Universidad Simón Bolívar
Departamento de Computación
Taller de Desarrollo de Software

Guía de Struts 1.x

Alejandro Ballesta #08-10090
Jesus Rondon #0437545
Andres Cordova #0538050

Introducción

El propósito de esta guía es proveer una breve introducción al Framework Struts 1.x para aquellas personas que deseen aprender a utilizarlo, o quienes quieran refrescar sus conocimientos. Es recomendable que utilice un IDE para programar este tipo de aplicaciones, ya que le permite mantener el orden necesario a la hora de crear aplicaciones web con java. En este ejemplo utilizaremos Eclipse Java EE, aunque también es recomendable que utilice Netbeans que viene con Tomcat ya incluido.

No vamos a constuir una aplicación en struts completa. Lo que haremos será mostrar los esqueletos básicos de todas las capas explicadas detalladamente, de manera que usted pueda tener un entendimiento integral de cómo se usa este framework.

Contenido

Modelo

El manejador de base de datos que utilizaremos será PostgreSQL. Nos conectaremos a la base de datos con el driver JDBC. Es necesario que lo descargue: http://jdbc.postgresql.org/ y lo agregue al classpath de su proyecto.

Ahora vamos a crear la capa del modelo de la aplicación, que se encargará de conectarse a la base de datos, y ejecutar las consultas a través de una serie de métodos que también programaremos en esta capa.

Creamos entonces un paquete que se llame, por ejemplo “mbd” por manejador de base de datos, y dentro del paquete creamos una clase que llamaremos ManejadorBasedeDatos.java. Esta clase contiene la capa del modelo.

Mostaremos ahora un esqueleto del modelo de una aplicación de struts con los métodos más importantes.

```
public class ManejadorBasedeDatos {
	private static Connection conexion;
	private static DatabaseManager instancia = null;


	public ManejadorBasedeDatos(){

	}
//En el metodo conectar usamos el driver JDBC de postgresql
	public static boolean conectar(){
		try{
		Class.forName("org.postgresql.Driver");
		conexion = DriverManager.getConnection(
				"jdbc:postgresql://localhost:5432/EOE", //EOE →nombre de bd
				"postgres", //nombre usuario bd
				"BrownTabl3" //password
				);
		return true;
		}catch (Exception e){//Si hay un error lo muestra
			System.out.println("**************aqui esta= ");
					e.printStackTrace();
		}
		return false;
	}

public void desconectar(){ //cierra la conexión con la base de datos
		try{
			DatabaseManager.conexion.close();
			conexion = null;
		}catch(Exception e){
			System.err.println(e.getMessage());
		}

	}
```

```
	//Crea una instancia de la base de datos. Esto se hace en el controlador
	//para poder llamar los métodos de la clase ManejadorBaseDatos que ejecutan
	//las consultas
	static public ManejadorBasedeDatos Instanciar(){
		if (null == ManejadorBasedeDatos.instancia){
			ManejadorBasedeDatos.instancia = new ManejadorBasedeDatos();
		}
		conectar();
		return DatabaseManager.instance;

		}

	}
```

Ahora, mostraremos un ejemplo de un método que ejecuta una consulta a la base de datos:

```
/*Consulta las “materias” en la base de datos. Es conveniente usar un “ArrayList”. Contiene elementos
* de la clase Materia, y además se comporta como una lista *dinámica, ya que no tiene que especificar
* previamente el tamaño del arreglo */
public ArrayList<Materia> consultarMaterias(){
		String consulta = "SELECT * FROM asignatura;";
		ArrayList<Materia> materias = new ArrayList<Materia>(0);
		 Connection c = DatabaseManager.conexion;
		try{
			Statement statement = c.createStatement();
			ResultSet resultados = statement.executeQuery(consulta);
			if(!resultados.next()){
				System.out.println("Error ejecutando las consultas");
				return null;
			}
			//En este ciclo, se crea una instacia de la clase Materia, y se
			//construye con los valores obtenidos de la base de datos
			while(resultados.next()){
//el metodo getString obtiene el dato de la fila actual con el nombre especificado
				Materia materia = new
Materia(resultados.getString("codigo"),resultados.getString("nombre"),"");

//se agregan las materias al arraylist
				materias.add(materia);
			}

		}catch(SQLException exception){
			System.err.println("Modelo: Error en consulta de materias");
			return null;
		}

		return materias; //El método retorna una lista de materias
	}
```

La clase Materia utilizada hay que implementarla también, ya que es un objeto propio del modelo de la base de datos. Podemos crear un paquete llamado “clases” u “objetos” que contendrá todos los objetos que se encuentran en la base de datos y con los cuales queremos interactuar. El esqueleto de una clase del Modelo sería de esta manera, en el caso de Materia:

```java
public class Materia{

	//Propiedades
	private String id;
	private String nombre;
	private String Mu;

	//Constructores
	public Materia(String id, String nombre, String Mu){
		this.id = id;
		this.nombre= nombre;
		this.Mu = Mu;
	}


	public Materia(){
	}


	//Estos son los “getters y setters” nos permiten obtener los valores
	// de las propiedades de las instancias de esta clase.
	public String getId(){//obtiene el ID de una materia
		return id;
	}

	public void setId(String id){ //actualiza el ID de una materia
		this.id = id;
	}

	public String getNombre(){ //obtiene el nombre de una materia
		return nombre;
	}

	public void setNombre(String nombre){ //actualiza el nomrbe de una materia
		this.nombre = nombre;
	}

}
```

Controlador

La capa del Controlador, se encarga de recibir información de la vista, y pasársela al modelo para que haga las consultas a la base de datos. Luego, recibe los datos que envía la función del modelo que ejecutó, y las pasa a la vista.

A continuación verá el esqueleto del controlador de una aplicación:

```
public class VerMaterias extends DispatchAction {

	private static final String SUCCESS   = "success";
	private static final String FAILURE = "failure";

	public ActionForward all(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)throws
Exception{

			DatabaseManager dbm = DatabaseManager.getInstance();
			ArrayList<Materia> resultadoConsulta = dbm.consultarMaterias();
			if (resultadoConsulta.isEmpty()){

				//Si retorna un arreglo vacío, falla la consulta
				return mapping.findForward(FAILURE);
			}//de lo contrario, ocurre un éxito
			//Este comando envía a la vista el arreglo de materias
			 //podra ser accesado a traves del atributo “materias”
			request.setAttribute("materias",resultadoConsulta);
			return mapping.findForward(SUCCESS);
		}
	}
```

StrutsConfig

El archivo strutsconfig.xml se encarga de crear la correspondencia de las acciones con las vistas. El esqueleto de un strutsconfig.xml sería de la siguiente manera.

```
//También deben crearse form beans por cada formulario que se cree en la
//aplicacion
<struts-config>
      <form-beans>
            <form-bean name="Materia" type="clases.Materia"/>
      </form-beans>

/*Mapea las acciones a las vistas. Es decir, integra las capas controlador y vista
 * input es la página web desde donde se llama el action. Name es el nombre logico
 * de la redireccion. Type es la acción que mapearemos, se encuentra en el paquete
 * consultas y se llama VerMaterias.java. Es una acción de tipo request ya que
 * pide información (materias que se quieren ver
 * el path es el nombre relativo al contexto al cual se mapea la redirección.
 * Luego, hacemos los “forward” (redirecciones) dependiendo de si el
 * Action retornó mapping.findForward(“success”) (exito) o fallo
 * “failure”. Si la consulta es exitosa, ir a la página verConsultas.jsp
 * que muestra las consultas, de lo contrario se va a otra pagina
 *llamada error.jsp. Ambas paginas deben existir en la vista para poder
 *redireccionar a ellas.
 */
      <action-mappings>
            <action input="/index.jsp" name="Materia"
                  parameter="method" path="/verMaterias" scope="request"
                  validate="false" type="consultas.VerMaterias">
                  <forward name="success" path="/verConsultas.jsp"/>
                  <forward name="failure" path="/error.jsp"/>
            </action>

      </action-mappings>

</struts-config>
```

Para más información sobre strutsconfig puede revisar el siguiente enlace: http://www.webnms.com/webnms/help/developer_guide/web_client/web_struts_config.html

Vista

La vista esta compuesta por las páginas .jsp que el usuario accesa. Ahora, presentamos un esqueleto básico de una vista:

```
//Librerias de etiquetas a utilizar
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>

<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Encuesta de Opini&oacuten Estudiantil</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
//Se importa la hoja de estilos styles.css
<link rel="stylesheet" href="<html:rewrite page='/styles.css'/>" type="text/css"
media="screen" />
</head>
<body>
<div id="content">
<!-- header begins -->
<div id="header">
      <div id="logo">

      </div>

</div>
<!-- header ends -->
<!-- content begins -->
<div id="main">
<div id="mainbot">
<div id="maintop"></div>
      <div id="right" class = "borders">
                  <h5><b>Bienvenidos al Servicio de Consultas de Opinion
Estudiantil </b></h5>

<p align="justify">Esta pagina esta diseñada con el fin de ayudar a los
estudiantes y profesores a obtener informacion sobre la Encuesta de Opini&oacuten
Estudiantil realizada
a los alumnos de la universidad Simon Bolivar.<br></p>

<p align="justify">Esta informacion permitira ayudar al usuario a tomar decisiones
sobre las materias o profesores de la universidad.</p>
<br><br>
                        <div class="read"><a href="#">Leer Mas</a></div>
                  <br><br>

              <p align="justify">

            <div id="col">
            <div id="leftcol">
              <br><br> </div>
            <div id="rightcol">
            <br><br></div>
            </div>
      </div>
```

```
      <div id="lefttop" >
      <div id="left" >

            <ul><b>
                        <li><b>Inicio</b></li>

                        <li ><a>Consultas</a>
                                                      <ul>
                                    <li><a href =
"Vista/verDedicacionDificultad.jsp" style =
"font-size:x-small;">Dedicacion/Dificultad</a></li>
                                    <li ><a href =
"Vista/verDedicacionCalificacion.jsp" style =
"font-size:x-small;">Dedicacion/Calificacion</font></a></li>
                                    <li><a href = "Vista/verProfesorDificultad.jsp"
style = "font-size:x-small;">Profesor/Dificultad</a></li>
                                </ul>
                                </li>


                        <li><a
href="Vista/configuracion.jsp">Configuraci&oacuten</a></li>
                        <br>
                        </b>
                  </ul>



      </div>
      </div><div style="clear:both"></div>
<!--content ends -->
<!--footer begins -->
      </div>
</div>

<div id="footer">
<p>Sede Sartenejas, Baruta, Edo. Miranda - Apartado 89000 - Cable Unibolivar -
Caracas Venezuela. Telefono +58 0212-9063111
Sede Litoral, Camuri Grande, Edo. Vargas Parroquia Naiguata. Telefono +58
0212-9069000
</p>

      </div>
</div>
<!-- footer ends-->

</body>
</html>
```