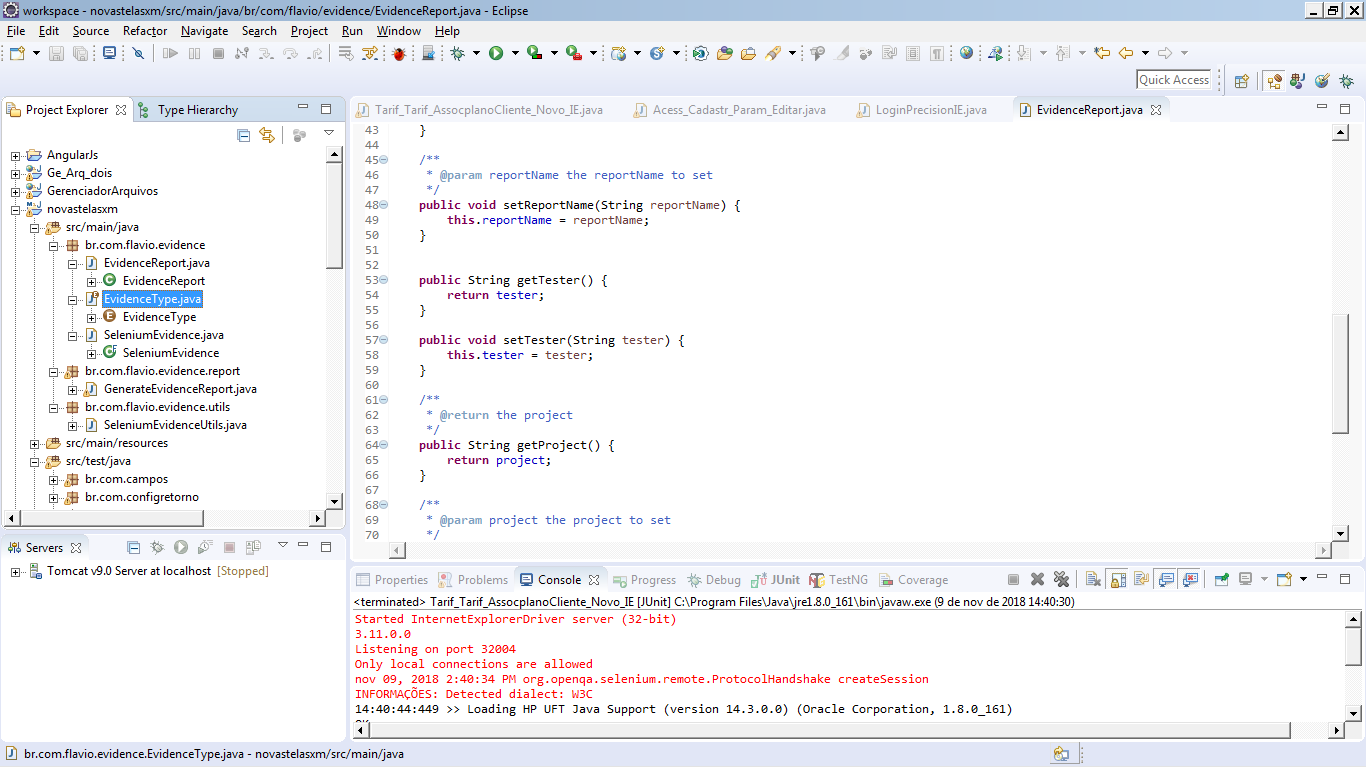
Selenium Evidence



Classe evidence Report.java

package br.com.flavio.evidence;

import java.util.List;

import br.com.flavio.evidence.SeleniumEvidence;

public class EvidenceReport {

private List<SeleniumEvidence> evidenceList = null;

private String reportName = "TVA\_XM";

private String tester = "Flavio Silva";

private String project = "Unifica TVAS\_XM";

private String exceptionString = null;

public EvidenceReport(List<SeleniumEvidence> evidenceList, String reportName, String tester,

String project, String exceptionString) {

this.evidenceList = evidenceList;

this.reportName = reportName;

this.tester = tester;

this.project = project;

this.exceptionString = exceptionString;

}

/\*\*

\* @return the evidenceList

\*/

public List<SeleniumEvidence> getEvidenceList() {

return evidenceList;

}

/\*\*

\* @param evidenceList the evidenceList to set

\*/

public void setEvidenceList(List<SeleniumEvidence> evidenceList) {

this.evidenceList = evidenceList;

}

/\*\*

\* @return the reportName

\*/

public String getReportName() {

return reportName;

}

/\*\*

\* @param reportName the reportName to set

\*/

public void setReportName(String reportName) {

this.reportName = reportName;

}

public String getTester() {

return tester;

}

public void setTester(String tester) {

this.tester = tester;

}

/\*\*

\* @return the project

\*/

public String getProject() {

return project;

}

/\*\*

\* @param project the project to set

\*/

public void setProject(String project) {

this.project = project;

}

/\*\*

\* @return the exceptionString

\*/

public String getExceptionString() {

return exceptionString;

}

/\*\*

\* @param exceptionString the exceptionString to set

\*/

public void setExceptionString(String exceptionString) {

this.exceptionString = exceptionString;

}

}

Classe Evidence TYPE

**package** br.com.flavio.evidence;

**public** **enum** EvidenceType {

***HTML***, ***DOC***, ***PDF***

}

Classe Selenium Evidence.java

package br.com.flavio.evidence;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import javax.imageio.ImageIO;

import org.apache.commons.codec.binary.Base64;

/\*\*

\* Bean to store evidence message and image in a BASE64Decoder

\*

\* @author

\*/

public final class SeleniumEvidence {

/\*\*

\* Evidence message

\*/

private String message;

/\*\*

\* String returned by Selenium

\*/

private String imageString;

/\*\*

\* Image to use in evidence report

\*/

private BufferedImage image;

/\*\*

\* Constructor to create a new instance of the evidence data

\* @param message

\* @param imageString

\* @throws Exception

\*/

public SeleniumEvidence(String message, String imageString) throws Exception {

setMessage(message);

setImageString(imageString);

}

/\*\*

\* Get the evidence message

\* @return the message

\*/

public String getMessage() {

return message;

}

/\*\*

\* Set the evidence message

\* @param message the message to set

\*/

public void setMessage(String message) {

this.message = message;

}

/\*\*

\* Get the image

\* @return the image

\*/

public BufferedImage getImage() {

return image;

}

/\*\*

\* Set the image

\* @param image the sel\_image to set

\*/

public void setImage(BufferedImage image) {

this.image = image;

}

/\*\*

\* Get the image in BASE64Decoder

\* @return the imageString

\*/

public String getImageStringg() {

return imageString;

}

/\*\*

\* Creates a ImageIO based on string coming to Selenium

\* @param sel\_image\_string the sel\_image\_string to set

\* @throws IOException if occurs any problem with the directory

\* @throws Exception if occurs any other problem with the code

\*/

public void setImageString(String imageString) throws IOException, Exception {

this.imageString = imageString;

setImage(ImageIO.read(new ByteArrayInputStream(toImage(imageString))));

}

/\*\*

\* Transform the string returned in BASE64Decoder

\* @param string string returned by Selenium

\* @return an instance of BASE64Decoder

\* @throws Exception if the string is malformed or null

\*/

public static byte[] toImage(String string) throws Exception {

return Base64.decodeBase64(string);

}

}

Classe Generate evidence

package br.com.flavio.evidence.report;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Properties;

import javax.imageio.ImageIO;

import br.com.flavio.evidence.EvidenceReport;

import br.com.flavio.evidence.EvidenceType;

import br.com.flavio.evidence.SeleniumEvidence;

import br.com.flavio.evidence.utils.SeleniumEvidenceUtils;

import net.\*;

import net.sf.jasperreports.engine.JRException;

import net.sf.jasperreports.engine.JasperCompileManager;

import net.sf.jasperreports.engine.JasperExportManager;

import net.sf.jasperreports.engine.JasperFillManager;

import net.sf.jasperreports.engine.JasperPrint;

import net.sf.jasperreports.engine.JasperReport;

import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

import net.sf.jasperreports.engine.export.ooxml.JRDocxExporter;

import net.sf.jasperreports.engine.export.ooxml.JRDocxExporterParameter;

/\*\*

\* Generate the test evidence in PDF file

\* @author Elias Nogueira <elias.nogueira@gmail.com>

\*/

public class GenerateEvidenceReport {

/\*\*

\* Generate evidence in PDF file. The evidence will be save on home dir on a folder called by the name of the project

\* @param list list of evidence made of an text and image

\* @param reportName report name

\* @param exception exception text that are throwing by Java

\* @throws IOException if occurs any problem with the directory

\*/

@Deprecated

public static void generatePDFEvidence(List<SeleniumEvidence> list, String reportName, String tester, String project, String exception) throws IOException {

List<SeleniumEvidence> data = list;

Properties properties = SeleniumEvidenceUtils.loadProperties();

String evidenceDir = System.getProperty("user.dir") + System.getProperty("file.separator") +

properties.getProperty("evidence.dir") + System.getProperty("file.separator");

createEvidenceDir(evidenceDir);

try {

String companyImage = properties.getProperty("image.company.path");

String customerImage = properties.getProperty("image.customer.path");

BufferedImage imageCompany;

BufferedImage imageClient;

if (companyImage == null || companyImage.equals("null")) {

imageCompany = null;

} else {

imageCompany = ImageIO.read(new File(companyImage));

}

if (customerImage == null || customerImage.equals("null")) {

imageClient = null;

} else {

imageClient = ImageIO.read(new File(customerImage));

}

if (reportName == null) {

reportName = "";

}

if (tester == null) {

tester = "";

}

if (project == null) {

project = "";

}

if (exception == null) {

exception = "";

}

Map<String, Object> parameters = new HashMap<String, Object>();

if (exception != null) {

parameters.put("SEL\_EXCEPTION", exception);

}

parameters.put("SEL\_COMPANY\_LOGO", imageCompany);

parameters.put("SEL\_CUSTOMER\_LOGO", imageClient);

parameters.put("SEL\_PROJECT", project);

parameters.put("SEL\_TESTER", tester);

parameters.put("SEL\_LABEL\_EVINDENCE\_TITLE", properties.getProperty("label.evidenceReport"));

parameters.put("SEL\_LABEL\_PROJECT", properties.getProperty("label.projetc"));

parameters.put("SEL\_LABEL\_TESTER", properties.getProperty("label.tester"));

//parameters.put("SEL\_LABEL\_STATUS", properties.getProperty("label.status"));

//parameters.put("SEL\_LABEL\_PASS", properties.getProperty("label.status.pass"));

//parameters.put("SEL\_LABEL\_FAILED", properties.getProperty("label.status.failed"));

parameters.put("SEL\_LABEL\_EVIDENCE\_REPORT", properties.getProperty("label.evidenceReport"));

parameters.put("SEL\_LABEL\_DATE", properties.getProperty("label.date"));

parameters.put("SEL\_LABEL\_FOOTER", properties.getProperty("label.footer"));

parameters.put("SEL\_LABEL\_ERROR\_DETAIL", properties.getProperty("label.errorDetail"));

parameters.put("SEL\_LABEL\_PAGE", properties.getProperty("label.page"));

parameters.put("SEL\_LABEL\_COMPANY\_NAME", properties.getProperty("label.company.name"));

JRBeanCollectionDataSource datasource = new JRBeanCollectionDataSource(data);

JasperReport jasperReport = JasperCompileManager.compileReport(properties.getProperty("evidence.file"));

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters, datasource);

JasperExportManager.exportReportToPdfFile(jasperPrint, evidenceDir + reportName + ".pdf");

} catch (SecurityException ex) {

ex.printStackTrace();

} catch (JRException jre) {

jre.printStackTrace();

}

}

/\*\*

\* Generate an evidence report based on EvidenceType (DOC, PDF, HTML)

\* @param evidenceReport and EvidenceReport object with basic informations

\* @param reportType filetype

\* @throws IOException if any problem with the files (jasper, EvidenceType) or directory occurs

\*/

public static void generareEvidenceReport(EvidenceReport evidenceReport, EvidenceType reportType) throws IOException {

List<SeleniumEvidence> data = evidenceReport.getEvidenceList();

Properties properties = SeleniumEvidenceUtils.loadProperties();

String evidenceDir = System.getProperty("user.dir") + System.getProperty("file.separator") +

properties.getProperty("evidence.dir") + System.getProperty("file.separator");

createEvidenceDir(evidenceDir);

try {

String companyImage = properties.getProperty("image.company.path");

String customerImage = properties.getProperty("image.customer.path");

BufferedImage imageCompany;

BufferedImage imageClient;

if (companyImage == null || companyImage.equals("null")) {

imageCompany = null;

} else {

imageCompany = ImageIO.read(new File(companyImage));

}

if (customerImage == null || customerImage.equals("null")) {

imageClient = null;

} else {

imageClient = ImageIO.read(new File(customerImage));

}

String reportName = evidenceReport.getReportName();

if (reportName == null) {

reportName = "";

}

String tester = evidenceReport.getTester();

if (tester == null) {

tester = "";

}

String project = evidenceReport.getProject();

if (project == null) {

project = "";

}

String exception = evidenceReport.getExceptionString();

if (exception == null) {

exception = "";

}

Map<String, Object> parameters = new HashMap<String, Object>();

if (exception != null) {

parameters.put("SEL\_EXCEPTION", exception);

}

parameters.put("SEL\_COMPANY\_LOGO", imageCompany);

parameters.put("SEL\_CUSTOMER\_LOGO", imageClient);

parameters.put("SEL\_PROJECT", project);

parameters.put("SEL\_TESTER", tester);

parameters.put("SEL\_LABEL\_EVINDENCE\_TITLE", properties.getProperty("label.evidenceReport"));

parameters.put("SEL\_LABEL\_PROJECT", properties.getProperty("label.projetc"));

parameters.put("SEL\_LABEL\_TESTER", properties.getProperty("label.tester"));

//parameters.put("SEL\_LABEL\_STATUS", properties.getProperty("label.status"));

parameters.put("SEL\_LABEL\_PASS", properties.getProperty("label.status.pass"));

parameters.put("SEL\_LABEL\_FAILED", properties.getProperty("label.status.failed"));

parameters.put("SEL\_LABEL\_EVIDENCE\_REPORT", properties.getProperty("label.evidenceReport"));

parameters.put("SEL\_LABEL\_DATE", properties.getProperty("label.date"));

parameters.put("SEL\_LABEL\_FOOTER", properties.getProperty("label.footer"));

parameters.put("SEL\_LABEL\_ERROR\_DETAIL", properties.getProperty("label.errorDetail"));

parameters.put("SEL\_LABEL\_PAGE", properties.getProperty("label.page"));

parameters.put("SEL\_LABEL\_COMPANY\_NAME", properties.getProperty("label.company.name"));

JRBeanCollectionDataSource datasource = new JRBeanCollectionDataSource(data);

JasperPrint jasperPrint = JasperFillManager.fillReport(properties.getProperty("evidence.file"), parameters, datasource);

switch (reportType) {

case PDF:

JasperExportManager.exportReportToPdfFile(jasperPrint, evidenceDir + reportName + ".pdf");

break;

case DOC:

JRDocxExporter exporter = new JRDocxExporter();

File archivo = new File(evidenceDir + reportName + ".doc");

FileOutputStream os = new FileOutputStream(archivo);

exporter.setParameter(JRDocxExporterParameter.JASPER\_PRINT, jasperPrint);

exporter.setParameter(JRDocxExporterParameter.CHARACTER\_ENCODING, "UTF-8");

exporter.setParameter(JRDocxExporterParameter.OUTPUT\_STREAM, os);

exporter.exportReport();

os.close();

break;

case HTML:

JasperExportManager.exportReportToHtmlFile(jasperPrint, evidenceDir + reportName + ".html");

break;

default:

break;

}

} catch (SecurityException ex) {

ex.printStackTrace();

} catch (JRException jre) {

jre.printStackTrace();

}

}

/\*\*

\* Create a directory with the project's name

\* @param project project name

\*/

private static boolean createEvidenceDir(String directory) {

boolean dirExists = false;

try {

File dir = new File(directory);

boolean exists = dir.exists();

if (!exists) {

dir.mkdir();

dirExists = false;

} else {

dirExists = true;

}

} catch (SecurityException se) {

se.printStackTrace();

}

return dirExists;

}

}

Classe evidence UTILS

package br.com.flavio.evidence.utils;

import java.io.FileInputStream;

import java.io.IOException;

import java.util.Properties;

/\*\*

\* Class with shared methods and action to Selenium Evidence

\* @author Elias Nogueira <elias.nogueira@gmail.com>

\*/

public class SeleniumEvidenceUtils {

/\*\*

\* Properties that loads report parameterization

\*/

static Properties properties;

/\*\*

\* Load Selenium Evidence property file

\* @return a property

\* @throws IOException if file do not exists

\*/

public static Properties loadProperties() throws IOException {

properties = new Properties();

properties.load(new FileInputStream("init.properties"));

return properties;

}

}

Bibliotecas

