estas dos, re-dirige a la página de error.

```
    if(action.equals("comentar")){
        dao.insertar(pojo);
        response.sendRedirect("buscar.jsp");
    } else {
        if(action.equals("buscar")){
            ArrayList <ComentariosPOJO> comentarios = dao.buscar(pojo);
            session.setAttribute("comentarios", comentarios);
            response.sendRedirect("buscar.jsp");
        } else {
            if(!"comentar".equals(action) & !"buscar".equals(action)){
            response.sendRedirect("error.jsp");
            }
        }
    }
}
```

Log:

Crea dos variables privadas, una final de tipo String llamada fileName, y otra estática de tipo Log llamada instance.

Crea un constructor con un parámetro de tipo String.

```
public class Log {
    private final String fileName;
    private static Log instance;

    private Log(String fileName){
        this.fileName=fileName;
    }
```

Crea un método público, estático y de tipo Log llamado geinstance con un parámetro String. Dentro de este compara si la instancia es nula, de ser así crea un objeto Log y la guarda en la variable instancia, sino devuelve la instancia.

```
public static Log getInstance(String fileName){
    if(instance==null)
    {
        instance = new Log("C:\\Users\\Valeria\\Downloads\\DOO\\laboratorio9\\Lo
    }
    return instance;
}
```

Crea un método llamado write, que sea público y no devuelva nada, con un parámetro de tipo String.

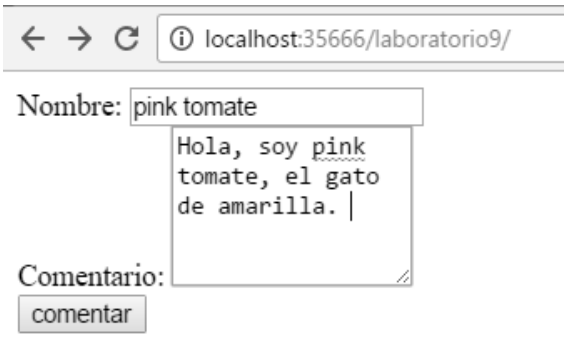
Dentro de este método arega un try y dentro del mismo otro try, esto para evitar errores. Y escribe lo siguiente.

```
public void write(String message){
    try{
        try (BufferedWriter br = new BufferedWriter(new FileWriter(fileName, true))) {
            DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            Calendar cal = Calendar.getInstance();

            //Create the name of the file from the path and current time
            String data = "\n" + dateFormat.format(cal.getTime()) + ": " + message ;
            br.write(data);
        }
    }catch(Exception ex){
    }
}
```

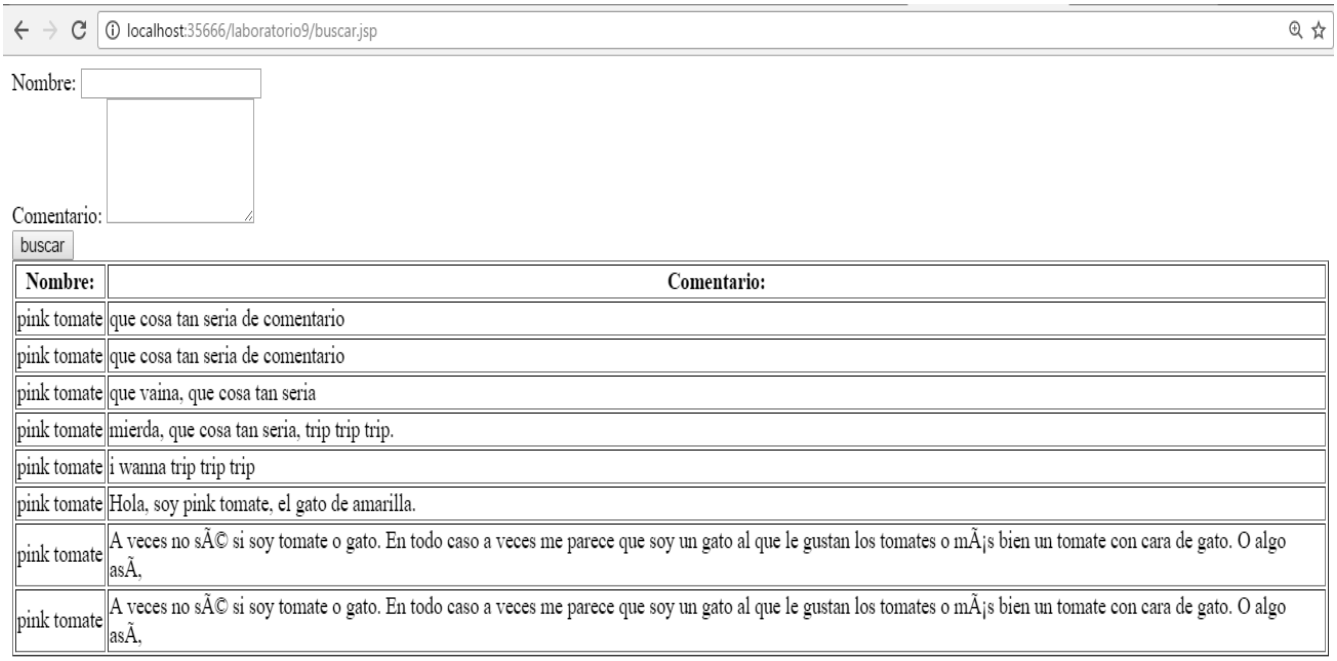
Pantallas:

La primera pantalla a aparecer por ende será el index. Dónde podemos agregar los comentarios a la base de datos.



Después te redirige a la página de buscar.

Podemos buscar todos los comentarios de un usuario ó todos los comentarios qué tengan una palabra en especifico.



Nombre:	Comentario:
pink tomate	que cosa tan seria de comentario
pink tomate	que cosa tan seria de comentario
pink tomate	que vaina, que cosa tan seria
pink tomate	mierda, que cosa tan seria, trip trip trip.
pink tomate	i wanna trip trip trip
pink tomate	Hola, soy pink tomate, el gato de amarilla.
pink tomate	A veces no sÃ© si soy tomate o gato. En todo caso a veces me parece que soy un gato al que le gustan los tomates o mÃ¡s bien un tomate con cara de gato. O algo asÃ,
pink tomate	A veces no sÃ© si soy tomate o gato. En todo caso a veces me parece que soy un gato al que le gustan los tomates o mÃ¡s bien un tomate con cara de gato. O algo asÃ,

Sino re-dirige a la página de error.

Preguntas:

- ¿Qué ventajas identificas con el uso de un sistema de Logging de eventos?

  Se crea un control, de cuantas veces y en qué horas cierto usuario uso la opágina.

- ¿Qué ventajas tienes al utilizar una clase singleton?

  Cuando tenemos datos en una aplicación que van a ser compartidos por muchas clases de la misma y no los queremos cargar en memoria todas las veces que los vayamos a utilizar porque nos ocupa mucho tiempo.
  Podemos hacer uso de la programación orientada a objetos, por ejemplo herencia, etc.

- ¿Qué "pros" y "contras" identificas al utilizar singleton vs clases estáticas?

  Las clases estáticas proporcionan un mejor rendimiento ya que son enlazadas en tiempo de compilación. Las clases Singleton, al tratarse de un objeto, proporciona características propias.

  Las clases estáticas son más rápidas que las singleton.