Generar Temas de Colores y Cambiarlos Dinámicamente en ZK

Hola, mi nombre es Manuel Martínez y soy de Colombia. Voy a explicarles desde cómo construir temas de colores para las interfaces graficas de ZK hasta cambiarlas dinámicamente en la aplicación que estén desarrollando. Al finalizar se van a dar cuenta de lo fácil que es y lo agradable que es experimentar con colores distintos a los que provee el Framework sin usar hojas de estilo.

Me puse en la tarea de realizar este tutorial debido a que demore mucho buscando información de ¿Cómo se hacía esto? Y la verdad no lo encontré en ningún sitio Web, lo más que encontré fue lo de generación del tema, pero siempre lo hacían con un archivo .bat, lo cual me pareció innecesario pues la misma librería nos provee como crearlos. Lo otro que encontré en la web fue como colocar un solo color, el problema era que para cambiarlo siempre había que bajar el servidor y cambiar la librería, cosa que me pareció engorrosa.

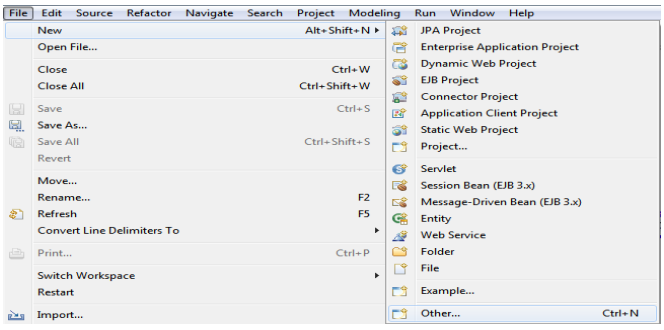
A continuación les voy a decir con qué herramientas trabaje para el ejemplo.

- Eclipse Helios IDE.
- ZK Plugin para Eclipse.
- Librería ZK Themer.
- Y como conocimiento básico, instalar el plugin de ZK en eclipse.

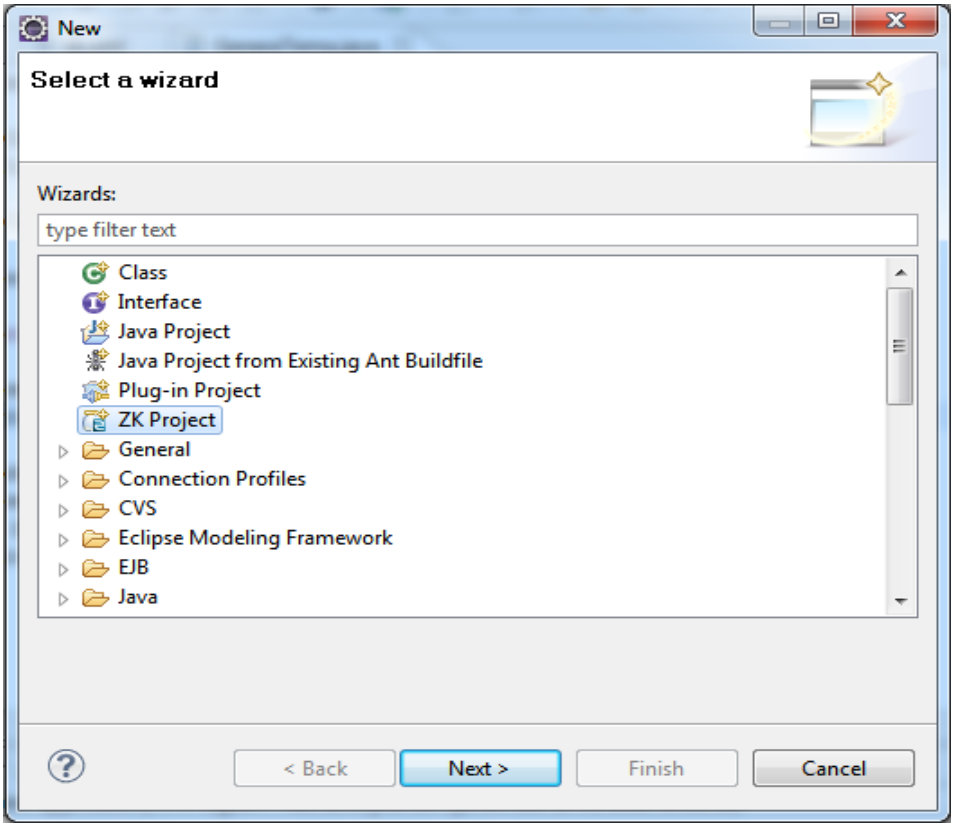
Ahora a crear y cambiar temas con ZK, existen varias formas, pero esta es la única que no encuentro en la Web.

Crear Temas.

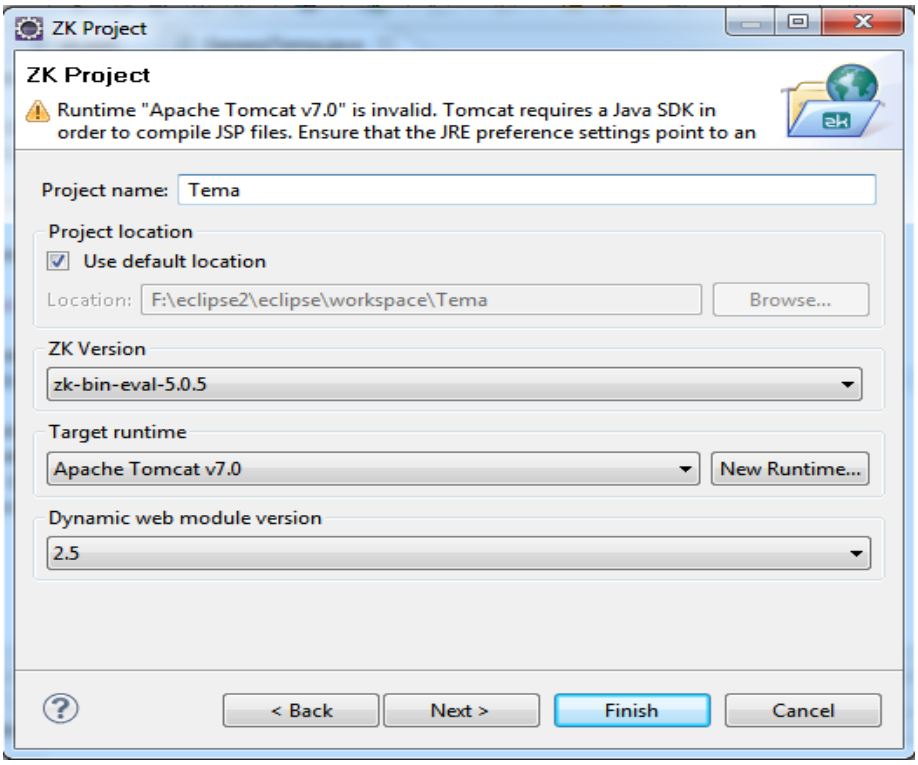
Lo primero es crear un proyecto ZK para eso vamos a File/New/Other



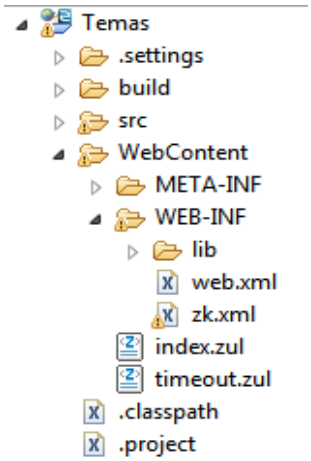
Aparecerá la siguiente Ventana y seleccionamos ZK Project



Clic en Next y configuramos el nombre del proyecto, la versión de ZK, en mi caso la 5.0.5 en evaluación y el servidor en el que va a correr, en mi caso Apache Tomcat 7.0.



Por ultimo Finish y el resultado será un proyecto con la siguiente estructura



El siguiente paso es tomar la librería ZK Themer y la pegamos en la carpeta lib. Para aquellos que no saben dónde conseguir la librería esta se encuentra en:

http://sourceforge.net/projects/zkforge/files/

Escoger colores:

Antes de seguir explicando hay que seleccionar el juego de colores que quieren para su aplicación. Para eso tenemos la gama de colores que nos ofrece java con la clase Color.

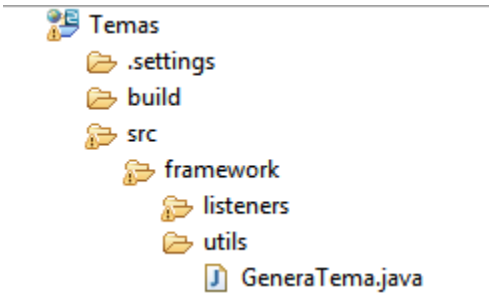
Para el ejemplo utilice:

- Color.BLACK
- Color.GREEN
- Color.RED

Para crear los colores, lo siguiente es usar los .jar que están en la carpeta lib. Yo decidí copiarlos a un directorio más accesible, pero se puede dejar en el mismo que se encuentra.

De la carpeta lib, vamos a copiar todos los .jar y los colocamos en un directorio dentro del equipo. Ojo, traten de no dejar ningún .jar por fuera, pues como resultado es posible que no encuentre alguno de los componentes y no le cambie el color.

Ahora, dentro de nuestro proyecto creamos una clase a la que le llamaremos GenaraTema



Para la clase hacemos un método main, pues esta no necesitamos llevarla a la Web, y le agregamos el contenido a continuación.

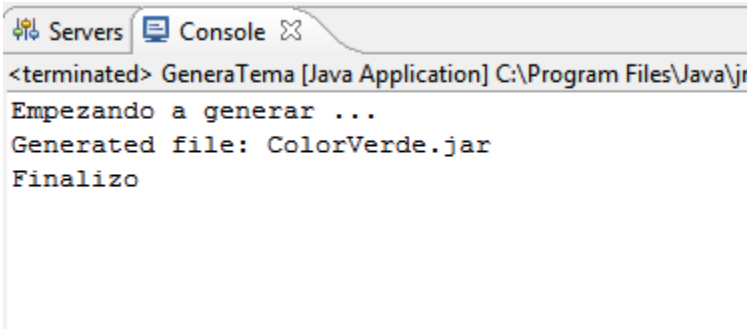
```
package framework.utils;

import java.awt.Color;
import java.io.File;

import zkthemer.CreateTheme;

public class GeneraTema {

    public static void main(String... arg) {
        try {
            System.out.println("Empezando a generar ...");
            CreateTheme creador = new CreateTheme("ColorVerde", Color.green,
                    new File("C:/zk5jars"), "n");
            creador.run();
            System.out.println("Finalizo");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

La clase CreateTheme se encuentra dentro de las librerías del ZK Themer, a partir de esta se desarrollan los .jar que al final son los colores de nuestra aplicación.

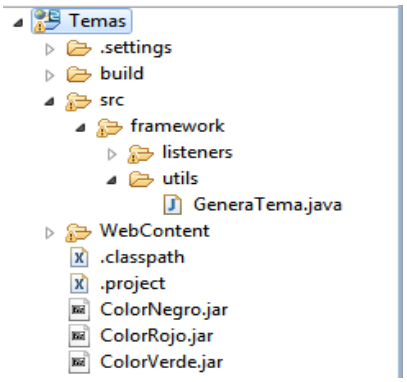
Como pueden apreciar el constructor de CreateTheme recibe 4 parámetros:

- El primero es el nombre con el que vamos a identificar nuestro .jar
- El segundo es el color que necesitamos, para esto la gama de colores de la clase Color.
- El tercer parámetro es un objeto File, con la dirección del directorio donde se encuentran los .jar
- Y el cuatro para pide si ignorar los archivos, hay le pasamos n.

Lo siguiente es correr esta clase como una aplicación java y esta debe arrojar un resultado como este en la consola.



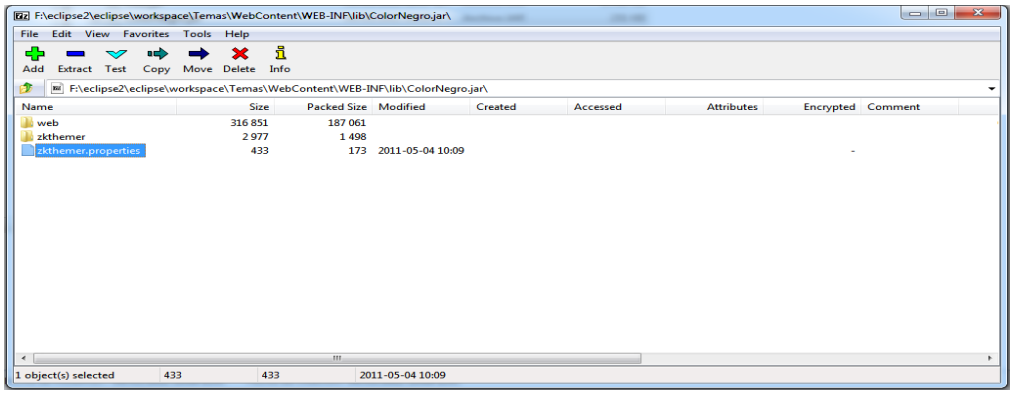
En caso de no ser así, eso significa que se ha hecho algo mal. Al generar, los .jar se crean en la carpeta de su proyecto, lo único que hay que hacer es moverlos de ahí a la carpeta lib.

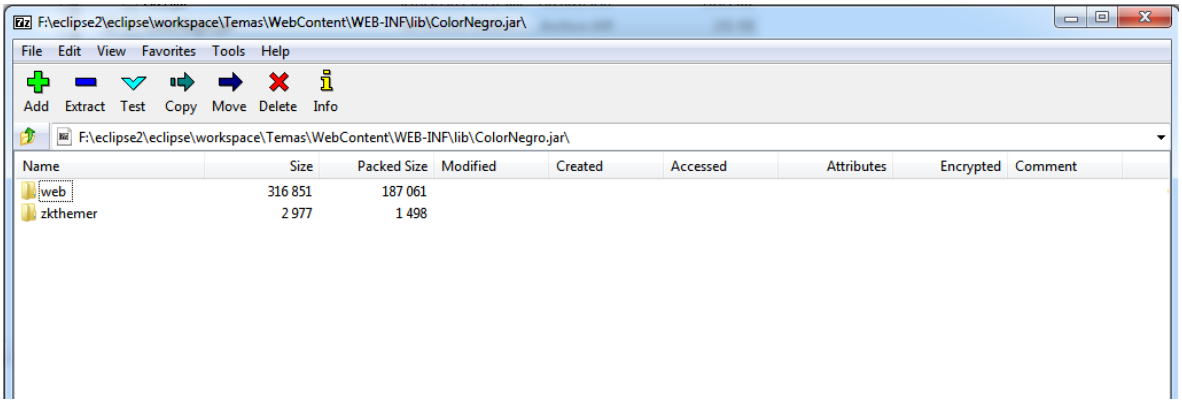


Configurar los .jar:

Al crearse todos los .jar, vienen con un archivo zkthemer.properties dentro de sí, la realidad es que solo necesitamos este archivo en uno solo, de lo contrario la aplicación se volverá loca y no sabrá por donde comenzar.

Para esto abrimos todos menos 1 de los jar generados y les borramos el archivo zkthemer.properties. Para abrirlos podemos usar cualquier programa comprensor como WinZip, WinRar, etc. yo utilice 7zip.

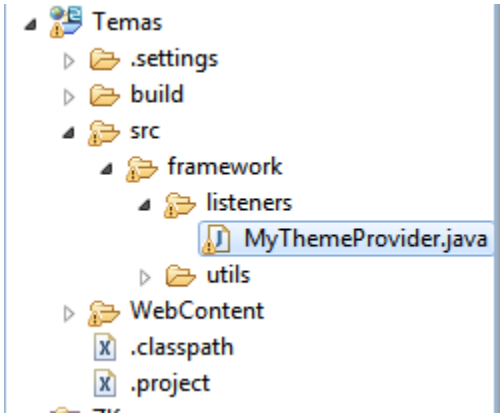




Proveedor de temas y la aplicación con nuestros temas.

Creando el Proveedor de temas de la aplicación:

Después de borrado el archivo creamos nuestro propio proveedor de temas. Aproveche una clase muy popular que hay en la Web llamada ThemeProvider y lo único que hice fue agregarle 2 métodos adicionales. A continuación muestro donde ubique la clase, y su contenido.




```
/*
 This program is free software; you can redistribute it and/or modify
 it under the terms of the GNU Lesser General Public License as
 published by the Free Software Foundation; either version 2 of the
 License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful, but
 WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 Lesser General Public License for more details.

 You should have received a copy of the GNU Lesser General Public
 License along with this program; if not, write to the Free Software
 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
 USA */

package framework.listeners;

import java.io.*;
import java.util.*;
import org.zkoss.zk.ui.Execution;
import org.zkoss.zk.ui.util.ThemeProvider;

public class MyThemeProvider implements ThemeProvider {

	private String themeName, fileList;

	public String getThemeName() {
		return themeName;
	}

	public void setThemeName(String themeName) {
		this.themeName = themeName;
	}

	public MyThemeProvider() {
		try {
			InputStream is = getClass().getResourceAsStream(
					"/zkthemer.properties");
			if (is == null)
				throw new RuntimeException("Cannot find zkthemer.properties");

			Properties prop = new Properties();
			prop.load(is);
			themeName = (String) prop.get("theme");
			fileList = (String) prop.get("fileList");

			if (themeName == null) {
				throw new RuntimeException(
						"zkthemer.properties found, but missing 'theme'
entry");
			}
			is.close();

		} catch (IOException e) {
			throw new RuntimeException(e);
		}
	}

	@SuppressWarnings("unchecked")
	public Collection getThemeURIs(Execution exec, List uris) {

		List newUris = new ArrayList(uris);
```

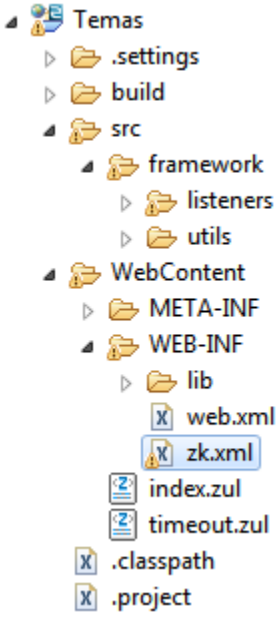
```
                for (Object object : newUris) {
                        String uri = (String) object;
                        if (uri.startsWith("~./")) {
                                uri = "~./" + themeName + "/" + uri.substring(3);
                        }
                        uris.add(uri);
                }
                return uris;
        }

        public String beforeWCS(Execution exec, String uri) {
                return uri;
        }

        public String beforeWidgetCSS(Execution exec, String uri) {
                String fileName = uri.substring(uri.lastIndexOf("/") + 1);

                if (!(fileList.indexOf(fileName) < 0))
                        uri = (new StringBuilder("~./")).append(themeName).append("/")
                                        .append(uri.substring(3)).toString();
                return uri;
        }

        public int getWCSCacheControl(Execution exec, String uri) {
                return -1;// safe to cache
        }
}
```

En este caso renombre la clase a MyThemeProvider y a ella le adicione estos dos métodos

```
    public String getThemeName() {
        return themeName;
    }

    public void setThemeName(String themeName) {
        this.themeName = themeName;
    }
```

Registrar el proveedor de temas a la aplicación:

Ahora registramos nuestro proveedor de temas al proyecto. Para eso nos vamos al archivo zk.xml



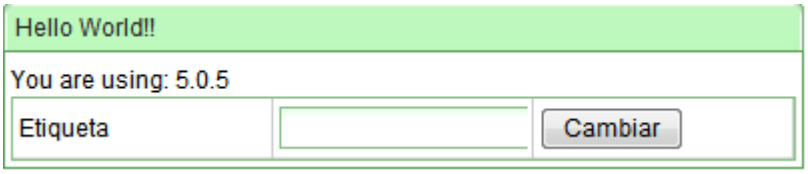
Y le agregamos el siguiente contenido:

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

<zk>
    <!-- Configuración del MyThemeprovider -->
    <desktop-config>
        <theme-provider-class>framework.listeners.MyThemeProvider</theme-provider-class>
    </desktop-config>
</zk>
```

Si corremos ahora el proyecto, por defecto pinta el color al cual le dejaron el archivo zkthemer.properties, en mi caso el negro.

Corriendo los temas:

Se preguntaran donde esta lo dinámico, pues para que sea dinámico eso va a consideración de quien lo use donde colocarlo, sin embargo yo dejo las líneas de código que lo hacen y los ejemplos.



Y estas son las líneas que puse en el evento onClick del botón cambiar

```
public void cambiar() {
    ((MyThemeProvider) Executions.getCurrent().getDesktop()
            .getWebApp().getConfiguration().getThemeProvider())
            .setThemeName("ColorRojo");

    Executions.sendRedirect(null);
}
```

Explico que significa esas líneas:

- Primero, capturo la ejecución actual de la aplicación.
- De ella extraigo si Desktop.
- Del Desktop capturo la aplicación Web.
- De la aplicación saco la configuración, esa que está en el archivo zk.xml.
- De la configuración tomo el proveedor de temas que le configure al arranque.
- Eso me devuelve un objeto de tipo ThemeProvider, a ese objeto le hago un casting a MyThemeProvider, ya que hay tengo los métodos setThemeName y getThemeName.
- Al resultado del casting, llamo al método setThemeName y como parámetro le envío el nombre de uno de los colores que antes había creado. Estos nombres los pueden tener en la DB, en un archivo de configuración o incluso en la misma aplicación si quieren.
- Por último la línea Executions.sendRedirect (null); esta línea me refresca la aplicación para que tome los cambios.

Bueno eso es todo, espero haya servido de ayuda este tutorial, y estén a la espera de próximas entradas.