MotorGenerador

```
/**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* $Id: MotorGenerador.java,v 1.1 2009/01/30 21:50:02 sil-de Exp $
* Universidad de los Andes (Bogotá - Colombia)
* Departamento de Ingeniería de Sistemas y Computación
* Licenciado bajo el esquema Academic Free License version 2.1
*
* Proyecto Cupi3 (http://cupi2.uniandes.edu.co)
* Ejercicio: Tutorial Generación Código (n10_paint)
* Autor: n-calder
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
package uniandes.cupi2.paint.velocity.mundo;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import org.apache.velocity.Template;
import org.apache.velocity.VelocityContext;
import org.apache.velocity.app.Velocity;
import org.apache.velocity.exception.MethodInvocationException;
import org.apache.velocity.exception.ParseErrorException;
import org.apache.velocity.exception.ResourceNotFoundException;

/**
* Generador de código Velocity para Paint
*/
public class MotorGenerador {

//-----------------------------------------------------------------
// Constantes
// -----------------------------------------------------------------

/**
* Directorio en donde se guarda el código generado
*/
private static final String DIR_GENERADO = "./generador/archivos/";

/**
* Plantilla que se utiliza para la generación
*/
private static final String PLANTILLA = "./generador/plantillas/template.vm";

//-----------------------------------------------------------------
// Constructores
// -----------------------------------------------------------------

/**
*  Constructor de la clase
*/
public MotorGenerador()
{

}

//-----------------------------------------------------------------
// Métodos
// -----------------------------------------------------------------


/**
* Este método prepara el mapa de propiedades con los que se construirá la nueva clase
* y ejecuta el proceso de generación con la máquina de velocity.
* @param nombreClase Nombre de la nueva clase
* @param nombreSuper Nombre de la superclase que se quiere tomar como prototipo
*/
public void generar(String nombreClase, String nombreSuper) throws GeneradorCodigoException
{

try {
Class prototipo = Class.forName(nombreSuper);
```

```
Map propiedades = new HashMap();
propiedades.put("paquete",prototipo.getPackage().getName());
propiedades.put("clase",nombreClase);
propiedades.put("superClase",prototipo);

Field[] atributos = prototipo.getDeclaredFields();
ArrayList<Field> seleccionados = new ArrayList<Field>();
for (int i = 0; i < atributos.length; i++) {
int modifier = atributos[i].getModifiers();
if(Modifier.isProtected(modifier))
{
seleccionados.add(atributos[i]);
}
}
propiedades.put("atributos",seleccionados);
propiedades.put("constructores",prototipo.getConstructors());

Method[] metodos = prototipo.getDeclaredMethods();
ArrayList<Method> abstractos = new ArrayList<Method>();
for (int i = 0; i < metodos.length; i++) {
int modifier = metodos[i].getModifiers();
if(Modifier.isAbstract(modifier))
{
abstractos.add(metodos[i]);
}
}
propiedades.put("metodos",abstractos );

Velocity.init();
VelocityContext context = new VelocityContext(propiedades);
Template template = null;
try {
template = Velocity.getTemplate(PLANTILLA);
}
catch (ResourceNotFoundException rnfe)
{
throw new GeneradorCodigoException("Imposible localizar plantilla: " + PLANTILLA +". " +
rnfe.getMessage(), rnfe);
}
catch (Exception e)
{
throw new GeneradorCodigoException("Error de sintaxis en plantilla. "+
e.getMessage(), e);
}

File generado = new File(DIR_GENERADO + nombreClase + ".java");
BufferedWriter out = new BufferedWriter(new FileWriter(generado, false));

if (template != null)
template.merge(context, out);
out.close();

}
catch (ResourceNotFoundException e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (ParseErrorException e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (MethodInvocationException e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (SecurityException e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (ClassNotFoundException e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (IOException e)
{
```

```
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
catch (Exception e)
{
e.printStackTrace();
throw new GeneradorCodigoException(e.getMessage(), e);
}
}

}
```