文件 Action.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CaseManager

{

public class Action

{

public int index;

public bool isRun;

public Action(int index, bool isRun)

{

this.index = index;

this.isRun = isRun;

}

}

public class NullAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add( this.index.ToString() );

list.Add("");

list.Add("");

list.Add("");

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String par1;

public String par2;

public NullAction(int index, String par1, String par2, bool isRun)

: base(index, isRun)

{

this.par1 = par1;

this.par2 = par2;

}

}

public class CreateIOSDriverAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("新建IOS驱动");

list.Add(this.macIP);

list.Add(this.appPath);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String macIP;

public String appPath;

public CreateIOSDriverAction(int index, String macIP, String appPath, bool isRun)

: base(index, isRun)

{

this.macIP = macIP;

this.appPath = appPath;

}

}

public class CreateAppiumDriverAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("新建Appium驱动");

list.Add(this.deviceName);

list.Add(this.appPackageAndAppActivity);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String deviceName;

public String appPackageAndAppActivity;

public CreateAppiumDriverAction(int index, String deviceName, String appPackageAndAppActivity, bool isRun)

: base(index, isRun)

{

this.deviceName = deviceName;

this.appPackageAndAppActivity = appPackageAndAppActivity;

}

}

public class CreateDriverAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("新建驱动");

list.Add(browserName);

list.Add(driverName);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String browserName;

public String driverName;

public CreateDriverAction(int index, String browserName, String driverName, bool isRun)

: base(index, isRun)

{

this.browserName = browserName;

this.driverName = driverName;

}

}

class GoAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("打开网址");

list.Add(driverName);

list.Add(url);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String driverName;

public String url;

public GoAction(int index,String driverName, String url, bool isRun)

: base(index, isRun)

{

this.driverName = driverName;

this.url = url;

}

}

class SetParameterAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("设置参数");

list.Add(parName);

list.Add(parValue);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String parName;

public String parValue;

public SetParameterAction(int index, String parName, String parValue, bool isRun)

: base(index, isRun)

{

this.parName = parName;

this.parValue = parValue;

}

}

class ExecuteFunctionAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("执行方法");

list.Add(driverName);

list.Add(functionName);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String driverName;

public String functionName;

public ExecuteFunctionAction(int index, String driverName, String functionName, bool isRun)

: base(index, isRun)

{

this.driverName = driverName;

this.functionName = functionName;

}

}

public class OneStepAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("单步方法");

list.Add(oneStepName);

list.Add(oneStepPars);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String oneStepName;

public String oneStepPars;

public OneStepAction(int index, String oneStepName, String oneStepPars, bool isRun)

: base(index, isRun)

{

this.oneStepName = oneStepName;

this.oneStepPars = oneStepPars;

}

}

class SleepAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("暂停");

list.Add(timeLong.ToString());

list.Add("");

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public int timeLong;

public SleepAction(int index, int timeLong, bool isRun)

: base(index, isRun)

{

this.timeLong = timeLong;

}

}

class ExeOtherCaseAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("调用其他用例");

list.Add("");

list.Add(casePath);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String casePath;

public ExeOtherCaseAction(int index, String casePath, bool isRun)

: base(index, isRun)

{

this.casePath = casePath;

}

}

class AnnotationAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("注释");

list.Add("");

list.Add(annotationText);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String annotationText;

public AnnotationAction(int index, String annotationText, bool isRun)

: base(index, isRun)

{

this.annotationText = annotationText;

}

}

class PythonCodeAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("Python代码");

list.Add("");

list.Add(pcode);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String pcode;

public PythonCodeAction(int index, String pcode, bool isRun)

: base(index, isRun)

{

this.pcode = pcode;

}

}

//{

class AddIndentAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("{");

list.Add("");

list.Add("");

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public AddIndentAction(int index, bool isRun)

: base(index, isRun)

{

}

}

//}

class RecoverIndentAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("}");

list.Add("");

list.Add("");

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public RecoverIndentAction(int index, bool isRun)

: base(index, isRun)

{

}

}

class QuitAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("退出");

list.Add(driverName);

list.Add("");

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String driverName;

public QuitAction(int index, String driverName, bool isRun)

: base(index, isRun)

{

this.driverName = driverName;

}

}

class ReadExternalConfAction : Action

{

public List<String> toList()

{

List<String> list = new List<String>();

list.Add(this.index.ToString());

list.Add("读取外部配置");

list.Add("");

list.Add(excelPath);

if (isRun)

{

list.Add("是");

}

else

{

list.Add("否");

}

return list;

}

public String excelPath;

public ReadExternalConfAction( int index,String excelPath,bool isRun )

: base(index, isRun)

{

this.excelPath = excelPath;

}

}

}

文件 CaseCreater.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace CaseManager

{

class CaseCreater

{

public void createpy(String casefile, String templatefile)

{

String scriptfile = casefile.Replace(".py", "\_.py");

Dictionary<String, String> dic = getAllModuleStr(casefile);

replace(templatefile, scriptfile, dic);

}

public void replace(String templatefile, String tofile, Dictionary<String, String> dic)

{

StreamReader sr = new StreamReader(templatefile, Encoding.Default);

String tmp = sr.ReadToEnd();

sr.Close();

foreach (String key in dic.Keys)

{

tmp = tmp.Replace("$" + key + "$", dic[key]);

}

StreamWriter sw = new StreamWriter(tofile, false, Encoding.Default);

sw.WriteLine(tmp);

sw.Close();

}

//将一个用例文件中的各个部分取出来

public Dictionary<String, String> getAllModuleStr(String filepath)

{

Dictionary<String, String> dic = new Dictionary<string, string>();

StreamReader sr = new StreamReader(filepath, Encoding.Default);

String str = sr.ReadToEnd();

sr.Close();

String[] tmps = str.Split(new String[] { "def " }, StringSplitOptions.RemoveEmptyEntries );

foreach( String tmp in tmps )

{

String[] tmps2 = tmp.Split(new String[] { "():\r\n" }, StringSplitOptions.None);

String moduleName = tmps2[0];

String moduleStr = tmps2[1];

dic.Add(moduleName, moduleStr);

//////////Console.WriteLine();

}

return dic;

}

}

}

文件 CommandExecutor.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Diagnostics;

using System.Windows.Forms;

namespace CaseManager

{

class CommandExecutor

{

public void execute(String command)

{

ProcessStartInfo info = new ProcessStartInfo();

info.FileName = "cmd.exe";

info.RedirectStandardInput = true;

info.RedirectStandardOutput = true;

info.RedirectStandardError = true;

info.CreateNoWindow = true;

info.UseShellExecute = false;

Process p = Process.Start(info);

p.StandardInput.WriteLine(command + "&exit");

p.StandardInput.AutoFlush = true;

String oo = p.StandardOutput.ReadToEnd();

String ee = p.StandardError.ReadToEnd();

////Console.WriteLine(oo);

////Console.WriteLine(ee);

p.WaitForExit();

p.StandardOutput.Close();

p.StandardInput.Close();

p.Close();

}

public void execute2(String pyFile)

{

ProcessStartInfo info = new ProcessStartInfo();

info.RedirectStandardInput = true;

info.RedirectStandardOutput = true;

info.RedirectStandardError = true;

info.CreateNoWindow = true;

info.UseShellExecute = false;

info.FileName = "python.exe";

info.Arguments = pyFile;

Process p = Process.Start(info);

p.Start();

while (!p.StandardOutput.EndOfStream)

{

String tmp = p.StandardOutput.ReadLine();

////////Console.WriteLine(tmp);

}

p.StandardOutput.Close();

p.StandardInput.Close();

p.Close();

}

}

}

文件 DirectoryManager.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace CaseManager

{

class DirectoryManager

{

public void clearDir(String path)

{

DirectoryInfo direct = new DirectoryInfo(path);

FileInfo[] files = direct.GetFiles();

foreach (FileInfo fInfo in files)

{

fInfo.Delete();

}

}

public List<String> getAllFiles(String path ,String fileFormat)

{

getAllFile2(path ,fileFormat);

return list;

}

private List<String> list = new List<string>();

private void getAllFile2(String path ,String fileFormat)

{

DirectoryInfo direct = new DirectoryInfo(path);

DirectoryInfo[] dirs = direct.GetDirectories();

foreach (DirectoryInfo dir in dirs)

{

getAllFile2(dir.FullName ,fileFormat);

}

FileInfo[] files = direct.GetFiles("\*." + fileFormat);

foreach (FileInfo f in files)

{

list.Add(f.FullName);

}

}

}

}

文件 ExcelProcessor.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.IO;

using NPOI;

using NPOI.HPSF;

using NPOI.HSSF;

using NPOI.HSSF.UserModel;

using NPOI.POIFS;

using NPOI.Util;

using NPOI.SS.UserModel;

namespace CaseManager

{

class ExcelProcessor

{

public List<String> getsheets(String filename)

{

using (FileStream fs = File.OpenRead(filename)) //打开myxls.xls文件

{

NPOI.HSSF.UserModel.HSSFWorkbook wk = new HSSFWorkbook(fs); //把xls文件中的数据写入wk中

int n = wk.NumberOfSheets;

List<String> sheetnames = new List<string>();

for (int i = 0; i < n; i++)

{

sheetnames.Add(wk.GetSheetName(i));

}

return sheetnames;

}

}

public void writeToExcelFile( List<List<String>> listList ,String fileName )

{

using (FileStream fs = File.OpenWrite(fileName))

{

HSSFWorkbook workbook = new HSSFWorkbook();

ISheet iSheet = workbook.CreateSheet("testCase");

int j = 0;

foreach (List<String> list in listList)

{

int i = 0;

IRow row = iSheet.CreateRow(j);

foreach (String value in list)

{

row.CreateCell(i).SetCellValue(value);

i++;

}

j++;

}

workbook.Write(fs);

}

}

public List<List<String>> readfirststr(String filename)

{

using (FileStream fs = File.OpenRead(filename)) //打开myxls.xls文件

{

List<List<String>> listlist = new List<List<string>>(); //--------------------------------

NPOI.HSSF.UserModel.HSSFWorkbook wk = new HSSFWorkbook(fs); //把xls文件中的数据写入wk中

ISheet sheet = wk.GetSheetAt(0); //第一个sheet

if (null != sheet)

{

for (int j = 0; j <= sheet.LastRowNum; j++) //LastRowNum 是当前表的总行数

{

List<String> list = new List<string>(); //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IRow row = sheet.GetRow(j); //读取当前行数据

if (row != null)

{

for (int k = 0; k < row.LastCellNum; k++) //LastCellNum 是当前行的总列数

{

ICell cell = row.GetCell(k); //当前表格

if (cell != null)

{

list.Add(cell.ToString());

}

else

{

list.Add("");

}

}

}

listlist.Add(list);

}

this.plastic(listlist);

return listlist;

}

else

{

return null;

}

}

}

private void plastic(List<List<String>> lists)

{

if (lists.Count > 1)

{

int w = lists[0].Count;

for (int i = 1; i < lists.Count;i++ )

{

List<String> list = lists[i];

if (list.Count < w)

{

int addNum = w- list.Count;

for (int j = 0; j < addNum; j++)

{

list.Add("");

}

}

}

}

}

public List<List<String>> readfirststr(String filename, String sheetname)

{

using (FileStream fs = File.OpenRead(filename)) //打开myxls.xls文件

{

List<List<String>> listlist = new List<List<string>>(); //--------------------------------

NPOI.HSSF.UserModel.HSSFWorkbook wk = new HSSFWorkbook(fs); //把xls文件中的数据写入wk中

ISheet sheet = wk.GetSheet(sheetname); //根据 sheet 名

if (null != sheet)

{

for (int j = 0; j <= sheet.LastRowNum; j++) //LastRowNum 是当前表的总行数

{

List<String> list = new List<string>(); //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IRow row = sheet.GetRow(j); //读取当前行数据

if (row != null)

{

for (int k = 0; k < row.LastCellNum; k++) //LastCellNum 是当前行的总列数

{

ICell cell = row.GetCell(k); //当前表格

if (cell != null)

{

list.Add(cell.ToString());

}

else

{

list.Add("");

}

}

}

listlist.Add(list);

}

this.plastic(listlist);

return listlist;

}

else

{

return null;

}

}

}

public List<Action> readActionListFromExcelFile(String fileName)

{

//////Console.WriteLine("readActionListFromExcelFile方法的参数= " + fileName);

//得到第一个页签的名字

String sheetName = getsheets(fileName)[0];

//得到所有的文本

List<List<String>> strListList = readfirststr(fileName, sheetName);

//转换成 Action的列表

List<Action> actionList = strListList2ActionList(strListList);

return actionList;

}

//一个二维数组转换成一个字典 里面这一维 是长度为2的数组，转换出来，一个做键一个做值

public Dictionary<String, String> strList2Dic(List<List<String>> strListList)

{

Dictionary<String, String> dic = new Dictionary<string, string>();

foreach (List<String> list in strListList)

{

String k = list[0];

String v = list[1];

dic.Add(k, v);

}

return dic;

}

List<Action> strListList2ActionList(List<List<String>> strListList)

{

List<Action> actionList = new List<Action>();

foreach (List<String> strList in strListList)

{

Action action = strList2Action(strList);

if (null != action)

{

actionList.Add(strList2Action(strList));

}

}

return actionList;

}

Action strList2Action( List<String> strList)

{

if (strList.Count == 5)

{

int index = Int32.Parse(strList[0]);

bool isRun = (strList[4].Trim() == "是");

String actionName = strList[1];

String driverName = "default";

switch (actionName)

{

case "新建驱动":

String browserName = strList[2];

driverName = strList[3];

return new CreateDriverAction(index, browserName, driverName, isRun);

case "新建Appium驱动":

String deviceName = strList[2];

String appPackageAndAppActivity = strList[3];

return new CreateAppiumDriverAction(index, deviceName, appPackageAndAppActivity, isRun);

case "新建IOS驱动":

String macIP = strList[2];

String appPath = strList[3];

return new CreateIOSDriverAction(index, macIP, appPath, isRun);

case "打开网址":

driverName = strList[2];

String url = strList[3];

return new GoAction(index,driverName,url,isRun);

case "设置参数":

String parName = strList[2];

String parValue = strList[3];

return new SetParameterAction(index, parName, parValue, isRun);

case "执行方法":

driverName = strList[2];

String functionName = strList[3];

return new ExecuteFunctionAction(index, driverName, functionName, isRun);

case "单步方法":

String oneStepAction = strList[2];

String oneStepParse = strList[3];

return new OneStepAction(index, oneStepAction, oneStepParse, isRun);

case "暂停":

int timeLong = Int32.Parse(strList[2].Trim());

return new SleepAction(index, timeLong, isRun);

case "调用其他用例":

String casePath = strList[3];

return new ExeOtherCaseAction(index, casePath, isRun);

case "退出":

driverName = strList[2];

return new QuitAction(index, driverName, isRun);

case "注释":

String annotationText = strList[3];

return new AnnotationAction(index, annotationText, isRun);

case "Python代码":

String code = strList[3];

return new PythonCodeAction(index, code, isRun);

case "{":

return new AddIndentAction(index, isRun);

case "}":

return new RecoverIndentAction(index, isRun);

case "读取外部配置":

String excelPath = strList[3];

return new ReadExternalConfAction(index, excelPath, isRun);

case "":

String par1 = strList[2];

String par2 = strList[3];

return new NullAction(index, par1, par2, isRun);

default:

return null;

}

}

else

{

return null;

}

}

public Dictionary<String, String> getDicByEnvNameFromExcel(String excelFileName, String envName)

{

List<List<String>> lists = this.readfirststr(excelFileName, "Sheet1");

List<String> head = lists[0];

int l = lists.Count;

Dictionary<String, String> dic = new Dictionary<string, string>();

int x = head.IndexOf(envName);

if (x > 0)

{

for (int y = 1; y < l; y++)

{

List<String> list = lists[y];

String key = list[0];

String value = list[x];

dic.Add(key, value);

}

}

return dic;

}

}

}

文件 Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

using System.IO;

using System.Threading;

using System.Diagnostics;

using Microsoft.VisualBasic;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

using Microsoft.Win32;

namespace CaseManager

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.DoubleBuffered = true;

SetStyle(ControlStyles.ResizeRedraw, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

SetStyle(ControlStyles.AllPaintingInWmPaint, true);

Control.CheckForIllegalCrossThreadCalls = false;

nodeManager = new iTreeNodeManager(this);

actionLineListClipboard = new List<ActionLine>();

}

private int numOfLoadPanel = 0;

public List<ActionLine> actionLineListClipboard;

public TextBox getTextBox()

{

return this.textBox\_log;

}

public void showLoading()

{

this.panel\_loading.Left = (this.Width - this.panel\_loading.Width) / 2;

this.panel\_loading.Top = (this.Height - this.panel\_loading.Height) / 2-80;

this.panel\_loading.Visible = true;

}

public void hideLoading()

{

this.panel\_loading.Visible = false;

}

public iTreeNodeManager nodeManager;// = new iTreeNodeManager( );

public PanelAndTabControl panelAndTabControl;

public Form\_CopySteps form\_CopySteps;

public bool closeAfterRunEnd = false;

private void Form1\_Load(object sender, EventArgs e)

{

this.panel\_loading.Visible = false;

String prPath = new TextProcessor().getAbsPath();

this.Text += "-" + Directory.CreateDirectory(prPath).Name;

//自动运行

if (File.Exists(prPath + "\\AUTORUN.txt"))

{

this.WindowState = FormWindowState.Minimized;

this.closeAfterRunEnd = true;

List<String> list = new DirectoryManager().getAllFiles(prPath + "\\测试用例", "xls");

new Runner().xxxxxx(new FormAndList(this, list));

}

else

{

Form1\_ResizeEnd(sender, e);

iTreeNode itn = nodeManager.getTreefromPath(prPath + "\\测试用例");

treeView1.Nodes.Add(itn);

treeView1.Width = treeView1.Parent.Width;

treeView1.Height = treeView1.Parent.Height;

treeView1.NodeMouseDoubleClick += new TreeNodeMouseClickEventHandler(treeView1\_NodeMouseDoubleClick);

panelAndTabControl = new PanelAndTabControl();

// splitContainer\_main.SplitterDistance = 100;

this.splitContainer\_2.Panel2.Controls.Add(panelAndTabControl);

panelAndTabControl.Top = 0;

panelAndTabControl.Left = 0;

panelAndTabControl.Width = panelAndTabControl.Parent.Width;

panelAndTabControl.Height = panelAndTabControl.Parent.Height;

panelAndTabControl.tabControl.Width = panelAndTabControl.Width;

panelAndTabControl.tabControl.Height = panelAndTabControl.Height;

this.Form1\_SizeChanged(null, null);

this.timer\_showLogo.Enabled = true;

}

}

int caseHadOpened(TabControl tabControl, String caseFilePath)

{

int i = 0;

foreach (TabPage tabPage in tabControl.Controls)

{

if (tabPage.Name == caseFilePath)

{

return i;

}

i++;

}

return -1;

}

//树结点双击

void treeView1\_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e)

{

//////////Console.WriteLine("双击");

iTreeNode node = (iTreeNode)e.Node;

if (node.ntype == NodeType.FILE)

{

//////////Console.WriteLine(node.path);

// System.Diagnostics.Process.Start("excel.exe", node.path);

//////////Console.WriteLine("文件路径" + node.path);

String filePath = node.path;

int i = caseHadOpened(this.panelAndTabControl.tabControl, filePath);

if (i > -1) //已经打开

{

this.panelAndTabControl.tabControl.SelectedIndex = i;

}

else

{

try

{

this.showLoading();

List<Action> actionList = new ExcelProcessor().readActionListFromExcelFile(node.path);

ActionLines actionLines = new ActionLines(actionList, this.panelAndTabControl.Width - 33 ,this);

//Console.WriteLine("所有actionLines生成完成" + DateTime.Now);

ETabPage eTabPage = new ETabPage(this, actionLines, filePath);

//Console.WriteLine("eTabPage生成完成" + DateTime.Now);

this.panelAndTabControl.tabControl.Controls.Add(eTabPage);

int count = this.panelAndTabControl.tabControl.Controls.Count;

this.panelAndTabControl.tabControl.SelectedIndex = count - 1;

}

catch (Exception e9)

{

MessageBox.Show("请不要用其他软件同时打开用例文件！", "提示");

}

}

}

}

private void treeView1\_AfterCheck(object sender, TreeViewEventArgs e)

{

nodeManager.selectTree((iTreeNode)(e.Node), e.Node.Checked);

}

private void button1\_Click(object sender, EventArgs e)

{

//Dictionary<String, String> dic = new Dictionary<string, string>();

//dic.Add("a", "A");

//dic.Add("b", "B");

new CaseCreater().createpy("case.py", "template.txt");

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void button1\_Click\_1(object sender, EventArgs e)

{

List<List<String>> list = new ExcelProcessor().readfirststr("a.xls", "Sheet1");

////////Console.WriteLine(list);

}

private void button\_save\_Click(object sender, EventArgs e)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

eTabPage.save();

}

private void splitContainer\_2\_SplitterMoved(object sender, SplitterEventArgs e)

{

this.treeView1.Width = this.treeView1.Parent.Width;

this.treeView1.Height = this.treeView1.Parent.Height;

if (this.panelAndTabControl != null)

{

this.panelAndTabControl.Width = this.panelAndTabControl.Parent.Width;

this.panelAndTabControl.Height = this.panelAndTabControl.Parent.Height;

this.panelAndTabControl.tabControl.Width = this.panelAndTabControl.Width;

this.panelAndTabControl.tabControl.Height = this.panelAndTabControl.Height;

this.panelAndTabControl.setETabPageSizes(this.panelAndTabControl.tabControl.Width, this.panelAndTabControl.tabControl.Height);

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

actionLines.setIndent();

}

}

}

private void button1\_Click\_2(object sender, EventArgs e)

{

List<String> list = new List<String>();

foreach (ETabPage eTabPage in this.panelAndTabControl.tabControl.Controls)

{

list.Add(eTabPage.Name);

}

form\_CopySteps = new Form\_CopySteps(this, list);

form\_CopySteps.Show();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

this.panelAndTabControl.copyOrCutSteps(0, 1, 5, 1, 1, "cut");

}

private void splitContainer\_main\_Resize(object sender, EventArgs e)

{

////////Console.WriteLine("ssssssssssss");

}

private void button3\_Click(object sender, EventArgs e)

{

ETabPage etabPage = (ETabPage)this.panelAndTabControl.tabControl.Controls[0];

etabPage.BackColor = Color.Blue;

etabPage.Width -= 20;

}

private void Form1\_SizeChanged(object sender, EventArgs e)

{

if (this.WindowState == FormWindowState.Maximized || this.WindowState == FormWindowState.Normal)

{

this.Form1\_ResizeEnd(sender, e);

}

}

private void 复制或剪切步骤ToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<String> targetCasePath = new List<string>();

foreach (ETabPage eTabPage in this.panelAndTabControl.tabControl.TabPages)

{

////////Console.WriteLine(eTabPage.Name);

targetCasePath.Add(eTabPage.Name);

}

new Form\_CopySteps(this, targetCasePath).ShowDialog();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void 保存ToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.panelAndTabControl.saveSelectedTab();

}

private void 保存全部ToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.panelAndTabControl.saveAllTab();

}

private void comboBox1\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

}

private void splitContainer\_main\_SplitterMoved(object sender, SplitterEventArgs e)

{

this.Form1\_ResizeEnd(sender, e);// .Form1\_SizeChanged(sender, e);

}

private void button1\_Click\_3(object sender, EventArgs e)

{

Dictionary<String, String> dic = new ExcelProcessor().getDicByEnvNameFromExcel("d:\\e.xls", "e3");

////////Console.WriteLine(dic);

}

private void button\_runAll\_Click(object sender, EventArgs e)

{

this.closeAfterRunEnd = true;

String prPath = new TextProcessor().getAbsPath();

List<String> list = new DirectoryManager().getAllFiles(prPath + "\\测试用例", "xls");

new Runner().xxxxxx(new FormAndList(this, list));

}

private void button1\_Click\_4(object sender, EventArgs e)

{

new CommandExecutor().execute2(@"E:\myCsharp\CaseManagerPy\CaseManager\bin\Debug\PythonSelenium2\src\testCase\aaa.py");

}

private void timerExecute\_Tick(object sender, EventArgs e)

{

}

private String path2ImportCode(String path)

{

String[] tmps = path.Split(new String[] { "src\\" }, StringSplitOptions.None);

String tmp = tmps[1].Replace(".py", "").Replace("\\", ".");

String[] tmps2 = tmp.Split(new String[] { "." }, StringSplitOptions.None);

String pythonClassName = tmps2[tmps2.Length - 1];

String code = "from " + tmp + " import " + pythonClassName;

return code;

}

private void button1\_Click\_5(object sender, EventArgs e)

{

List<String> list = new DirectoryManager().getAllFiles(@"E:\myCsharp\CaseManagerPy\CaseManager\bin\Debug\PythonSelenium2\src\testPage", "py");

List<String> list2 = new List<string>();

foreach (String str in list)

{

if (!str.Contains("\_\_init\_\_"))

{

String codeLine = path2ImportCode(str);

list2.Add(codeLine);

}

}

foreach (String str in list2)

{

//////Console.WriteLine(str);

}

}

private void button1\_Click\_6(object sender, EventArgs e)

{

String tt = new TextProcessor().getRandom(6);

//////Console.WriteLine(tt);

}

private void button1\_Click\_7(object sender, EventArgs e)

{

this.Height -= 1;

}

private void Form1\_ResizeEnd(object sender, EventArgs e)

{

this.splitContainer\_main.Width = this.Width - splitContainer\_main.Left - 18;

this.splitContainer\_main.Top = 30;

this.splitContainer\_main.Height = this.Height - splitContainer\_main.Top - 40;

this.splitContainer\_2.Width = this.splitContainer\_2.Parent.Width;

this.splitContainer\_2.Height = this.splitContainer\_2.Parent.Height - 4;

treeView1.Width = treeView1.Parent.Width;

treeView1.Height = treeView1.Parent.Height;

if (null != panelAndTabControl)

{

panelAndTabControl.Width = panelAndTabControl.Parent.Width;

panelAndTabControl.Height = panelAndTabControl.Parent.Height;

panelAndTabControl.tabControl.Width = panelAndTabControl.Width;

panelAndTabControl.tabControl.Height = panelAndTabControl.Height;

panelAndTabControl.setETabPageSizes(panelAndTabControl.tabControl.Width, panelAndTabControl.tabControl.Height);

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

actionLines.setIndent();

}

}

this.textBox\_log.Size = this.textBox\_log.Parent.Size;

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

//Console.WriteLine(e.KeyCode);

//Console.WriteLine("shift=" + e.Shift);

//Console.WriteLine("control=" + e.Control);

//Console.WriteLine("shift + control=" + (e.Shift && e.Control));

//Console.WriteLine("按下c=" + ( e.KeyCode== Keys.C));

//Console.WriteLine(e.KeyCode == Keys.C && e.Shift && e.Control);

\* \*/

//快捷键

//插入一行

if (e.KeyCode == Keys.I && e.Control || e.KeyCode == Keys.Q && e.Control)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

int currentIndex = actionLines.currentIndex;

actionLines.insertActionLine(currentIndex, 1);

}

}

//删除一行

else if (e.KeyCode == Keys.D && e.Control)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

int currentIndex = actionLines.currentIndex;

actionLines.deleteActionLine(currentIndex);

}

}

//设置为单步

else if (e.KeyCode == Keys.D1 && e.Control)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

int currentIndex = actionLines.currentIndex;

((ActionLine)actionLines.Controls[currentIndex]).comboBox\_key.Text = "单步方法";

}

}

else if (e.KeyCode == Keys.Up && e.Control)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

int currentIndex = actionLines.currentIndex;

actionLines.up(currentIndex);

}

}

else if (e.KeyCode == Keys.Down && e.Control)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

ActionLines actionLines = eTabPage.actionLines;

int currentIndex = actionLines.currentIndex;

actionLines.down(currentIndex);

}

}

}

private void 当前用例ToolStripMenuItem\_Click(object sender, EventArgs e)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

String filePath = eTabPage.Name;

List<String> list = new List<String>();

list.Add(filePath);

new Runner().xxxxxx(new FormAndList(this, list));

}

}

private void 打开的用例ToolStripMenuItem\_Click(object sender, EventArgs e)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

if (null != eTabPage)

{

List<String> list = new List<String>();

System.Windows.Forms.Control.ControlCollection eTabPages = eTabPage.Parent.Controls;

foreach (ETabPage tmp in eTabPages)

{

list.Add(tmp.Name);

}

new Runner().xxxxxx(new FormAndList(this, list));

}

}

private void 测试用例ToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("explorer.exe", new TextProcessor().getAbsPath() + "\\测试用例");

}

private void 配置ToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("explorer.exe", new TextProcessor().getAbsPath() + "\\conf");

}

private void python工程ToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("explorer.exe", new TextProcessor().getAbsPath() + "\\PythonSelenium2");

}

private void seleniumToolStripMenuItem\_Click(object sender, EventArgs e)

{

new CommandExecutor().execute("pip install selenium==2.53.0");

}

private void pillowToolStripMenuItem\_Click(object sender, EventArgs e)

{

new CommandExecutor().execute("pip install pillow");

}

private void pymysqlToolStripMenuItem\_Click(object sender, EventArgs e)

{

new CommandExecutor().execute("pip install pymysql");

}

private void requestsToolStripMenuItem\_Click(object sender, EventArgs e)

{

new CommandExecutor().execute("pip install requests");

}

private void pynputToolStripMenuItem\_Click(object sender, EventArgs e)

{

new CommandExecutor().execute("pip install pynput");

}

private void 需安装的包ToolStripMenuItem\_Click(object sender, EventArgs e)

{

String command = new TextProcessor().getAbsPath() + "\\pip\\pip.bat";

new CommandExecutor().execute(command);

}

private void button1\_Click\_8(object sender, EventArgs e)

{

List<List<String>> lists = new ExcelProcessor().readfirststr("d:\\Book1.xls");

// //Console.WriteLine(lists);

}

private void timer\_hideLogo\_Tick(object sender, EventArgs e)

{

this.panel1.Visible = false;

this.timer\_hideLogo.Enabled = false;

}

private Bitmap KiCut(Bitmap b, int StartX, int StartY, int iWidth, int iHeight)

{

if (b == null)

{

return null;

}

int w = b.Width;

int h = b.Height;

if (StartX >= w || StartY >= h)

{

return null;

}

if (StartX + iWidth > w)

{

iWidth = w - StartX;

}

if (StartY + iHeight > h)

{

iHeight = h - StartY;

}

try

{

Bitmap bmpOut = new Bitmap(iWidth, iHeight, PixelFormat.Format24bppRgb);

Graphics g = Graphics.FromImage(bmpOut);

g.DrawImage(b, new Rectangle(0, 0, iWidth, iHeight), new Rectangle(StartX, StartY, iWidth, iHeight), GraphicsUnit.Pixel);

g.Dispose();

return bmpOut;

}

catch

{

return null;

}

}

private void showLogo()

{

panel1.Left = (this.Width - panel1.Width) / 2;

panel1.Top = (this.Height - panel1.Height) / 2;

panel1.Visible = false;

pictureBox\_logo.Visible = false;

Bitmap b = new Bitmap(this.Width, this.Height);

//得到窗口的图片

this.DrawToBitmap(b, new Rectangle(0, 0, this.Width, this.Height));

Bitmap b2 = this.KiCut(b, panel1.Left + 8, panel1.Top + 31, panel1.Width, panel1.Height);

panel1.BackgroundImage = b2;

pictureBox\_logo.BackColor = Color.Transparent;

panel1.Visible = true;

pictureBox\_logo.Visible = true;

}

private void button1\_Click\_9(object sender, EventArgs e)

{

}

private void timer\_showLogo\_Tick(object sender, EventArgs e)

{

this.showLogo();

this.timer\_hideLogo.Enabled = true;

}

private void timer\_showLogo\_Tick\_1(object sender, EventArgs e)

{

this.timer\_showLogo.Enabled = false;

this.showLogo();

this.timer\_hideLogo.Enabled = true;

}

private void button1\_Click\_10(object sender, EventArgs e)

{

this.showLogo();

}

private void 下载各版本chromedriverToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.openUrl("http://chromedriver.storage.googleapis.com/index.html");

}

private void openUrl(String url)

{

RegistryKey key = Registry.ClassesRoot.OpenSubKey(@"http\shell\open\command\");

string s = key.GetValue("").ToString();

String[] tmps = s.Split(new String[] { "\"" }, StringSplitOptions.RemoveEmptyEntries);

String tmp = tmps[0];

System.Diagnostics.Process.Start(tmp,url);

}

private void 关于ToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void 查找ToolStripMenuItem\_Click(object sender, EventArgs e)

{

ETabPage eTabPage = (ETabPage)this.panelAndTabControl.tabControl.SelectedTab;

new Form\_Search(eTabPage).ShowDialog();

}

private void timer\_loading\_Tick(object sender, EventArgs e)

{

this.numOfLoadPanel = (this.numOfLoadPanel + 1) % 7;

Console.WriteLine(numOfLoadPanel);

if (0 == this.numOfLoadPanel)

{

this.panel\_0.Visible = false;

this.panel\_1.Visible = false;

this.panel\_2.Visible = false;

this.panel\_3.Visible = false;

this.panel\_4.Visible = false;

this.panel\_5.Visible = false;

}

else if (1 == this.numOfLoadPanel)

{

this.panel\_0.Visible = true;

}

else if (2 == this.numOfLoadPanel)

{

this.panel\_1.Visible = true;

}

else if (3 == this.numOfLoadPanel)

{

this.panel\_2.Visible = true;

}

else if (4 == this.numOfLoadPanel)

{

this.panel\_3.Visible = true;

}

else if (5 == this.numOfLoadPanel)

{

this.panel\_4.Visible = true;

}

else

{

this.panel\_5.Visible = true;

}

}

}

public class AnnotationTextBox : TextBox

{

public AnnotationTextBox()

: base()

{

this.ForeColor = Color.Green;

}

}

public class PythonTextBox : TextBox

{

public PythonTextBox()

: base()

{

this.ForeColor = Color.Blue;

}

}

//带有一个TabControl的Panel

public class PanelAndTabControl : Panel

{

public TabControl tabControl;

//构造方法

public PanelAndTabControl()

: base()

{

tabControl = new TabControl();

tabControl.Left = 0;

tabControl.Top = 0;

////////Console.WriteLine("tabControl宽度=" + tabControl.Width);

this.Controls.Add(tabControl);

}

//增加一个 tabPage

public void addTabPage(ETabPage tabPage)

{

this.tabControl.Controls.Add(tabPage);

////////Console.WriteLine(tabPage.Name);

}

//设置所有 eTabPage的宽度和高度

public void setETabPageSizes(int width, int height)

{

this.Width = width;

this.Height = height;

foreach (ETabPage tabPage in this.tabControl.Controls)

{

// tabPage.BackColor = Color.FromArgb(196, 196, 196);

tabPage.Width = width;

tabPage.Height = height;

tabPage.setActionLinesSize(width - 30, height);

}

}

//list1为插入的一个list

//list2为被插入的

public List<Action> insertListToList(List<Action> list1, List<Action> list2, int i)

{

List<Action> list3 = new List<Action>();

if (i >= 0 && i < list2.Count)

{

List<Action> tmp1 = list2.GetRange(0, i);

List<Action> tmp2 = list2.GetRange(i, list2.Count - i);

list3.AddRange(tmp1);

list3.AddRange(list1);

list3.AddRange(tmp2);

}

else if (i < 0)

{

list3.AddRange(list1);

list3.AddRange(list2);

}

else if (i >= list2.Count)

{

list3.AddRange(list2);

list3.AddRange(list1);

}

return list3;

}

public void saveSelectedTab()

{

ETabPage eTabPage = (ETabPage)this.tabControl.SelectedTab;

if (null != eTabPage)

{

eTabPage.save();

}

}

public void saveAllTab()

{

foreach (ETabPage eTabPage in this.tabControl.TabPages)

{

eTabPage.save();

}

}

//复制或剪切一些步骤

public void copyOrCutSteps(int sourceTabIndex, int sourceStepStart, int sourceStepEnd, int targetTabIndex, int targetStep, String copyOrCut)

{

ETabPage eTabPage\_source = (ETabPage)this.tabControl.Controls[sourceTabIndex];

ETabPage eTabPage\_target = (ETabPage)this.tabControl.Controls[targetTabIndex];

eTabPage\_source.actionLines.refreshActionList();

eTabPage\_target.actionLines.refreshActionList();

List<Action> actionList\_source = eTabPage\_source.actionLines.actionList;

List<Action> actionList\_target = eTabPage\_target.actionLines.actionList;

if (actionList\_source.Count > 0)

{

//sourceStepStart

if (sourceStepStart < 0)

{

sourceStepStart = 0;

}

if (sourceStepStart > actionList\_source.Count - 1)

{

sourceStepStart = actionList\_source.Count - 1;

}

//sourceStepEnd

if (sourceStepEnd < 0)

{

sourceStepEnd = 0;

}

if (sourceStepEnd > actionList\_source.Count - 1)

{

sourceStepEnd = actionList\_source.Count - 1;

}

if (targetStep < 0)

{

targetStep = 0;

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if (targetStep > actionList\_target.Count)

{

targetStep = actionList\_target.Count;

}

List<Action> listTmp = actionList\_source.GetRange(sourceStepStart, sourceStepEnd - sourceStepStart + 1);

List<Action> actionList\_targetNew = new List<Action>();

//复制或剪切给自己

if (sourceTabIndex == targetTabIndex)

{

//List<Action> actionList\_targetNew = new List<Action>();

//复制

if (copyOrCut.ToUpper() == "COPY")

{

//复制到自己中 不允许

if (targetStep > sourceStepStart && targetStep <= sourceStepEnd)

{

MessageBox.Show("不允许往源步骤中粘贴");

}

//复制中间其他地方复制

else

{

actionList\_targetNew = insertListToList(listTmp, actionList\_source, targetStep);

eTabPage\_source.actionLines.actionList = actionList\_targetNew;

eTabPage\_source.actionLines.refreshGui();

}

}

//剪切

else

{

if (targetStep > sourceStepStart && targetStep <= sourceStepEnd)

{

MessageBox.Show("不允许往源步骤中粘贴");

}

//后面剪切到前面 //先删除 后插入

else if (targetStep <= sourceStepStart)

{

int l = listTmp.Count;

for (int i = 0; i < l; i++)

{

actionList\_source.RemoveAt(sourceStepStart);

}

actionList\_targetNew = insertListToList(listTmp, actionList\_source, targetStep);

eTabPage\_source.actionLines.actionList = actionList\_targetNew;

eTabPage\_source.actionLines.refreshGui();

}

//前面剪切到后面 //先插入 后删除

else if (targetStep > sourceStepEnd)

{

actionList\_targetNew = insertListToList(listTmp, actionList\_source, targetStep);

int l = listTmp.Count;

for (int i = 0; i < l; i++)

{

actionList\_targetNew.RemoveAt(sourceStepStart);

}

eTabPage\_source.actionLines.actionList = actionList\_targetNew;

eTabPage\_source.actionLines.refreshGui();

}

else

{

MessageBox.Show("不允许");

}

}

}

else //复制或剪切到其他tab

{

//List<Action> actionList\_targetNew = new List<Action>();

//复制

if (copyOrCut.ToUpper() == "COPY")

{

actionList\_targetNew = insertListToList(listTmp, actionList\_target, targetStep);

eTabPage\_target.actionLines.actionList = actionList\_targetNew;

eTabPage\_target.actionLines.refreshGui();

this.tabControl.SelectedIndex = targetTabIndex;

}

else

{

actionList\_targetNew = insertListToList(listTmp, actionList\_target, targetStep);

eTabPage\_target.actionLines.actionList = actionList\_targetNew;

eTabPage\_target.actionLines.currentIndex = targetStep;

eTabPage\_target.actionLines.refreshGui();

this.tabControl.SelectedIndex = targetTabIndex;

int l = listTmp.Count;

for (int i = 0; i < l; i++)

{

actionList\_source.RemoveAt(sourceStepStart);

eTabPage\_source.actionLines.actionList = actionList\_source;

}

eTabPage\_source.actionLines.currentIndex = sourceStepStart - 1;

eTabPage\_source.actionLines.refreshGui();

}

}

}

else

{

MessageBox.Show("源长度为0");

}

}

}

public class ETabPage : TabPage

{

public Form1 form;

public ActionLines actionLines;

public ETabPage(Form1 form, ActionLines actionLines, String filePath)

: base()

{

this.DoubleBuffered = true;

SetStyle(ControlStyles.ResizeRedraw, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

SetStyle(ControlStyles.AllPaintingInWmPaint, true);

this.form = form;

this.Name = filePath;

String[] tmps = filePath.Split(new String[] { "\\" }, StringSplitOptions.RemoveEmptyEntries);

String fileName = tmps[tmps.Length - 1];

this.Text = fileName;

this.actionLines = actionLines;

this.Controls.Add(this.actionLines);

this.AutoScroll = true;

this.BackColor = Color.FromArgb(196, 196, 196);

this.ContextMenuStrip = new EtagContextMenuStrip(this.form, this);

}

public void save()

{

List<List<String>> listList = this.actionLines.toListList();

String filePath = this.Name;

new ExcelProcessor().writeToExcelFile(listList, filePath);

}

public void setActionLinesSize(int width, int height)

{

this.Width = width;

this.Height = height;

this.actionLines.setActionLineWidth(width);

}

}

//iTreeNode管理器

public class iTreeNodeManager

{

Form1 form;

public iTreeNodeManager(Form1 form)

{

this.form = form;

}

public iTreeNode getTreefromPath(String path)

{

iTreeNode itn = new iTreeNode(this.form, path, NodeType.DIR);

itn.ChildNodesAddChildNodes();

return itn;

}

//得到一个文件夹节点下的所有选中的文件节点

public ArrayList getSelectedFileNodes(iTreeNode itn)

{

selectedFileNodes.Clear();

addChildNodestoList(itn);

return selectedFileNodes;

}

public ArrayList selectedFileNodes = new ArrayList();

//得到这个节点下的所有的选中的文件

public void addChildNodestoList(iTreeNode itn)

{

foreach (iTreeNode n in itn.Nodes)

{

//是目录的

if (n.ntype == NodeType.DIR)

{

//递归调用自己

addChildNodestoList(n);

}

//是文件

else

{

//是选中的

if (n.Checked == true)

{

selectedFileNodes.Add(n);

}

}

}

}

public void selectTree(iTreeNode itn, bool isSelect)

{

TreeNodeCollection list = itn.Nodes;

foreach (iTreeNode n in list)

{

n.Checked = isSelect;

if (n.ntype == NodeType.DIR)

{

selectTree(n, isSelect);

}

}

}

}

class FormAndList

{

public Form1 form;

public List<String> list;

public FormAndList(Form1 form, List<String> list)

{

this.form = form;

this.list = list;

}

}

class FormAndListAndIndex

{

public Form1 form;

public String filePath;

public int index;

public FormAndListAndIndex(Form1 form, String filePath, int index)

{

this.form = form;

this.filePath = filePath;

this.index = index;

}

}

class Runner

{

String reportFile = new TextProcessor().getAbsPath() + "\\report\\report.html";

String pyFile = new TextProcessor().getAbsPath() + "\\PythonSelenium2\\src\\testCase\\TmpCase.py";

String logFile = new TextProcessor().getAbsPath() + "\\PythonSelenium2\\log\\testCase\\TmpCase\\log.txt";

String picFile = new TextProcessor().getAbsPath() + "\\PythonSelenium2\\log\\testCase\\TmpCase\\screenCapture\\img.jpg";

public void xxxxxx(Object o)

{

ParameterizedThreadStart Pts = new ParameterizedThreadStart(a);

Thread thread = new Thread(Pts);

thread.Start(o);

}

public void a(Object o)

{

FormAndList formAndList = (FormAndList)o;

Form1 form = formAndList.form;

List<String> filePaths = formAndList.list;

runFiles(form, filePaths);//

}

public void runFiles(Form1 form, List<String> filePaths)

{

TextProcessor tp = new TextProcessor();

int beginIndex = tp.getBeginIndex();

String prPath = new TextProcessor().getAbsPath();

//得到报告目录下的日志的目录

String reportLogFile = prPath + "\\report\\log";

//得到报告目录下日志目录下的图片目录

String reportPicFile = prPath + "\\report\\pic";

//清空这两个目录

new DirectoryManager().clearDir(reportLogFile);

new DirectoryManager().clearDir(reportPicFile);

tp.writeContent(reportFile, "<html>", false);

tp.writeContent(reportFile, "<head>", true);

tp.writeContent(reportFile, "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\" />", true);

tp.writeContent(reportFile, "<title>自动化测试报告</title>", true);

tp.writeContent(reportFile, "<style type=\"text/css\">", true);

tp.writeContent(reportFile, ".success {background-color: #66FF33}", true);

tp.writeContent(reportFile, ".failed {background-color: red}", true);

tp.writeContent(reportFile, "table {border-spacing:0}", true);

tp.writeContent(reportFile, "td {padding: 4px;border:1px solid #7F7F7F}", true);

tp.writeContent(reportFile, "</style>", true);

tp.writeContent(reportFile, "</head>", true);

tp.writeContent(reportFile, "<body>", true);

tp.writeContent(reportFile, "<table>", true);

tp.writeContent(reportFile, "<tr><td>序号</td><td>用例文件</td><td>结果</td><td>日志</td><td>截图</td></tr>", true);

int l = filePaths.Count;

String exeStatuFile = prPath + "\\executestatu.txt";

for (int i = 0; i < l; i++)

{

//tp.writeContent(exeStatuFile, "RUNNING\_" + (i + beginIndex), false);

String trStr = runFile(form, filePaths[i], i + beginIndex);

tp.writeContent(reportFile, trStr, true);

}

tp.writeContent(reportFile, "</table>", true);

tp.writeContent(reportFile, "</body>", true);

tp.writeContent(reportFile, "</html>", true);

//如果是使用 执行全部用例 这个按钮调用的 ，跑完了就会关闭窗口

if (form.closeAfterRunEnd)

{

form.Close();

}

}

String runFile(Form1 form, String filePath, int num)

{

String prPath = new TextProcessor().getAbsPath();

String script = new TextProcessor().getWholePyCodeFromScriptFile(prPath + "\\conf\\pythoncode.txt", prPath + "\\PythonSelenium2\\src\\testPage", filePath, prPath + "\\conf\\env.txt", prPath + "\\conf\\data.xls",prPath + "\\conf\\import.txt");

File.WriteAllText(pyFile, script, new UTF8Encoding(false));

FormAndListAndIndex formAndListAndIndex = new FormAndListAndIndex(form, filePath, num);

ParameterizedThreadStart Pts = new ParameterizedThreadStart(ant);

Thread thread = new Thread(Pts);

this.tr = null;

thread.Start(formAndListAndIndex);

//每一秒检查 tr 的值，如果是 null ，说明这个用例还没跑，如果达到次数还没跑完，说明就是超时了

int t = 0;

while (this.tr == null && t < 300)

{

Thread.Sleep(1000);

t++;

}

if (tr == null)

{

form.getTextBox().AppendText("用例超时，被强行停止。");

thread.Abort();

new CommandExecutor().execute("taskkill /im chromedriver.exe /F");

new CommandExecutor().execute("taskkill /im chrome.exe /F");

String tdNum = "<td>" + num + "</td>";

String tdFilePath = "<td>" + filePath + "</td>";

String tdResult = "<td class ='timeout'>超时</td>";

String tdLog = "<td>无日志</td>";

String tdPic = "<td>无截屏</td>";

String prPah = new TextProcessor().getAbsPath();

//得到日志的文本

String reportLogFile = prPah + "\\report\\log\\" + num + "log.txt";

String reportLogFileInHtml = "log/" + num + "log.txt";

String reportPicFile = prPah + "\\report\\pic\\" + num + "pic.jpg";

String reportPicFileInHtml = "pic/" + num + "pic.jpg";

if (File.Exists(logFile))

{

if (File.Exists(reportLogFile))

{

File.Delete(reportLogFile);

}

File.Copy(logFile, reportLogFile, true);

tdLog = "<td><a href='" + reportLogFileInHtml + "' target='\_black'>日志下载</a></td>";

}

if (File.Exists(picFile))

{

if (File.Exists(reportPicFile))

{

File.Delete(reportPicFile);

}

File.Copy(picFile, reportPicFile, true);

tdPic = "<td><a href='" + reportPicFileInHtml + "' target='\_black' class='failed'>错误截屏</a></td>";

}

this.tr = "<tr>" + tdNum + tdFilePath + tdResult + tdLog + tdPic + "</tr>";

return this.tr;

}

else

{

return this.tr;

}

}

public String tr = null;

public void ant(Object o)

{

this.tr = null;

FormAndListAndIndex t = (FormAndListAndIndex)o;

this.tr = ant(t.form, t.filePath, t.index);

}

public String ant(Form1 form, String filePath, int num)

{

String tdNum = "<td>" + num + "</td>";

String tdFilePath = "<td>" + filePath + "</td>";

String tdResult = "<td>未执行</td>";

String tdLog = "<td>无日志</td>";

String tdPic = "<td>无截屏</td>";

String prPath = new TextProcessor().getAbsPath();

System.IO.Directory.SetCurrentDirectory(prPath + "\\PythonSelenium2\\src\\testCase");

form.getTextBox().AppendText("当前目录=" + prPath + "\\PythonSelenium2\\src\\testCase");

String srcPath = prPath + "\\PythonSelenium2\\src";

String setpythonpathCommand = "set pythonpath=" + srcPath;

ProcessStartInfo info = new ProcessStartInfo();

info.FileName = "cmd.exe";

info.RedirectStandardInput = true;

info.RedirectStandardOutput = true;

info.RedirectStandardError = true;

info.CreateNoWindow = true;

info.UseShellExecute = false;

Process p = Process.Start(info);

p.StandardInput.WriteLine(setpythonpathCommand);

p.StandardInput.WriteLine("echo %pythonpath%");

p.StandardInput.WriteLine("Runner.py&exit");

p.StandardInput.AutoFlush = true;

String resultLine = "";

// DateTime dateTime = DateTime.Now;

//dateTime = dateTime.AddMinutes(1);

// dateTime = dateTime.AddSeconds(10);

while (!p.StandardOutput.EndOfStream)

{

String tmp = p.StandardOutput.ReadLine();

////////Console.WriteLine("tmp=" + tmp);

form.getTextBox().AppendText(tmp + "\r\n");

//如果到了这一句 就可以跳出了

if (tmp.StartsWith("测试结果") && tmp.EndsWith("通过"))

{

resultLine = tmp;

break;

}

}

////////Console.WriteLine("读取到最末尾");

p.StandardOutput.Close();

p.StandardInput.Close();

p.Close();

System.IO.Directory.SetCurrentDirectory(prPath);

//if (DateTime.Now >= dateTime)

if (false)

{

//////Console.WriteLine("超时");

tdResult = "<td class ='failed'>超时</td>";

new CommandExecutor().execute("taskkill /im chromedriver.exe /F");

new CommandExecutor().execute("taskkill /im chrome.exe /F");

}

else

{

bool success = false;

success = (resultLine == "测试结果通过");

if (success)

{

tdResult = "<td class='success'>成功</td>";

}

else

{

tdResult = "<td class ='failed'>失败</td>";

}

}

String prPah = new TextProcessor().getAbsPath();

//得到日志的文本

String reportLogFile = prPah + "\\report\\log\\" + num + "log.txt";

String reportLogFileInHtml = "log/" + num + "log.txt";

String reportPicFile = prPah + "\\report\\pic\\" + num + "pic.jpg";

String reportPicFileInHtml = "pic/" + num + "pic.jpg";

if (File.Exists(logFile))

{

if (File.Exists(reportLogFile))

{

File.Delete(reportLogFile);

}

File.Copy(logFile, reportLogFile, true);

tdLog = "<td><a href='" + reportLogFileInHtml + "' target='\_black'>日志下载</a></td>";

}

if (File.Exists(picFile))

{

if (File.Exists(reportPicFile))

{

File.Delete(reportPicFile);

}

File.Copy(picFile, reportPicFile, true);

tdPic = "<td><a href='" + reportPicFileInHtml + "' target='\_black' class='failed'>错误截屏</a></td>";

}

String tr = "<tr>" + tdNum + tdFilePath + tdResult + tdLog + tdPic + "</tr>";

return tr;

}

}

//右键菜单的类

class iContextMenuStrip : ContextMenuStrip

{

Form1 form;

//构造函数

public iContextMenuStrip(Form1 form, iTreeNode node)

: base()

{

ToolStripMenuItem item\_run = new ToolStripMenuItem("运行");

ToolStripMenuItem item\_clone = new ToolStripMenuItem("克隆");

ToolStripMenuItem item\_rename = new ToolStripMenuItem("改名");

// item\_rename.ShortcutKeys = Keys.F2 ;

// item\_rename.ShowShortcutKeys = true;

ToolStripMenuItem item\_delete = new ToolStripMenuItem("删除");

ToolStripMenuItem item\_createDir = new ToolStripMenuItem("新建目录");

ToolStripMenuItem item\_createCase = new ToolStripMenuItem("新建用例");

ToolStripMenuItem item\_copyPath = new ToolStripMenuItem("复制路径到剪贴板");

item\_run.Click += new EventHandler(this.runNode);

item\_clone.Click += new EventHandler(this.cloneNode);

item\_rename.Click += new EventHandler(this.renameNode);

item\_delete.Click += new EventHandler(this.deleteNode);

item\_createDir.Click += new EventHandler(this.createDirNode);

item\_createCase.Click += new EventHandler(this.createCaseNode);

item\_copyPath.Click += new EventHandler(this.copyPath);

this.sendernode = node;

this.Items.Add(item\_run);

if (node.ntype == NodeType.DIR)

{

this.Items.Add(item\_createDir);

this.Items.Add(item\_createCase);

}

else

{

this.Items.Add(item\_clone);

this.Items.Add(item\_rename);

}

this.Items.Add(item\_delete);

this.Items.Add(item\_copyPath);

this.form = form;

}

//成员变量

iTreeNode sendernode = null;

//增加目录节点

void createDirNode(object sender, EventArgs e)

{

if (this.sendernode.ntype == NodeType.DIR)

{

String dirName = Interaction.InputBox("请输入要新建的目录名", "提示", "", 100, 100).Trim();

if (dirName.Length > 0 && !dirName.Contains("\\") && !dirName.Contains("/") && !dirName.Contains(":") && !dirName.Contains("\*") && !dirName.Contains("?") && !dirName.Contains("\"") && !dirName.Contains("<") && !dirName.Contains(">") && !dirName.Contains("|"))

{

String path = this.sendernode.path;

String dirPath = path + "\\" + dirName;

if (new DirectoryInfo(dirPath).Exists)

{

MessageBox.Show("已经存在这个目录！", "新建目录失败");

}

else

{

Directory.CreateDirectory(dirPath);

iTreeNode node = new iTreeNode(form, dirPath, NodeType.DIR);

this.sendernode.Nodes.Add(node);

}

}

else

{

MessageBox.Show("输入值长度为0或包含非法字符，请重新输入！");

}

}

else

{

MessageBox.Show("文件下怎么新建目录？", "呵呵");

}

}

void copyPath(object sender, EventArgs e)

{

String path = this.sendernode.path;

int i = path.IndexOf("测试用例");

path = path.Substring(i);

//////Console.WriteLine(path);

Clipboard.SetDataObject(path, true);

}

//增加文件节点

void createCaseNode(object sender, EventArgs e)

{

String caseName = Interaction.InputBox("请输入要新建的用例名", "提示", "", 100, 100).Trim();

if (caseName.Length > 0 && !caseName.Contains("\\") && !caseName.Contains("/") && !caseName.Contains(":") && !caseName.Contains("\*") && !caseName.Contains("?") && !caseName.Contains("\"") && !caseName.Contains("<") && !caseName.Contains(">") && !caseName.Contains("|"))

{

String path = this.sendernode.path;

String casePath = path + "\\" + caseName + ".xls";

if (File.Exists(casePath))

{

MessageBox.Show("已经存在这个用例！", "新建用例失败");

}

else

{

try

{

String prPath = new TextProcessor().getAbsPath();

File.Copy(prPath + "\\conf\\初始用例.xls", casePath);

iTreeNode node = new iTreeNode(form, casePath, NodeType.FILE);

this.sendernode.Nodes.Add(node);

}

catch (Exception e16)

{

//////////Console.WriteLine(e16.Message);

MessageBox.Show("新建用例失败，是否使用了特殊字符！", "提示");

}

}

}

else

{

MessageBox.Show("输入值长度为0或包含非法字符，请重新输入！");

}

}

//重命名节点

void renameNode(object sender, EventArgs e)

{

if (this.sendernode.ntype == NodeType.FILE)

{

String oldName = this.sendernode.Text.Split(new String[] { "." }, StringSplitOptions.None)[0];

//弹出一个对话框，得到对话框的输入值

String newName = Interaction.InputBox("请输入新的用例名", "提示", oldName, 100, 100).Trim();

//////Console.WriteLine("新名=" + newName);

if (newName.Length > 0 && !newName.Contains("\\") && !newName.Contains("/") && !newName.Contains(":") && !newName.Contains("\*") && !newName.Contains("?") && !newName.Contains("\"") && !newName.Contains("<") && !newName.Contains(">") && !newName.Contains("|"))

{

String path = this.sendernode.path;

String newPath = ((iTreeNode)(this.sendernode.Parent)).path + "\\" + newName + ".xls";

if (File.Exists(newPath))

{

MessageBox.Show("已经存在这个文件！", "改名失败");

}

else

{

File.Move(path, newPath);

this.sendernode.path = newPath;

this.sendernode.Text = newName + ".xls";

}

}

else

{

//MessageBox.Show("输入值长度为0或包含非法字符，请重新输入！");

}

}

else

{

MessageBox.Show("只能给用例改名，不能给目录改名！");

}

}

//删除节点

void deleteNode(object sender, EventArgs e)

{

//文件节点

if (this.sendernode.ntype == NodeType.FILE)

{

DialogResult dr = MessageBox.Show("确认删除！", "警告", MessageBoxButtons.OKCancel);

if (dr == DialogResult.OK)

{

iTreeNode parent = (iTreeNode)this.sendernode.Parent;

String path = this.sendernode.path;

//////////Console.WriteLine(path);

File.Delete(path);

parent.Nodes.Remove(this.sendernode);

}

}

//目录节点

else

{

//空的目录节点

if (this.sendernode.Nodes.Count == 0)

{

DialogResult dr = MessageBox.Show("确认删除！", "警告", MessageBoxButtons.OKCancel);

{

iTreeNode parent = (iTreeNode)this.sendernode.Parent;

String path = this.sendernode.path;

new DirectoryInfo(path).Delete();

parent.Nodes.Remove(this.sendernode);

}

}

//非空的目录节点

else

{

MessageBox.Show("只能删除空的目录！", "提示");

}

}

}

//克隆节点

void cloneNode(object sender, EventArgs e)

{

//只有文件节点支持克隆

if (this.sendernode.ntype == NodeType.FILE)

{

String path = this.sendernode.path;

//////////Console.WriteLine(path);

String pathClone = path.Replace(".xls", "\_副本.xls");

try

{

File.Copy(path, pathClone);

iTreeNode nodeClone = new iTreeNode(form, pathClone, NodeType.FILE);

this.sendernode.Parent.Nodes.Add(nodeClone);

}

catch (Exception e3)

{

MessageBox.Show("可能目标文件已经存在！");

//////////Console.WriteLine(e3.Message);

}

}

else

{

MessageBox.Show("只支持克隆用例，不支持克隆目录！");

}

}

//运行一个节点

void runNode(object sender, EventArgs e)

{

// ////////Console.WriteLine(this.sendernode.Text);

List<String> runFileList = new List<string>();

//如果是一个目录节点

if (this.sendernode.ntype == NodeType.DIR)

{

ArrayList list = new iTreeNodeManager(this.form).getSelectedFileNodes(this.sendernode);

foreach (iTreeNode n in list)

{

//////////Console.WriteLine(n.path);

runFileList.Add(n.path);

}

}

//如果是一个文件节点

else

{

//////////Console.WriteLine(this.sendernode.path); // .sendernode.path

runFileList.Add(this.sendernode.path);

}

new Runner().xxxxxx(new FormAndList(this.form, runFileList));

}

}

//自定义的节点类

public class iTreeNode : TreeNode

{

Form1 form;

public NodeType ntype;

public String path;

//增加多个节点的方法

public void AddNodes(ArrayList nodes)

{

foreach (iTreeNode n in nodes)

{

this.Nodes.Add(n);

}

}

//得到孩子节点 返回一个 ArrayList

public ArrayList getChildNodes()

{

ArrayList nodes = new ArrayList();

String path = this.path;

DirectoryInfo dirinfo = new DirectoryInfo(path);

//遍历目录下的目录

DirectoryInfo[] child\_dirinfos = dirinfo.GetDirectories();

foreach (DirectoryInfo di in child\_dirinfos)

{

String p = di.FullName;

iTreeNode n = new iTreeNode(this.form, p, NodeType.DIR);

nodes.Add(n);

}

//遍历目录下的 电子表格 文件

FileInfo[] child\_fileinfos = dirinfo.GetFiles("\*.xls");

foreach (FileInfo fi in child\_fileinfos)

{

String p = fi.FullName;

iTreeNode n = new iTreeNode(this.form, p, NodeType.FILE);

nodes.Add(n);

}

return nodes;

}

//增加孩子节点

public void AddChildNodes()

{

//如果一个节点是文件夹节点 那么就得到这个节点的孩子节点 然后增加

if (this.ntype == NodeType.DIR)

{

ArrayList childnodes = getChildNodes();

this.AddNodes(childnodes);

}

}

//给孩子节点增加孩子节点

public void ChildNodesAddChildNodes()

{

this.AddChildNodes();

//遍历这个节点的孩子节点

foreach (iTreeNode itn in this.Nodes)

{

//递归调用

itn.ChildNodesAddChildNodes();

}

}

//构造函数

// path 文件夹或文件的全路径

// ntype 是文件夹 还是 文件

public iTreeNode(Form1 form, String path, NodeType ntype)

: base()

{

this.form = form;

this.ntype = ntype;

this.path = path;

this.Text = getShortName(path);

//增加右键菜单

this.ContextMenuStrip = new iContextMenuStrip(this.form, this);

}

//

public String getShortName(String path)

{

String[] tmp = path.Split('\\');

int l = tmp.Length;

return tmp[l - 1];

}

}

public class ActionLines : Panel

{

public void search(String keyWord)

{

int l = this.Controls.Count;

for (int i =0; i <l; i++)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

actionLine.search(keyWord);

}

}

public int searchAndReplace(String keyWord, String replaceWord)

{

int l = this.Controls.Count;

int count = 0;

for (int i = 0; i < l; i++)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

if (actionLine.searchAndReplace(keyWord, replaceWord))

{

count++;

}

}

return count;

}

public void recoverySearchColor()

{

int l = this.Controls.Count;

for (int i = 0; i < l; i++)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

actionLine.recoverySearchColor();

}

}

//移除所有空白的步骤

public void removeAllNull()

{

List<Int32> indexes = new List<int>();

int l = this.Controls.Count;

for (int i = l - 1; i > -1; i--)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

if (actionLine.comboBox\_key.Text=="")

{

indexes.Add(i);

}

}

foreach (Int32 index in indexes)

{

removeByIndex(index);

}

l = this.Controls.Count;

for (int i = 0; i < l; i++)

{

((ActionLine)(this.Controls[i])).label\_index.Text = (i + 1).ToString();

}

this.refreshActionList();

this.setIndent();

}

//移除所有不运行的步骤

public void removeAllNotRun()

{

List<Int32> indexes = new List<int>();

int l = this.Controls.Count;

for (int i = l - 1; i > -1; i--)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

if (actionLine.checkBox\_isRun.Checked == false)

{

indexes.Add(i);

}

}

foreach (Int32 index in indexes)

{

removeByIndex(index);

}

l = this.Controls.Count;

for (int i = 0; i < l; i++)

{

((ActionLine)(this.Controls[i])).label\_index.Text = (i + 1).ToString();

}

this.refreshActionList();

this.setIndent();

}

private void removeByIndex(int index)

{

int l = this.Controls.Count;

for (int i = index + 1; i < l; i++)

{

ActionLine actionLine = ((ActionLine)(this.Controls[i]));

actionLine.Top -= 23;

}

this.Controls.RemoveAt(index);

}

public void insertListInControls(List<ActionLine> list2, int index)

{

int l = this.Controls.Count;

if (index < l)

{

List<ActionLine> tmp = new List<ActionLine>();

int width = this.Controls[index].Width;

int top = this.Controls[index].Top;

//将从 i 到最后一个 actionLine 全部转移到 tmp

for (int i = index; i < l; i++)

{

// //Console.WriteLine(i);

ActionLine control = (ActionLine)(this.Controls[i]);

// //Console.WriteLine(control.label\_index.Text);

// //Console.WriteLine(control.comboBox\_key.Text);

// //Console.WriteLine(control.Top);

tmp.Add(control);

}

//将 从 i 到最后一个 actionLine 全部 移除

for (int i = 0; i < tmp.Count; i++)

{

this.Controls.RemoveAt(index);

}

//将要增加到增加到末尾

foreach (ActionLine control in list2)

{

control.Top = top;

this.Controls.Add(control);

top += 23;

}

foreach (ActionLine control in tmp)

{

// //Console.WriteLine("增加高的个数=" + list2.Count);

control.Top += 23 \* list2.Count;

// //Console.WriteLine("增加后到top=" + control.Top);

this.Controls.Add(control);

}

this.Height += 23 \* list2.Count;

l = this.Controls.Count;

// //Console.WriteLine("新的长度=" + l);

for (int i = index; i < l; i++)

{

((ActionLine)(this.Controls[i])).label\_index.Text = (i + 1).ToString();

}

this.refreshActionList();

this.setIndent();

}

else

{

int width = this.Controls[index-1].Width;

int top = this.Controls[index-1].Top+23;

//将要增加到增加到末尾

foreach (ActionLine control in list2)

{

control.Top = top;

this.Controls.Add(control);

top += 23;

}

this.Height += 23 \* list2.Count;

l = this.Controls.Count;

// //Console.WriteLine("新的长度=" + l);

for (int i = index; i < l; i++)

{

((ActionLine)(this.Controls[i])).label\_index.Text = (i + 1).ToString();

}

this.refreshActionList();

this.setIndent();

}

}

//设置缩进

public void setIndent()

{

foreach (ActionLine actionLine in this.Controls)

{

actionLine.setWidth(actionLine.Width);

}

int indent = 0;

foreach (ActionLine actionLine in this.Controls)

{

if ("{" == actionLine.comboBox\_key.Text && actionLine.checkBox\_isRun.Checked)

{

actionLine.setIndent(indent);

indent++;

}

else if ("}" == actionLine.comboBox\_key.Text && actionLine.checkBox\_isRun.Checked)

{

indent--;

actionLine.setIndent(indent);

}

else

{

actionLine.setIndent(indent);

}

}

}

Color currentColor = Color.FromArgb(88, 117, 184);

public List<Action> actionList;

//转换为二维list

public List<List<String>> toListList()

{

this.refreshActionList();

List<List<String>> listList = new List<List<string>>();

foreach (Action action in this.actionList)

{

List<String> list = new List<string>();

if (action.GetType().FullName.EndsWith("CreateDriverAction"))

{

list = ((CreateDriverAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("CreateAppiumDriverAction"))

{

list = ((CreateAppiumDriverAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("CreateIOSDriverAction"))

{

list = ((CreateIOSDriverAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("GoAction"))

{

list = ((GoAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("SetParameterAction"))

{

list = ((SetParameterAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("ExecuteFunctionAction"))

{

list = ((ExecuteFunctionAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("OneStepAction"))

{

list = ((OneStepAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("SleepAction"))

{

list = ((SleepAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("ExeOtherCaseAction"))

{

list = ((ExeOtherCaseAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("QuitAction"))

{

list = ((QuitAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("AnnotationAction"))

{

list = ((AnnotationAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("PythonCodeAction"))

{

list = ((PythonCodeAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("AddIndentAction"))

{

list = ((AddIndentAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("RecoverIndentAction"))

{

list = ((RecoverIndentAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("ReadExternalConfAction"))

{

list = ((ReadExternalConfAction)(action)).toList();

}

else if (action.GetType().FullName.EndsWith("NullAction"))

{

list = ((NullAction)(action)).toList();

}

listList.Add(list);

}

return listList;

}

//当前被选中的序号 从0开始

public int currentIndex;

//最后一行的top

public int top;

//最后一行的 index

public int i;

//设置当前选中行的颜色

public void setCurrentColor()

{

//this.Controls 就是一些 ActionLine ，现将所有的 ActionLine 的背景设置为白色

foreach (Control control in this.Controls)

{

((ActionLine)(control)).BackColor = Color.White;

}

if (currentIndex > -1)

{

//如果越界了 ，就到最后一行

if (currentIndex > this.actionList.Count - 1)

{

currentIndex = this.actionList.Count - 1;

}

((ActionLine)(this.Controls[currentIndex])).BackColor = currentColor;

}

else

{

if (this.actionList.Count > 0)

{

currentIndex = 0;

((ActionLine)(this.Controls[currentIndex])).BackColor = currentColor; //Color.FromArgb(88, 117, 184);

}

}

}

//

// public List<Action> actionList;

//用界面刷新list

public void refreshActionList()

{

actionList.Clear();

foreach (ActionLine actionLine in this.Controls)

{

actionLine.refreshAction();

actionList.Add(actionLine.action);

}

}

//设置 ActionLine 的宽度

public void setActionLineWidth(int width)

{

this.Width = width;

int height = 0;

foreach (ActionLine actionLine in this.Controls)

{

actionLine.setWidth(width);

height += actionLine.Height;

}

}

//用list刷新界面

public void refreshGui()

{

this.Visible = false;

//Console.WriteLine("refreshGui开始的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

DateTime d1 = DateTime.Now;

this.Controls.Clear();

top = 0;

i = 0;

foreach (Action action in actionList)

{

action.index = i + 1;

// //Console.WriteLine("new ActionLine开始的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

ActionLine actionLine = new ActionLine(action, this.Width);

// //Console.WriteLine("new ActionLine结束的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

actionLine.Top += top;

top += 23;

// //Console.WriteLine("Controls.Add开始的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

this.Controls.Add(actionLine);

// //Console.WriteLine("Controls.Add结束的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

i++;

}

this.Height = top;

setCurrentColor();

//Console.WriteLine("refreshGui结束的时间点" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

DateTime d2 = DateTime.Now;

this.setIndent();

Console.WriteLine("设置为可见之前" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

this.Visible = true;

this.form.hideLoading();

Console.WriteLine("设置为可见之后" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

}

public Form1 form;

//构造方法

public ActionLines(List<Action> list, int width ,Form1 form)

: base()

{

Control.CheckForIllegalCrossThreadCalls = false;

this.form = form;

this.DoubleBuffered = true;

SetStyle(ControlStyles.ResizeRedraw, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

SetStyle(ControlStyles.AllPaintingInWmPaint, true);

this.Width = width;

this.actionList = list;

if (list.Count == 0)

{

this.currentIndex = -1;

}

refreshGui();

}

public void currentIndexAdd()

{

if (this.currentIndex < this.Controls.Count - 1)

{

this.currentIndex++;

}

this.setCurrentColor();

}

public void currentIndexReduce()

{

if (this.currentIndex > 0)

{

this.currentIndex--;

}

this.setCurrentColor();

}

//在某行插入一个空行

public void insertActionLine(int index, int num)

{

refreshActionList();

List<Action> list1 = this.actionList.GetRange(0, index);

List<Action> list2 = this.actionList.GetRange(index, actionList.Count - index);

NullAction nullAction = new NullAction(0, "", "", true);

actionList.Clear();

actionList.AddRange(list1);

for (int i = 0; i < num; i++)

{

actionList.Add(nullAction);

}

actionList.AddRange(list2);

//refreshGui();

insertRefreshGui(index, num);

this.setCurrentColor();

}

public void deleteRefreshGui(int index)

{

int leng = this.Controls.Count;

for (int i2 = index + 1; i2 < leng; i2++)

{

ActionLine actionLine = (ActionLine)Controls[i2];

actionLine.indexReduce(1);

}

this.Controls.RemoveAt(index);

this.Height -= 23;

this.setCurrentColor();

this.setIndent();

}

public void upRefreshGui(int index)

{

int leng = this.Controls.Count;

List<Control> list = new List<Control>();

foreach (Control control in this.Controls)

{

list.Add(control);

}

((ActionLine)list[index]).indexReduce(1);

((ActionLine)list[index - 1]).indexAdd(1);

Control tmp = list[index];

list[index] = list[index - 1];

list[index - 1] = tmp;

//this.Controls.Clear();

for (int i = index - 1; i < leng; i++)

{

this.Controls.RemoveAt(index - 1);

}

for (int i = index - 1; i < leng; i++)

{

this.Controls.Add(list[i]);

}

this.setCurrentColor();

this.setIndent();

}

public void downRefreshGui(int index)

{

int leng = this.Controls.Count;

List<Control> list = new List<Control>();

foreach (Control control in this.Controls)

{

list.Add(control);

}

((ActionLine)list[index]).indexAdd(1);

((ActionLine)list[index + 1]).indexReduce(1);

Control tmp = list[index];

list[index] = list[index + 1];

list[index + 1] = tmp;

for (int i = index; i < leng; i++)

{

this.Controls.RemoveAt(index);

}

for (int i = index; i < leng; i++)

{

this.Controls.Add(list[i]);

}

this.setCurrentColor();

this.setIndent();

}

public void insertRefreshGui(int index, int num)

{

int width = ((ActionLine)Controls[index]).Width;

int top = ((ActionLine)Controls[index]).Top;

List<ActionLine> addList = new List<ActionLine>();

for (int i = 0; i < num; i++)

{

NullAction nullAction = new NullAction(index + 1 + i, "", "", true);

ActionLine nullLine = new ActionLine(nullAction, width);

nullLine.Top = top + i \* 23;

addList.Add(nullLine);

}

List<ActionLine> tmpList = new List<ActionLine>();

int leng = this.Controls.Count;

for (int i2 = index; i2 < leng; i2++)

{

ActionLine actionLine = (ActionLine)Controls[index];

actionLine.indexAdd(num);

tmpList.Add(actionLine);

Controls.Remove(actionLine);

}

this.Height += 23 \* num;

foreach (ActionLine tmp in addList)

{

this.Controls.Add(tmp);

}

foreach (ActionLine tmp in tmpList)

{

this.Controls.Add(tmp);

}

this.setIndent();

}

//最后一行增加一个空行

public void addActionLineAtEnd()

{

NullAction nullAction = new NullAction(0, "", "", true);

nullAction.index = i + 1;

ActionLine nullActionLine = new ActionLine(nullAction, this.Width);

nullActionLine.Top = top;

top += 20;

i++;

this.Controls.Add(nullActionLine);

this.Height = top;

this.currentIndex = i - 1;

this.setCurrentColor();

}

public void deleteActionLine(int index)

{

refreshActionList();

this.actionList.RemoveAt(index);

this.deleteRefreshGui(index);

// refreshGui();

}

//某行上移

public void up(int index)

{

List<Action> list = this.actionList;

if (index > 0 && index < list.Count)

{

Action tmpAction = list[index];

list[index] = list[index - 1];

list[index - 1] = tmpAction;

this.currentIndex = index - 1;

this.upRefreshGui(index);

}

else if (index == 0)

{

MessageBox.Show("已经是第一个");

}

else

{

MessageBox.Show("index太大");

}

}

//某行下移

public void down(int index)

{

List<Action> list = this.actionList;

if (index >= 0 && index < list.Count - 1)

{

Action tmpAction = list[index];

list[index] = list[index + 1];

list[index + 1] = tmpAction;

this.currentIndex = index + 1;

this.downRefreshGui(index);

}

else if (index == list.Count - 1)

{

MessageBox.Show("已经是最后一个");

}

else

{

MessageBox.Show("index太小");

}

}

public void thisAndAfterNotRun(int index)

{

int leng = this.Controls.Count;

for (int i = index; i < leng; i++)

{

ActionLine actionLine = (ActionLine)(this.Controls[i]);

actionLine.setColorByIsRun(false);

}

}

public void thisAndAfterNotRun(int index, int index2)

{

int leng = this.Controls.Count;

if (index2 >= leng)

{

index2 = leng;

}

for (int i = index; i < index2; i++)

{

ActionLine actionLine = (ActionLine)(this.Controls[i]);

actionLine.setColorByIsRun(false);

}

}

public void thisAndAfterRecoveryRun(int index)

{

int leng = this.Controls.Count;

for (int i = index; i < leng; i++)

{

ActionLine actionLine = (ActionLine)(this.Controls[i]);

actionLine.setColorByIsRun(true);

}

}

public void thisAndAfterRecoveryRun(int index, int index2)

{

int leng = this.Controls.Count;

if (index2 >= leng)

{

index2 = leng;

}

for (int i = index; i < index2; i++)

{

ActionLine actionLine = (ActionLine)(this.Controls[i]);

actionLine.setColorByIsRun(true);

}

}

}

public delegate void XXXX();

public class EtagContextMenuStrip : ContextMenuStrip

{

ETabPage clickedETabPage;

public Form1 form;

public EtagContextMenuStrip(Form1 form, ETabPage t)

: base()

{

this.form = form;

clickedETabPage = t;

//////////Console.WriteLine(t.Name);

ToolStripMenuItem item\_runThis = new ToolStripMenuItem("运行本用例");

ToolStripMenuItem item\_runAllOpened = new ToolStripMenuItem("运行所有已打开用例");

ToolStripMenuItem item\_closeThis = new ToolStripMenuItem("关闭本用例");

ToolStripMenuItem item\_closeOthers = new ToolStripMenuItem("关闭其他用例");

ToolStripMenuItem item\_closeAll = new ToolStripMenuItem("关闭所有用例");

item\_runThis.Click += new EventHandler(item\_runThis\_Click);

item\_runAllOpened.Click += new EventHandler(item\_runAllOpened\_Click);

item\_closeThis.Click += new EventHandler(item\_closeThis\_Click);

item\_closeOthers.Click += new EventHandler(item\_closeOther\_Click);

item\_closeAll.Click += new EventHandler(item\_closeAll\_Click);

this.Items.Add(item\_runThis);

this.Items.Add(item\_runAllOpened);

this.Items.Add(item\_closeThis);

this.Items.Add(item\_closeOthers);

this.Items.Add(item\_closeAll);

}

void item\_runThis\_Click(object sender, EventArgs e)

{

String filePath = clickedETabPage.Name;

List<String> list = new List<String>();

list.Add(filePath);

new Runner().xxxxxx(new FormAndList(this.form, list));

}

void item\_runAllOpened\_Click(object sender, EventArgs e)

{

List<String> list = new List<String>();

ControlCollection eTabPages = clickedETabPage.Parent.Controls;

foreach (ETabPage eTabPage in eTabPages)

{

list.Add(eTabPage.Name);

}

new Runner().xxxxxx(new FormAndList(this.form, list));

}

//关闭当前的这个 page

void item\_closeThis\_Click(object sender, EventArgs e)

{

ControlCollection eTabPages = clickedETabPage.Parent.Controls;

foreach (ETabPage eTabPage in eTabPages)

{

if (eTabPage.Equals(clickedETabPage))

{

((TabControl)(clickedETabPage.Parent)).TabPages.Remove(eTabPage);

break;

}

}

}

//关闭除这个之外的所有page

void item\_closeOther\_Click(object sender, EventArgs e)

{

List<ETabPage> closeList = new List<ETabPage>();

ControlCollection eTabPages = clickedETabPage.Parent.Controls;

foreach (ETabPage eTabPage in eTabPages)

{

if (!eTabPage.Equals(clickedETabPage))

{

closeList.Add(eTabPage);

}

}

foreach (ETabPage tmp in closeList)

{

((TabControl)(clickedETabPage.Parent)).TabPages.Remove(tmp);

}

}

//关闭所有用例

void item\_closeAll\_Click(object sender, EventArgs e)

{

ControlCollection eTabPages = clickedETabPage.Parent.Controls;

int l = eTabPages.Count;

TabControl tabControl = (TabControl)(clickedETabPage.Parent);

for (int i = 0; i < l; i++)

{

tabControl.TabPages.RemoveAt(0);

}

}

}

public class ActionLineContextMenuStrip : ContextMenuStrip

{

ActionLine clickedActionLine;

void item\_add\_befor\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.currentIndex = clickedIndex;

actionLines.insertActionLine(clickedIndex, 1);

}

void item\_ImportStepsFromXls\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.currentIndex = clickedIndex;

int width = actionLines.Controls[clickedIndex].Width;

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = @"xls文件|\*.xls";

openFileDialog.ShowDialog();

String fileName = openFileDialog.FileName;

// //Console.WriteLine(fileName);

if ("" != fileName)

{

try

{

List<Action> actionList = new ExcelProcessor().readActionListFromExcelFile(fileName);

List<ActionLine> actionLineList = new List<ActionLine>();

actionLineList.Add(new ActionLine(new NullAction(0, "", "", true), width));

foreach (Action action in actionList)

{

ActionLine actionLine = new ActionLine(action, width);

actionLineList.Add(actionLine);

}

actionLineList.Add(new ActionLine(new NullAction(0, "", "", true), width));

actionLines.insertListInControls(actionLineList, clickedIndex);

}

catch (Exception e10)

{

MessageBox.Show("请不要用其他软件同时打开用例文件！", "提示");

}

}

}

public void item\_Paste\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

Form1 form = (Form1)(clickedActionLine.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent);

actionLines.insertListInControls(form.actionLineListClipboard, clickedIndex);

}

public void item\_Copy\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

ActionLine actionLine = ((ActionLine)(actionLines.Controls[clickedIndex]));

Form1 form = (Form1)(clickedActionLine.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent);

form.actionLineListClipboard.Clear();

form.actionLineListClipboard.Add(actionLine.deepCopy());

}

public void item\_QuicklyCopyALine\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

ActionLine actionLine = ((ActionLine)(actionLines.Controls[clickedIndex]));

List<ActionLine> tmpList = new List<ActionLine>();

tmpList.Add(actionLine.deepCopy());

actionLines.insertListInControls(tmpList, clickedIndex+1);

}

public void item\_CopyMul\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

String stepNum = Interaction.InputBox("复制从当前步骤至第几个步骤", "请输入一个数字");

int step2 = 0;

try

{

step2 = Int32.Parse(stepNum);

if (step2 > actionLines.Controls.Count)

{

step2 = actionLines.Controls.Count;

}

int toIndex = step2 - 1;

Form1 form = (Form1)(clickedActionLine.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent.Parent);

form.actionLineListClipboard.Clear();

if (toIndex >= clickedIndex)

{

for (int i = clickedIndex; i <= toIndex; i++)

{

ActionLine actionLine = ((ActionLine)(actionLines.Controls[i]));

form.actionLineListClipboard.Add(actionLine.deepCopy());

}

}

}

catch (Exception e10)

{

}

}

void item\_AddStepsOfRunByXls\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.currentIndex = clickedIndex;

List<ActionLine> list = new List<ActionLine>();

int width = actionLines.Controls[clickedIndex].Width;

list.Add(new ActionLine(new OneStepAction(0, "读取多行数据到序列", "电子表格文件的路径||多行测试数据", true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "allPass=True", true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "for data in self.getPar('多行测试数据'):", true), width));

list.Add(new ActionLine(new AddIndentAction(0, true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "self.addDic(data)", true), width));

list.Add(new ActionLine(new AnnotationAction(0, "请在这里增加循环执行的内容", true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "allPass=allPass and self.getPar('LASTSTEP')", true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "self.setPar('LASTSTEP',True)", true), width));

list.Add(new ActionLine(new RecoverIndentAction(0, true), width));

list.Add(new ActionLine(new PythonCodeAction(0, "self.setPar('LASTSTEP',allPass)", true), width));

actionLines.insertListInControls(list, clickedIndex);

}

void item\_add5\_befor\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.currentIndex = clickedIndex;

actionLines.insertActionLine(clickedIndex, 5);

}

void item\_delete\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.currentIndex = clickedIndex;

actionLines.deleteActionLine(clickedIndex);

}

void item\_up\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.up(clickedIndex);

}

void item\_down\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.down(clickedIndex);

}

void item\_ThisAndAfterNotRun\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.thisAndAfterNotRun(clickedIndex);

}

void item\_ThisAndAfterRecoveryRun\_Click(object sender, EventArgs e)

{

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.thisAndAfterRecoveryRun(clickedIndex);

}

void item\_ThisToDesignatedNotRun\_Click(object sender, EventArgs e)

{

String stepNum = Interaction.InputBox("从当前步骤至第几个步骤被禁用", "请输入一个数字");

int step2 = 0;

try

{

step2 = Int32.Parse(stepNum);

}

catch (Exception e10)

{

}

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.thisAndAfterNotRun(clickedIndex, step2);

}

void item\_ThisToDesignatedRecoveryRun\_Click(object sender, EventArgs e)

{

String stepNum = Interaction.InputBox("从当前步骤至第几个步骤被恢复", "请输入一个数字");

int step2 = 0;

try

{

step2 = Int32.Parse(stepNum);

}

catch (Exception e10)

{

}

int clickedIndex = clickedActionLine.action.index - 1;

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.thisAndAfterRecoveryRun(clickedIndex, step2);

}

//移除所有不跑的行

void item\_RemoveAllNotRun\_Click(object sender, EventArgs e)

{

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

DialogResult dr = MessageBox.Show("删除后不可恢复，确定？", "警告", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr == DialogResult.OK)

{

actionLines.removeAllNotRun();

}

}

//移除所有空白到行

void item\_RemoveAllNull\_Click(object sender, EventArgs e)

{

ActionLines actionLines = (ActionLines)clickedActionLine.Parent;

actionLines.removeAllNull();

}

public ActionLineContextMenuStrip(ActionLine actionLine)

: base()

{

////Console.WriteLine("ActionLineContextMenuStrip构造方法开始" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

clickedActionLine = actionLine;

ToolStripMenuItem item\_Copy = new ToolStripMenuItem("复制");

ToolStripMenuItem item\_CopyMul = new ToolStripMenuItem("复制多行");

ToolStripMenuItem item\_Paste = new ToolStripMenuItem("粘贴");

ToolStripMenuItem item\_QuicklyCopyALine = new ToolStripMenuItem("快速复制一行");

ToolStripMenuItem item\_add\_befor = new ToolStripMenuItem("插入一个步骤 Ctrl+I");

ToolStripMenuItem item\_add5\_befor = new ToolStripMenuItem("插入五个步骤");

ToolStripMenuItem item\_delete = new ToolStripMenuItem("删除这个步骤 Ctrl+D");

ToolStripMenuItem item\_up = new ToolStripMenuItem("上移 Ctrl+上");

ToolStripMenuItem item\_down = new ToolStripMenuItem("下移 Ctrl+下");

ToolStripMenuItem item\_ThisAndAfterNotRun = new ToolStripMenuItem("禁用以下所有步骤");

ToolStripMenuItem item\_ThisAndAfterRecoveryRun = new ToolStripMenuItem("恢复以下所有步骤");

ToolStripMenuItem item\_ThisToDesignatedNotRun = new ToolStripMenuItem("禁用多个步骤");

ToolStripMenuItem item\_ThisToDesignatedRecoveryRun = new ToolStripMenuItem("恢复多个步骤");

ToolStripMenuItem item\_RemoveAllNotRun = new ToolStripMenuItem("删除所有禁用的步骤");

ToolStripMenuItem item\_RemoveAllNull = new ToolStripMenuItem("删除所有空的步骤");

ToolStripMenuItem item\_AddStepsOfRunByXls = new ToolStripMenuItem("增加根据数据表执行的前后python步骤");

ToolStripMenuItem item\_ImportStepsFromXls = new ToolStripMenuItem("导入步骤");

item\_add\_befor.Click += new EventHandler(item\_add\_befor\_Click);

item\_add5\_befor.Click += new EventHandler(item\_add5\_befor\_Click);

item\_delete.Click += new EventHandler(item\_delete\_Click);

item\_up.Click += new EventHandler(item\_up\_Click);

item\_down.Click += new EventHandler(item\_down\_Click);

item\_ThisAndAfterNotRun.Click += new EventHandler(item\_ThisAndAfterNotRun\_Click);

item\_ThisAndAfterRecoveryRun.Click += new EventHandler(item\_ThisAndAfterRecoveryRun\_Click);

item\_ThisToDesignatedNotRun.Click += new EventHandler(item\_ThisToDesignatedNotRun\_Click);

item\_ThisToDesignatedRecoveryRun.Click += new EventHandler(item\_ThisToDesignatedRecoveryRun\_Click);

item\_RemoveAllNotRun.Click += new EventHandler(item\_RemoveAllNotRun\_Click);

item\_RemoveAllNull.Click+=new EventHandler(item\_RemoveAllNull\_Click);

item\_AddStepsOfRunByXls.Click += new EventHandler(item\_AddStepsOfRunByXls\_Click);

item\_Copy.Click+=new EventHandler(item\_Copy\_Click);

item\_CopyMul.Click+=new EventHandler(item\_CopyMul\_Click);

item\_Paste.Click+=new EventHandler(item\_Paste\_Click);

item\_QuicklyCopyALine.Click+=new EventHandler(item\_QuicklyCopyALine\_Click);

item\_ImportStepsFromXls.Click+=new EventHandler(item\_ImportStepsFromXls\_Click);

//复制和粘贴

this.Items.Add(item\_Copy);

this.Items.Add(item\_CopyMul);

this.Items.Add(item\_Paste);

this.Items.Add(item\_QuicklyCopyALine);

//插入和删除

this.Items.Add(" ");

this.Items.Add(item\_add\_befor);

this.Items.Add(item\_add5\_befor);

this.Items.Add(item\_delete);

//上移 下移

this.Items.Add(" ");

this.Items.Add(item\_up);

this.Items.Add(item\_down);

//禁用和恢复

this.Items.Add(" ");

this.Items.Add(item\_ThisToDesignatedNotRun);

this.Items.Add(item\_ThisToDesignatedRecoveryRun);

this.Items.Add(item\_ThisAndAfterNotRun);

this.Items.Add(item\_ThisAndAfterRecoveryRun);

this.Items.Add(" ");

this.Items.Add(item\_RemoveAllNotRun);

this.Items.Add(item\_RemoveAllNull);

//快捷增加代码块

this.Items.Add(" ");

this.Items.Add(item\_AddStepsOfRunByXls);

this.Items.Add(item\_ImportStepsFromXls);

////Console.WriteLine("ActionLineContextMenuStrip构造方法结束" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

}

}

public class ActionLine : Panel

{

List<OneStepMenuAndPrompt> oneStepMenuAndPrompts;

public void search(String keyWord)

{

if (this.control\_par2.Text.Contains(keyWord))

{

this.control\_par2.BackColor = Color.Yellow;

}

}

public bool searchAndReplace(String keyWord ,String replaceWord )

{

if (this.control\_par2.Text.Contains(keyWord))

{

this.control\_par2.BackColor = Color.Yellow;

this.control\_par2.Text = this.control\_par2.Text.Replace(keyWord, replaceWord);

return true;

}

else

{

return false;

}

}

public void recoverySearchColor()

{

this.control\_par2.BackColor = Color.White;

this.setColorByIsRun( this.checkBox\_isRun.Checked );

}

public ActionLine deepCopy()

{

Action action = this.action;

int width = this.Width;

return new ActionLine(action, width);

}

Dictionary<String, String> oneStepNameAndPrompt = new Dictionary<string, string>();

public void refreshAction()

{

int index = Int32.Parse(this.label\_index.Text);

String key = this.comboBox\_key.Text;

String par1 = this.control\_par1.Text;

String par2 = this.control\_par2.Text;

bool isRun = this.checkBox\_isRun.Checked;

if ("新建驱动" == key)

{

this.action = new CreateDriverAction(index, par1, par2, isRun);

}

else if ("新建Appium驱动" == key)

{

this.action = new CreateAppiumDriverAction(index, par1, par2, isRun);

}

else if ("新建IOS驱动" == key)

{

this.action = new CreateIOSDriverAction(index, par1, par2, isRun);

}

else if ("打开网址" == key)

{

this.action = new GoAction(index, par1, par2, isRun);

}

else if ("设置参数" == key)

{

this.action = new SetParameterAction(index, par1, par2, isRun);

}

else if ("执行方法" == key)

{

this.action = new ExecuteFunctionAction(index, par1, par2, isRun);

}

else if ("单步方法" == key)

{

this.action = new OneStepAction(index, par1, par2, isRun);

}

else if ("暂停" == key)

{

this.action = new SleepAction(index, Int32.Parse(par1), isRun);

}

else if ("调用其他用例" == key)

{

this.action = new ExeOtherCaseAction(index, par2, isRun);

}

else if ("退出" == key)

{

this.action = new QuitAction(index, par1, isRun);

}

else if ("注释" == key)

{

this.action = new AnnotationAction(index, par2, isRun);

}

else if ("Python代码" == key)

{

this.action = new PythonCodeAction(index, par2, isRun);

}

else if ("{" == key)

{

this.action = new AddIndentAction(index, isRun);

}

else if ("}" == key)

{

this.action = new RecoverIndentAction(index, isRun);

}

else if ("读取外部配置" == key)

{

this.action = new ReadExternalConfAction(index, par2, isRun);

}

else if ("" == key)

{

this.action = new NullAction(index, par1, par2, true);

}

}

public void setIndent(int indent)

{

int oneIndentWidth = 32;

int indentWidth = indent \* oneIndentWidth;

this.comboBox\_key.Left += indentWidth;

this.control\_par1.Left += indentWidth;

this.control\_par2.Left += indentWidth;

this.control\_par2.Width -= indent \* oneIndentWidth;

}

public void setColorByIsRun(bool isRun)

{

if (isRun)

{

this.checkBox\_isRun.Checked = true;

control\_par1.BackColor = Color.White;

control\_par2.BackColor = Color.White;

}

else

{

this.checkBox\_isRun.Checked = false;

control\_par1.BackColor = Color.Gray;

control\_par2.BackColor = Color.Gray;

}

}

public void checkBox\_isRun\_CheckedChanged(object sender, EventArgs e)

{

CheckBox checkBox = (CheckBox)sender;

setColorByIsRun(checkBox.Checked);

}

//被点击

public void ActionLine\_Click(object sender, EventArgs e)

{

((ActionLines)(this.Parent)).currentIndex = this.action.index - 1;

((ActionLines)(this.Parent)).setCurrentColor();

}

public Action action;

public Label label\_index;

public ComboBox comboBox\_key;

public Control control\_par1;

public Control control\_par2;

public CheckBox checkBox\_isRun;

public void indexAdd(int x)

{

for (int i = 0; i < x; i++)

{

this.Top += 23;

}

this.label\_index.Text = (Int32.Parse(this.label\_index.Text) + x).ToString();

this.action.index += x;

}

public void indexReduce(int x)

{

for (int i = 0; i < x; i++)

{

this.Top -= 23;

}

this.label\_index.Text = (Int32.Parse(this.label\_index.Text) - x).ToString();

this.action.index -= x;

}

public void setWidth(int width)

{

this.Width = width;

int width\_label\_index = 30;

int width\_comboBox\_key = (width - 50) \* 10 / 100;

int width\_textBox\_par1 = (width - 50) \* 15 / 100;

int width\_control\_par2 = (width - 50) \* 75 / 100;

int width\_checkBox\_isRun = 20;

label\_index.Left = 0;

label\_index.Top = 1;

label\_index.Width = width\_label\_index;

comboBox\_key.Left = label\_index.Left + label\_index.Width;

comboBox\_key.Top = 1;

comboBox\_key.Width = width\_comboBox\_key;

control\_par1.Left = comboBox\_key.Left + comboBox\_key.Width;

control\_par1.Top = 1;

control\_par1.Width = width\_textBox\_par1;

control\_par2.Left = control\_par1.Left + control\_par1.Width;

control\_par2.Top = 1;

control\_par2.Width = width\_control\_par2;

checkBox\_isRun.Left = control\_par2.Left + control\_par2.Width;

checkBox\_isRun.Top = 1;

checkBox\_isRun.Width = width\_checkBox\_isRun;

}

private void comboBox\_oneStep\_SelectedIndexChanged(object sender, EventArgs e)

{

String oneStepName = ((ComboBox)control\_par1).Text;

// //Console.WriteLine(oneStepName);

control\_par2.Text = this.oneStepNameAndPrompt[oneStepName];

}

private void comboBox\_key\_SelectedIndexChanged(object sender, EventArgs e)

{

//////Console.WriteLine("更改key");

Point lacation1 = control\_par1.Location;

Size size1 = control\_par1.Size;

Color backColor1 = control\_par1.BackColor;

Point lacation2 = control\_par2.Location;

Size size2 = control\_par2.Size;

Color backColor2 = control\_par2.BackColor;

String key = comboBox\_key.Text;

this.Controls.Remove(control\_par1);

this.Controls.Remove(control\_par2);

if ("执行方法" == key)

{

this.control\_par1 = new TextBox();

this.setPar2Items();

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

((ComboBox)sender).BackColor = KeyColor.ExecuteFunctionActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.ExecuteFunctionActionFontColor;

}

else if ("单步方法" == key)

{

this.setPar1Items();

this.control\_par2 = new TextBox();

((ComboBox)control\_par1).SelectedIndexChanged += comboBox\_oneStep\_SelectedIndexChanged;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = true;

((ComboBox)sender).BackColor = KeyColor.OneStepActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.OneStepActionFontColor;

}

else

{

this.control\_par1 = new TextBox();

this.control\_par2 = new TextBox();

if ("新建驱动" == key)

{

((ComboBox)sender).BackColor = KeyColor.CreateDriverActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.CreateDriverActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = false;

}

else if ("打开网址" == key)

{

((ComboBox)sender).BackColor = KeyColor.GoActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.GoActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if ("暂停" == key)

{

((ComboBox)sender).BackColor = KeyColor.SleepActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.SleepActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = false;

}

else if ("设置参数" == key)

{

((ComboBox)sender).BackColor = KeyColor.SetParameterActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.SetParameterActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = true;

}

else if ("调用其他用例" == key)

{

((ComboBox)sender).BackColor = KeyColor.ExeOtherCaseActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.ExeOtherCaseActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if ("退出" == key)

{

((ComboBox)sender).BackColor = KeyColor.QuitActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.QuitActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if ("注释" == key)

{

this.control\_par2 = new AnnotationTextBox();

((ComboBox)sender).BackColor = KeyColor.AnnotationActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.AnnotationActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if ("Python代码" == key)

{

this.control\_par2 = new PythonTextBox();

((ComboBox)sender).BackColor = KeyColor.PythonCodeActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if ("{" == key)

{

((ComboBox)sender).BackColor = KeyColor.PythonCodeActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if ("}" == key)

{

((ComboBox)sender).BackColor = KeyColor.PythonCodeActionBackColor;

((ComboBox)sender).ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if ("读取外部配置" == key)

{

((ComboBox)sender).BackColor = KeyColor.ReadExternalConfBackAction;

((ComboBox)sender).ForeColor = KeyColor.ReadExternalConfFontAction;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

}

control\_par1.Location = lacation1;

control\_par1.Size = size1;

control\_par1.BackColor = backColor1;

control\_par2.Location = lacation2;

control\_par2.Size = size2;

control\_par2.BackColor = backColor2;

this.Controls.Add(control\_par1);

this.Controls.Add(control\_par2);

}

public void setPar1Items()

{

this.control\_par1 = new MyComboBox();

((ComboBox)control\_par1).DropDownStyle = ComboBoxStyle.DropDownList;

((ComboBox)control\_par1).FlatStyle = FlatStyle.Flat;

foreach (OneStepMenuAndPrompt oneStepMenuAndPrompt in oneStepMenuAndPrompts)

{

((ComboBox)control\_par1).Items.Add(oneStepMenuAndPrompt.menuText);

}

}

public void setPar2Items()

{

String prPath = new TextProcessor().getAbsPath();

this.control\_par2 = new MyComboBox();

((ComboBox)control\_par2).DropDownStyle = ComboBoxStyle.DropDownList;

((ComboBox)control\_par2).FlatStyle = FlatStyle.Flat;

// List<String> funList = new TextProcessor().getFuns(prPath + "\\conf\\function.txt");

// List<String> funList =

Dictionary<String, String> functionDic = new TextProcessor().createFunctionDic(prPath + "\\PythonSelenium2\\src\\testPage");

foreach (KeyValuePair<String, String> kv in functionDic)

{

String fun = kv.Key;

((ComboBox)control\_par2).Items.Add(fun);

}

/\*

foreach (String fun in funList)

{

((ComboBox)control\_par2).Items.Add(fun);

}

\* \*/

}

//构造方法

public ActionLine(Action action, int width)

: base()

{

Control.CheckForIllegalCrossThreadCalls = false;

this.DoubleBuffered = true;

SetStyle(ControlStyles.ResizeRedraw, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

SetStyle(ControlStyles.AllPaintingInWmPaint, true);

//Console.WriteLine("初始化ActionLine开始" + DateTime.Now.ToString("yyyyMMddHHmmssfff"));

TextProcessor tp = new TextProcessor();

String prPath = tp.getAbsPath();

String oneStepConfFile = prPath + "\\conf\\onestep.txt";

oneStepMenuAndPrompts = tp.getOneStepMenuAndPromptList(oneStepConfFile);

foreach (OneStepMenuAndPrompt oneStepMenuAndPrompt in oneStepMenuAndPrompts)

{

this.oneStepNameAndPrompt.Add(oneStepMenuAndPrompt.menuText, oneStepMenuAndPrompt.promptText);

}

//Console.WriteLine("初始化ActionLine结束" + DateTime.Now.ToString("yyyyMMddHHmmssfff"));

this.action = action;

int index = action.index;

String key = "";

String par1 = "";

String par2 = "";

bool isRun = action.isRun;

this.BackColor = Color.White;

this.Click += new EventHandler(ActionLine\_Click);

this.ContextMenuStrip = new ActionLineContextMenuStrip(this);

if (action.GetType().FullName.EndsWith("CreateDriverAction"))

{

key = "新建驱动";

par1 = ((CreateDriverAction)(action)).browserName;

par2 = ((CreateDriverAction)(action)).driverName;

}

else if (action.GetType().FullName.EndsWith("CreateAppiumDriverAction"))

{

key = "新建Appium驱动";

par1 = ((CreateAppiumDriverAction)(action)).deviceName;

par2 = ((CreateAppiumDriverAction)(action)).appPackageAndAppActivity;

}

else if (action.GetType().FullName.EndsWith("CreateIOSDriverAction"))

{

key = "新建IOS驱动";

par1 = ((CreateIOSDriverAction)(action)).macIP;

par2 = ((CreateIOSDriverAction)(action)).appPath;

}

else if (action.GetType().FullName.EndsWith("GoAction"))

{

key = "打开网址";

par1 = ((GoAction)(action)).driverName;

par2 = ((GoAction)(action)).url;

}

else if (action.GetType().FullName.EndsWith("SetParameterAction"))

{

key = "设置参数";

par1 = ((SetParameterAction)(action)).parName;

par2 = ((SetParameterAction)(action)).parValue;

}

else if (action.GetType().FullName.EndsWith("ExecuteFunctionAction"))

{

key = "执行方法";

par1 = ((ExecuteFunctionAction)(action)).driverName;

par2 = ((ExecuteFunctionAction)(action)).functionName;

}

else if (action.GetType().FullName.EndsWith("OneStepAction"))

{

key = "单步方法";

par1 = ((OneStepAction)(action)).oneStepName;

par2 = ((OneStepAction)(action)).oneStepPars;

}

else if (action.GetType().FullName.EndsWith("SleepAction"))

{

key = "暂停";

par1 = ((SleepAction)(action)).timeLong.ToString();

par2 = "";

}

else if (action.GetType().FullName.EndsWith("ExeOtherCaseAction"))

{

key = "调用其他用例";

par1 = "";

par2 = ((ExeOtherCaseAction)(action)).casePath; ;

}

else if (action.GetType().FullName.EndsWith("QuitAction"))

{

key = "退出";

par1 = ((QuitAction)(action)).driverName;

par2 = "";

}

else if (action.GetType().FullName.EndsWith("AnnotationAction"))

{

key = "注释";

par1 = "";

par2 = ((AnnotationAction)(action)).annotationText;

}

else if (action.GetType().FullName.EndsWith("PythonCodeAction"))

{

key = "Python代码";

par1 = "";

par2 = ((PythonCodeAction)action).pcode;

}

else if (action.GetType().FullName.EndsWith("AddIndentAction"))

{

key = "{";

par1 = "";

par2 = "";

}

else if (action.GetType().FullName.EndsWith("RecoverIndentAction"))

{

key = "}";

par1 = "";

par2 = "";

}

else if (action.GetType().FullName.EndsWith("ReadExternalConfAction"))

{

key = "读取外部配置";

par1 = "";

par2 = ((ReadExternalConfAction)action).excelPath;

}

else if (action.GetType().FullName.EndsWith("NullAction"))

{

key = "";

par1 = ((NullAction)(action)).par1;

par2 = ((NullAction)(action)).par2;

}

label\_index = new Label();

label\_index.Text = index.ToString();

label\_index.Click += new EventHandler(ActionLine\_Click);

comboBox\_key = new MyComboBox();

comboBox\_key.DropDownStyle = ComboBoxStyle.DropDownList;

comboBox\_key.FlatStyle = FlatStyle.Flat;

comboBox\_key.Items.Add("");

comboBox\_key.Items.Add("新建驱动");

// comboBox\_key.Items.Add("新建Appium驱动");

// comboBox\_key.Items.Add("新建IOS驱动");

comboBox\_key.Items.Add("打开网址");

comboBox\_key.Items.Add("设置参数");

comboBox\_key.Items.Add("执行方法");

comboBox\_key.Items.Add("单步方法");

comboBox\_key.Items.Add("Python代码");

comboBox\_key.Items.Add("{");

comboBox\_key.Items.Add("}");

comboBox\_key.Items.Add("暂停");

comboBox\_key.Items.Add("调用其他用例");

comboBox\_key.Items.Add("读取外部配置");

comboBox\_key.Items.Add("退出");

comboBox\_key.Items.Add("注释");

comboBox\_key.Text = key;

////////////////

comboBox\_key.SelectedIndexChanged += new EventHandler(comboBox\_key\_SelectedIndexChanged);

//control\_par1 = new TextBox();

////Console.WriteLine("判断动作名称结束" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

////Console.WriteLine("判断key开始" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

if (key == "执行方法")

{

this.control\_par1 = new TextBox();

this.setPar2Items();

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

comboBox\_key.BackColor = KeyColor.ExecuteFunctionActionBackColor;

comboBox\_key.ForeColor = KeyColor.ExecuteFunctionActionFontColor;

}

else if (key == "单步方法")

{

this.setPar1Items();

((MyComboBox)control\_par1).SelectedIndexChanged += new EventHandler(comboBox\_oneStep\_SelectedIndexChanged);

this.control\_par2 = new TextBox();

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = true;

comboBox\_key.BackColor = KeyColor.OneStepActionBackColor;

comboBox\_key.ForeColor = KeyColor.OneStepActionFontColor;

}

else

{

control\_par1 = new TextBox();

control\_par2 = new TextBox();

if (key == "新建驱动")

{

comboBox\_key.BackColor = KeyColor.CreateDriverActionBackColor;

comboBox\_key.ForeColor = KeyColor.CreateDriverActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = false;

}

else if (key == "打开网址")

{

comboBox\_key.BackColor = KeyColor.GoActionBackColor;

comboBox\_key.ForeColor = KeyColor.GoActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if (key == "设置参数")

{

comboBox\_key.BackColor = KeyColor.SetParameterActionBackColor;

comboBox\_key.ForeColor = KeyColor.SetParameterActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = true;

}

else if (key == "暂停")

{

comboBox\_key.BackColor = KeyColor.SleepActionBackColor;

comboBox\_key.ForeColor = KeyColor.SleepActionFontColor;

this.control\_par1.Enabled = true;

this.control\_par2.Enabled = false;

}

else if (key == "调用其他用例")

{

comboBox\_key.BackColor = KeyColor.ExeOtherCaseActionBackColor;

comboBox\_key.ForeColor = KeyColor.ExeOtherCaseActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if (key == "退出")

{

comboBox\_key.BackColor = KeyColor.QuitActionBackColor;

comboBox\_key.ForeColor = KeyColor.QuitActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if (key == "注释")

{

this.control\_par2 = new AnnotationTextBox();

comboBox\_key.BackColor = KeyColor.AnnotationActionBackColor;

comboBox\_key.ForeColor = KeyColor.AnnotationActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if (key == "Python代码")

{

this.control\_par2 = new PythonTextBox();

comboBox\_key.BackColor = KeyColor.PythonCodeActionBackColor;

comboBox\_key.ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else if (key == "{")

{

comboBox\_key.BackColor = KeyColor.PythonCodeActionBackColor;

comboBox\_key.ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if (key == "}")

{

comboBox\_key.BackColor = KeyColor.PythonCodeActionBackColor;

comboBox\_key.ForeColor = KeyColor.PythonCodeActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

else if (key == "读取外部配置")

{

comboBox\_key.BackColor = KeyColor.ReadExternalConfBackAction;

comboBox\_key.ForeColor = KeyColor.ReadExternalConfFontAction;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = true;

}

else

{

comboBox\_key.BackColor = KeyColor.NullActionBackColor;

comboBox\_key.ForeColor = KeyColor.NullActionFontColor;

this.control\_par1.Enabled = false;

this.control\_par2.Enabled = false;

}

}

////Console.WriteLine("判断key结束" + DateTime.Now.ToString("yyyyMMss HH:mm:ss.fff"));

control\_par1.Text = par1;

control\_par2.Text = par2;

checkBox\_isRun = new CheckBox();

checkBox\_isRun.Checked = isRun;

checkBox\_isRun.Left = control\_par2.Left + control\_par2.Width;

checkBox\_isRun.CheckedChanged += new EventHandler(checkBox\_isRun\_CheckedChanged);

setColorByIsRun(isRun);

setWidth(width);

this.Controls.Add(label\_index);

this.Controls.Add(comboBox\_key);

this.Controls.Add(control\_par1);

this.Controls.Add(control\_par2);

this.Controls.Add(checkBox\_isRun);

this.Height = comboBox\_key.Height + 2;

}

private void comboBox\_key\_MouseWheel(object sender, MouseEventArgs e)

{

//Console.WriteLine("ssss");

}

}

//节点类型的枚举 目录 或者 文件

public enum NodeType { DIR, FILE }

public class Form\_Search : Form

{

ETabPage eTabPage;

String searchKeyword;

String replaceWord;

TextBox textBox\_searchKeyword ;

TextBox textBox\_replaceWord;

Button button\_search ;

Button button\_searchReplace;

public Form\_Search(ETabPage eTabPage)

: base()

{

this.Text = "查找 替换";

this.StartPosition = FormStartPosition.CenterParent;

this.MaximizeBox = false;

this.MinimizeBox = false;

this.FormBorderStyle = FormBorderStyle.FixedDialog;

this.eTabPage = eTabPage;

Label label1 = new Label();

label1.Text = "关键词：";

Label label2 = new Label();

label2.Text = "替换词：";

textBox\_searchKeyword = new TextBox();

textBox\_replaceWord = new TextBox();

button\_search = new Button();

button\_searchReplace = new Button();

button\_search.Text = "查找";

button\_searchReplace.Text = "查找并替换";

label1.Width = label2.Width = 60;

label1.Left = label2.Left = 10;

label1.Top = 20;

label2.Top = label1.Top + label1.Height + 10;

textBox\_searchKeyword.Top = label1.Top-5;

textBox\_searchKeyword.Left = label1.Left + 60;

textBox\_replaceWord.Top = label2.Top-5;

textBox\_replaceWord.Left = textBox\_searchKeyword.Left;

textBox\_searchKeyword.Width = textBox\_replaceWord.Width = 200;

button\_search.Top = textBox\_replaceWord.Top + textBox\_replaceWord.Height + 16;

button\_searchReplace.Top = button\_search.Top;

button\_search.Left = label1.Left;

button\_searchReplace.Left = button\_search.Left + button\_search.Width + 32;

this.Height = button\_searchReplace.Top + button\_searchReplace.Height + 40;

this.Controls.Add(label1);

this.Controls.Add(label2);

this.Controls.Add(textBox\_searchKeyword );

this.Controls.Add(textBox\_replaceWord);

this.Controls.Add(button\_searchReplace);

this.Controls.Add(button\_search);

button\_search.Click+=new EventHandler(button\_search\_Click);

button\_searchReplace.Click+=new EventHandler(button\_searchReplace\_Click);

}

public void button\_search\_Click(object sender, EventArgs e)

{

eTabPage.actionLines.recoverySearchColor();

searchKeyword = this.textBox\_searchKeyword.Text.Trim();

if (null != eTabPage)

{

eTabPage.actionLines.search(searchKeyword);

}

}

public void button\_searchReplace\_Click(object sender, EventArgs e)

{

eTabPage.actionLines.recoverySearchColor();

searchKeyword = this.textBox\_searchKeyword.Text;

replaceWord = this.textBox\_replaceWord.Text;

if (null != eTabPage)

{

int count = eTabPage.actionLines.searchAndReplace(searchKeyword, replaceWord);

MessageBox.Show("一共有 "+ count + " 行中的文本被替换！");

}

}

}

public class OneStepMenuAndPrompt

{

public String menuText;

public String promptText;

public OneStepMenuAndPrompt(String menuText, String promptText)

{

this.menuText = menuText;

this.promptText = promptText;

}

}

public class Form\_CopySteps : Form

{

Form fatherForm;

int sourceTabStepSize;

//源的步骤范围

Label label\_sourceSteps;

public TextBox textBox\_sourceSteps;

//复制还是剪切

Label label\_copyOrCut;

public ComboBox comboBox\_copyOrCut;

//复制或剪切到哪个已打开的用例

Label label\_targetCase;

public ComboBox comboBox\_targetCase;

//复制或剪切到哪一步

Label label\_targetStep;

public ComboBox comboBox\_targetStep;

//确定按钮

public Button button\_OK;

private void comboBox\_targetCase\_SelectedIndexChanged(object sender, EventArgs e)

{

//////////Console.WriteLine(comboBox\_targetCase.SelectedIndex);

int targetIndex = comboBox\_targetCase.SelectedIndex;

ETabPage eTabPage = (ETabPage)((Form1)(this.fatherForm)).panelAndTabControl.tabControl.TabPages[targetIndex];

int size = eTabPage.actionLines.actionList.Count;

//////////Console.WriteLine("步骤数量=" + size);

this.comboBox\_targetStep.Items.Clear();

for (int i = 0; i < size; i++)

{

this.comboBox\_targetStep.Items.Add(i + 1);

}

comboBox\_targetStep.Items.Add("最后");

}

public Form\_CopySteps(Form fatherForm, List<String> targetCasePath)

: base()

{

this.fatherForm = fatherForm;

//得到源tag中一共有多少步骤

sourceTabStepSize = ((ETabPage)(((Form1)(this.fatherForm)).panelAndTabControl.tabControl.SelectedTab)).actionLines.actionList.Count;

label\_sourceSteps = new Label();

label\_sourceSteps.Text = "源步骤范围";

textBox\_sourceSteps = new TextBox();

label\_sourceSteps.Top = 0;

label\_sourceSteps.Left = 0;

textBox\_sourceSteps.Top = 0;

textBox\_sourceSteps.Left = 100;

if (0 == sourceTabStepSize)

{

textBox\_sourceSteps.Enabled = false;

}

label\_copyOrCut = new Label();

label\_copyOrCut.Text = "复制或剪切";

comboBox\_copyOrCut = new ComboBox();

comboBox\_copyOrCut.Items.AddRange(new String[] { "复制", "剪切" });

label\_copyOrCut.Top = 50;

label\_copyOrCut.Left = 0;

comboBox\_copyOrCut.Top = 50;

comboBox\_copyOrCut.Left = 100;

label\_targetCase = new Label();

label\_targetCase.Text = "目标用例";

comboBox\_targetCase = new ComboBox();

comboBox\_targetCase.SelectedIndexChanged += new EventHandler(comboBox\_targetCase\_SelectedIndexChanged);

label\_targetCase.Top = 100;

label\_targetCase.Left = 0;

comboBox\_targetCase.Top = 100;

comboBox\_targetCase.Left = 100;

int selectedIndex = ((Form1)(this.fatherForm)).panelAndTabControl.tabControl.SelectedIndex;

int i = 0;

foreach (String casePath in targetCasePath)

{

int l = casePath.LastIndexOf("\\");

String fileName = casePath.Substring(l + 1);

if (i == selectedIndex)

{

fileName += "(自己)";

}

comboBox\_targetCase.Items.Add(fileName);

i++;

}

label\_targetStep = new Label();

label\_targetStep.Text = "从哪一步插入";

comboBox\_targetStep = new ComboBox();

label\_targetStep.Top = 150;

label\_targetStep.Left = 0;

comboBox\_targetStep.Top = 150;

comboBox\_targetStep.Left = 100;

button\_OK = new Button();

button\_OK.Text = "确定";

button\_OK.Top = 200;

this.Controls.Add(this.label\_sourceSteps);

this.Controls.Add(this.textBox\_sourceSteps);

this.Controls.Add(this.label\_copyOrCut);

this.Controls.Add(this.comboBox\_copyOrCut);

this.Controls.Add(this.label\_targetCase);

this.Controls.Add(this.comboBox\_targetCase);

this.Controls.Add(this.label\_targetStep);

this.Controls.Add(this.comboBox\_targetStep);

this.Controls.Add(this.button\_OK);

button\_OK.Click += new EventHandler(button\_OK\_Click);

}

public void button\_OK\_Click(object sender, EventArgs e)

{

int sourceTabIndex = 0;

int sourceStepStart = 0;

int sourceStepEnd = 0;

int targetTabIndex = 0;

int targetStep = 0;

String copyOrCut = "";

sourceTabIndex = ((Form1)(this.fatherForm)).panelAndTabControl.tabControl.SelectedIndex;

String tmpStr = this.textBox\_sourceSteps.Text;

try

{

if (tmpStr.Contains("-"))

{

String[] tmps = tmpStr.Split(new String[] { "-" }, StringSplitOptions.None);

sourceStepStart = Int32.Parse(tmps[0]) - 1;

sourceStepEnd = Int32.Parse(tmps[1]) - 1;

}

else

{

sourceStepStart = Int32.Parse(tmpStr) - 1;

sourceStepEnd = sourceStepStart;

}

if (sourceStepStart >= 0 && sourceStepEnd <= this.sourceTabStepSize - 1 && sourceStepStart <= sourceStepEnd)

{

if ("复制" == comboBox\_copyOrCut.Text)

{

copyOrCut = "COPY";

}

else

{

copyOrCut = "CUT";

}

targetTabIndex = this.comboBox\_targetCase.SelectedIndex;

targetStep = this.comboBox\_targetStep.SelectedIndex;

((Form1)(this.fatherForm)).panelAndTabControl.copyOrCutSteps(sourceTabIndex, sourceStepStart, sourceStepEnd, targetTabIndex, targetStep, copyOrCut);

}

else

{

MessageBox.Show("请输入正确的范围", "错误");

}

}

catch (Exception e2)

{

//////////Console.WriteLine(e2.Message);

MessageBox.Show("请输入正确的格式，如“2-6”", "错误");

}

}

}

}

文件 TextProcessor.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Windows.Forms;

namespace CaseManager

{

class TextProcessor

{

public String getAbsPath()

{

String exePath = System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName;

int i = exePath.LastIndexOf("\\");

return exePath.Substring(0, i);

}

public int getBeginIndex()

{

String tmp = File.ReadAllText(this.getAbsPath() + "\\conf\\index.txt", Encoding.UTF8);

int i = Int32.Parse(tmp);

return i;

}

//写文本

public void writeContent(String filePath, String content, bool append)

{

StreamWriter sw = new StreamWriter(filePath, append, new UTF8Encoding(false));

sw.WriteLine(content);

sw.Close();

}

//通过环境文件 和 数据文件 得到数据字典 并加入几个特殊的键

private Dictionary<String, String> getTestData(String envFileName, String excelFileName)

{

String envName = File.ReadAllText(envFileName, Encoding.UTF8).Trim().Replace("\r\n","") ;

Dictionary<String, String> dic = new ExcelProcessor().getDicByEnvNameFromExcel(excelFileName, envName);

return dic;

}

public List<OneStepMenuAndPrompt> getOneStepMenuAndPromptList(String filePath)

{

//Console.WriteLine(DateTime.Now);

//Console.WriteLine("调用getOneStepMenuAndPromptList");

List<OneStepMenuAndPrompt> list = new List<OneStepMenuAndPrompt>();

String allText = File.ReadAllText(filePath, Encoding.UTF8);

String[] tmps = allText.Split(new String[] { "\r\n" } , StringSplitOptions.RemoveEmptyEntries);

foreach (String line in tmps)

{

String line2 = line.Trim();

String menuText="";

String prompText = "";

if( line2.Contains("=" ))

{

int i = line2.IndexOf("=");

menuText = line2.Substring(0, i);

prompText = line2.Substring(i + 1);

}

else

{

menuText = line2;

prompText = "";

}

OneStepMenuAndPrompt oneStepMenuAndPrompt = new OneStepMenuAndPrompt(menuText, prompText);

list.Add(oneStepMenuAndPrompt);

}

return list;

}

//从一个文本文件中得到字典 每行用“=”分割

private Dictionary<String, String> getDic(String filePath)

{

Dictionary<String, String> dic = new Dictionary<string, string>();

String allText = File.ReadAllText(filePath, Encoding.UTF8);

String[] lines = allText.Split(new String[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

foreach (String line in lines)

{

String[] tmps = line.Split(new String[] { "=" }, StringSplitOptions.RemoveEmptyEntries);

dic.Add(tmps[0], tmps[1]);

}

return dic;

}

//从方法配置文件中得到所有的方法名

public List<String> getFuns(String filePath)

{

List<String> list = new List<string>();

Dictionary<String, String> dic = getDic(filePath);

foreach (String str in dic.Keys)

{

list.Add(str);

}

return list;

}

//codeModulFilePath 代码模版路径

//importFilePath 代码中导入部分的文件路径

//excelFilePath 用例的电子表格文件路径

//functionFilePath 方法配置文件路径

//envFilePath 环境文件路径

//dataFilePath 数据文件路径

//得到整个临时python文件内容

public String getWholePyCodeFromScriptFile(String codeModulFilePath, String testPagePath, String excelFilePath, String envFilePath, String dataFilePath ,String importFilePath)

{

String importText = this.createImportCode(testPagePath);

String importText2 = "";

if (File.Exists(importFilePath))

{

importText2 = File.ReadAllText(importFilePath, Encoding.UTF8);

}

String pyCode = getPyCodes(excelFilePath, testPagePath, envFilePath, dataFilePath);

String modulText = File.ReadAllText(codeModulFilePath, Encoding.UTF8);

return "#encoding=utf-8\r\n\r\nfrom frame.MyTestCase import MyTestCase\r\nfrom frame.DriverInit import DriverInit\r\nfrom frame.PageOfPublic import PageOfPublic\r\nimport time\r\nimport json\r\n" + importText + "\r\n\r\n" + importText2 + "\r\n\r\n\r\n" + modulText.Replace("$CODES$", pyCode);

}

//自动生成 import 代码

private String createImportCode(String testPagePath)

{

List<String> list = new DirectoryManager().getAllFiles(testPagePath, "py");

List<String> list2 = new List<string>();

foreach (String str in list)

{

if (!str.Contains("\_\_init\_\_"))

{

String codeLine = path2ImportCode(str);

list2.Add(codeLine);

}

}

StringBuilder sb = new StringBuilder();

foreach (String str in list2)

{

//////Console.WriteLine(str);

sb.Append( str + "\r\n" );

}

sb.Append("\r\n");

return sb.ToString();

}

private String path2ImportCode(String path)

{

String[] tmps = path.Split(new String[] { "src\\" }, StringSplitOptions.None);

String tmp = tmps[1].Replace(".py", "").Replace("\\", ".");

String[] tmps2 = tmp.Split(new String[] { "." }, StringSplitOptions.None);

String pythonClassName = tmps2[tmps2.Length - 1];

String code = "from " + tmp + " import " + pythonClassName;

return code;

}

//自动生成方法字典

public Dictionary<String, String> createFunctionDic(String testPagePath)

{

List<String> list = new DirectoryManager().getAllFiles(testPagePath, "py");

List<String> list2 = new List<string>();

foreach (String str in list)

{

if (!str.Contains("\_\_init\_\_"))

{

list2.Add(str);

}

}

Dictionary<String, String> dicAll = new Dictionary<string, string>();

foreach (String filePath in list2)

{

Dictionary<String, String> dic = getFunctionFromPythonFile(filePath);

foreach (KeyValuePair<String, String> kv in dic)

{

if (!dicAll.Keys.Contains(kv.Key))

{

dicAll.Add(kv.Key, kv.Value);

}

else

{

dicAll.Add(kv.Key+"X" , kv.Value);

}

}

}

return dicAll;

}

//必须传入一个 python文件的路径

private Dictionary<String,String > getFunctionFromPythonFile(String pythonFile)

{

Dictionary<String, String> dic = new Dictionary<string, string>();

String pythonClassName = Path.GetFileName(pythonFile).Replace(".py", "");

String[] lines = File.ReadAllLines(pythonFile, Encoding.UTF8);

for (int i = 0; i < lines.Length; i++)

{

String line = lines[i].Trim() ;

if (line.StartsWith("####") && line.EndsWith("####"))

{

String bFunctionName = line.Replace("#","");

String nextLine = lines[i + 1];

String pythonFunction = nextLine.Replace(":", "").Replace("def", "").Replace("(self)", "").Trim();

String pythonCode = pythonClassName + "." + pythonFunction;

dic.Add(bFunctionName, pythonCode);

}

}

return dic;

}

//excelFile 用例的电子表格文件

//functionDic 方法字典

//dataDic 数据字典

//生成中间的test里面的执行部分的python代码

private String getPyCodes(String excelFile, String testPagePath, String envFilePath, String dataFilePath)

{

List<Action> list = new ExcelProcessor().readActionListFromExcelFile(excelFile);

Dictionary<String, String> functionDic = new TextProcessor().createFunctionDic(testPagePath);

Dictionary<String, String> dataDic = this.getTestData(envFilePath, dataFilePath);

//这里再加入几个特殊的键

String timeStr = DateTime.Now.ToString("yyyyMMddHHmmss");

dataDic.Add("TIME", timeStr);

dataDic.Add("RANDOM2", getRandom(2));

dataDic.Add("RANDOM3", getRandom(3));

dataDic.Add("RANDOM4", getRandom(4));

dataDic.Add("RANDOM5", getRandom(5));

dataDic.Add("RANDOM6", getRandom(6));

return getPyCodes(list, functionDic, dataDic,0);

}

public String getRandom(int i)

{

Random rd = new Random();

int r = rd.Next((int)System.Math.Pow(10, i));

String tmp = "00000000000000000000" + r;

int l = tmp.Length;

tmp = tmp.Substring(l - i);

return tmp;

}

private String getPyCodes(List<Action> list, Dictionary<String, String> functionDic, Dictionary<String, String> dataDic, int indent)

{

return getPyCodes(list, functionDic, dataDic, TextProcessor.indent , false);

}

//list 动作的列表

//functionDic 方法字典

//dataDic 数据字典

//remove3Action 对新建驱动 打开网址 退出 忽略

//生成中间的test里面的执行部分的python代码

private String getPyCodes(List<Action> list, Dictionary<String, String> functionDic, Dictionary<String, String> dataDic , int indent, bool remove3Action )

{

StringBuilder sb = new StringBuilder();

foreach (Action action in list)

{

String pythonCode = "";

if (remove3Action)

{

if (action.GetType().FullName.EndsWith("CreateDriverAction") || action.GetType().FullName.EndsWith("GoAction") || action.GetType().FullName.EndsWith("QuitAction"))

{

pythonCode = "";

}

else

{

pythonCode = getPyCode(action, functionDic, dataDic,indent);

}

}

else

{

pythonCode = getPyCode(action, functionDic, dataDic ,indent);

}

sb.Append(pythonCode + "\r\n");

}

return sb.ToString();

}

public static int indent = 0;

private String getIndentSpace(int indent)

{

String tmp = "";

for (int i = 0; i < indent; i++)

{

tmp += " ";

}

return tmp;

}

//action 动作

//functionDic 方法字典

//dataDic 数据字典

private String getPyCode(Action action, Dictionary<String, String> functionDic, Dictionary<String, String> dataDic , int indent)

{

String code = "";

int index = action.index;

bool isRun = action.isRun;

String note = this.getIndentSpace(TextProcessor.indent) + " #步骤" + index + "生成的代码\r\n";

//+this.getIndentSpace(TextProcessor.indent) + " print('步骤" + index + "')";

if (isRun)

{

//创建驱动

if (action.GetType().FullName.EndsWith("CreateDriverAction"))

{

String browserName = ((CreateDriverAction)action).browserName;

String driverName = ((CreateDriverAction)action).driverName;

code = this.getIndentSpace(TextProcessor.indent) + " if self.getPar('LASTSTEP'):";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " self.driver = DriverInit(self.logger).getWebDriver('" + browserName + "')";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " pP=PageOfPublic(self.pars,self.driver,self.logger)";

}

/\*

//创建Appium驱动

else if (action.GetType().FullName.EndsWith("CreateAppiumDriverAction"))

{

String deviceName = ((CreateAppiumDriverAction)action).deviceName;

String appPackageAndAppActivity = ((CreateAppiumDriverAction)action).appPackageAndAppActivity;

String[] tmp = appPackageAndAppActivity.Split( new String[] {","},StringSplitOptions.None );

String appPackage = tmp[0];

String appActivity = tmp[1];

code = " this.appiumDriver = new Init().getAppiumDriver(\"" + deviceName + "\", \"" + appPackage + "\", \"" + appActivity + "\");" + "\r\n this.appPackage=\"" + appPackage + "\";";

}

//创建IOS驱动

else if (action.GetType().FullName.EndsWith("CreateIOSDriverAction"))

{

String macIP = ((CreateIOSDriverAction)action).macIP;

String appPath = ((CreateIOSDriverAction)action).appPath;

code = " this.appiumDriver = new Init().getIOSDriver(\""+ macIP +"\",\""+ appPath +"\");";

}

\* \*/

// 打开网址

else if (action.GetType().FullName.EndsWith("GoAction"))

{

String driverName = ((GoAction)action).driverName;

String url = ((GoAction)action).url;

code = this.getIndentSpace(TextProcessor.indent) + " if self.getPar('LASTSTEP'):";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " self.openURL('" + url + "')";

}

//设置参数

else if (action.GetType().FullName.EndsWith("SetParameterAction"))

{

String parName = ((SetParameterAction)action).parName;

String parValue = ((SetParameterAction)action).parValue;

code = this.getIndentSpace( TextProcessor.indent ) + " self.setPar(\"" + parName + "\", \"" + parValue + "\")";

}

//执行方法

else if (action.GetType().FullName.EndsWith("ExecuteFunctionAction"))

{

String driverName = ((ExecuteFunctionAction)action).driverName;

String functionName = ((ExecuteFunctionAction)action).functionName;

if (functionDic.Keys.Contains(functionName))

{

String classAndFunction = functionDic[functionName];

code = this.getIndentSpace( TextProcessor.indent ) + " " + classAndFunction.Replace(".", "(self.pars, self.driver, self.logger).") + "()";

}

else

{

code = this.getIndentSpace(TextProcessor.indent) + " print('没有找到配置的方法')";

}

}

//读取外部配置

else if (action.GetType().FullName.EndsWith("ReadExternalConfAction"))

{

String excelPath = ((ReadExternalConfAction)action).excelPath;

if( ":"!= excelPath.Substring(1,1 ))

{

String absPath = new TextProcessor().getAbsPath();

excelPath = absPath + "\\files\\" + excelPath;

}

ExcelProcessor excelProcessor =new ExcelProcessor();

List<List<String>> lists = excelProcessor.readfirststr(excelPath);

Dictionary<String,String> dic = excelProcessor.strList2Dic(lists);

//code = this.getIndentSpace(TextProcessor.indent) + " print('步骤" + index + "')";

foreach( KeyValuePair<String,String> kv in dic )

{

String k = kv.Key;

String v = kv.Value;

code += this.getIndentSpace( TextProcessor.indent ) + " self.setPar(\"" + k + "\", \"" + v + "\")";

code += "\r\n";

}

}

//单步方法

else if (action.GetType().FullName.EndsWith("OneStepAction"))

{

String oneStepAction = ((OneStepAction)action).oneStepName;

String oneStepPars = ((OneStepAction)action).oneStepPars;

code =this.getIndentSpace( TextProcessor.indent ) + " pP.oneStep('" + oneStepAction + "','''" + oneStepPars + "''')";

}

//暂停

else if (action.GetType().FullName.EndsWith("SleepAction"))

{

int timeLong = ((SleepAction)action).timeLong;

code = this.getIndentSpace( TextProcessor.indent ) + " time.sleep(" + timeLong + ")";

}

//调用其他用例

else if (action.GetType().FullName.EndsWith("ExeOtherCaseAction"))

{

String casePath = ((ExeOtherCaseAction)action).casePath;

String caseFullPath = new TextProcessor().getAbsPath() + "\\" + casePath;

//////Console.WriteLine(casePath);

List<Action> list = new ExcelProcessor().readActionListFromExcelFile(caseFullPath);

code = this.getIndentSpace(TextProcessor.indent) + " ################################################################";

code += "\r\n\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " #执行用例[" + casePath + "]";

code += "\r\n\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " ################################################################";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " self.logger.appendContent('\\r\\n################################################################\\r\\n执行用例[" + casePath + "]\\r\\n################################################################')";

code += "\r\n";

code += this.getPyCodes(list, functionDic, dataDic, TextProcessor.indent);

}

//退出

else if (action.GetType().FullName.EndsWith("QuitAction"))

{

String driverName = ((QuitAction)action).driverName;

code = this.getIndentSpace(TextProcessor.indent) + " if None!=self.driver:";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " self.driver.quit()";

code += "\r\n";

code += this.getIndentSpace(TextProcessor.indent) + " self.logger.appendContent(\"退出\")";

}

//注释

else if (action.GetType().FullName.EndsWith("QuitAction"))

{

String annotationText = ((AnnotationAction)action).annotationText;

code = this.getIndentSpace(TextProcessor.indent) + " #" + annotationText;

}

//Python代码

else if (action.GetType().FullName.EndsWith("PythonCodeAction"))

{

String pcode = ((PythonCodeAction)action).pcode;

code = this.getIndentSpace( TextProcessor.indent ) + " " + pcode;

}

//{

else if (action.GetType().FullName.EndsWith("AddIndentAction"))

{

TextProcessor.indent++;

code =this.getIndentSpace( TextProcessor.indent )+ " #开始缩进";

}

//{

else if (action.GetType().FullName.EndsWith("RecoverIndentAction"))

{

code += this.getIndentSpace(TextProcessor.indent) + " #恢复缩进";

TextProcessor.indent--;

}

return note + "\r\n" + replaceVars(code, dataDic) + "\r\n";

}

else

{

return "";

}

/\*

String fun = executeObject.fun;

String code = "";

if ("设置参数" == fun)

{

String par1 = executeObject.par1;

String par2 = executeObject.par2;

code = " pars.put(\"" + par1 + "\", \"" + par2 + "\");";

}

else if ("打开网址" == fun || "转到网址" == fun)

{

String par1 = executeObject.par1;

code = " openURL(\"" + par1 + "\");";

}

else if ("暂停" == fun)

{

String par1 = executeObject.par1;

code = " sleep("+par1+");";

}

else

{

if (functionDic.Keys.Contains(fun))

{

String classAndFunction = functionDic[fun];

code = " " + classAndFunction;

}

else

{

code = "";

}

}

return replaceVars(code, dataDic);

\*/

}

//替换多个参数

private String replaceVars(String str, Dictionary<String, String> dataDic)

{

String tmp = str;

foreach (String key in dataDic.Keys)

{

String a = "$" + key + "$";

String b = dataDic[key];

tmp = tmp.Replace(a, b);

}

return tmp;

}

}

class BusinessFunction

{

String bFunctionName;

String pythonClassName;

String pythonFunctionName;

public BusinessFunction(String bFunctionName,String pythonClassName,String pythonFunctionName)

{

this.bFunctionName = bFunctionName;

this.pythonClassName = pythonClassName;

this.pythonFunctionName = pythonFunctionName;

}

}

/\*

class ExecuteObject

{

public String fun;

public String par1;

public String par2;

public ExecuteObject(String fun, String par1, String par2)

{

this.fun = fun;

this.par1 = par1;

this.par2 = par2;

}

public ExecuteObject(String fun, String par1)

{

this.fun = fun;

this.par1 = par1;

}

public ExecuteObject(String fun)

{

this.fun = fun;

}

}

\* \*/

}

文件 PageOfPublic.py

#encoding=utf-8

from frame.EleDescription import EleDescription

from frame.tools.UpLoadWindow import UpLoadWindow

from frame.FrameSwitcher import FrameSwitcher

from frame.WindowSwitcher import WindowSwitcher

from frame.tools.DingSender import DingSender

from frame.tools.MySqlConnect import MySqlConnect

from frame.tools.KeyboardAndMouse import KeyboardAndMouse

from frame.tools.IdentifyingCodeDistinguish import IdentifyingCodeDistinguish

from frame.tools.ExcelOperator import ExcelOperator

import time

import datetime

import os

import autoit

import random

class PageOfPublic(object):

pars=None

driver=None

logger=None

def \_\_init\_\_(self, pars,driver,logger):

self.pars=pars

self.driver= driver

self.logger = logger

def replacePars(self,text):

tmp = text

keys = self.pars.map.keys()

for key in keys:

if key!="LASTSTEP":

tmp=tmp.replace("$"+ key +"$", str(self.getPar(key)) )

return tmp

def lastStepIsPass(self):

return self.pars.get("LASTSTEP")

def executeScript(self,jsStr):

self.driver.execute\_script(jsStr)

def setFailed(self,e) :

print('异常!!!!!!!! '+str(e))

self.pars.put("LASTSTEP", False);

self.logger.snap()

#单步方法

def oneStep(self,actionName,actionPars):

if self.lastStepIsPass():

try:

actionPars = self.replacePars(actionPars)

pars = actionPars.split("||")

if "点击" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

self.click(idOrXpath, eleName)

elif "选择单选按钮或复选框"== actionName:

idOrXpath = pars[0]

eleName = pars[1]

checked = pars[2]

self.selectRadioOrCheckbox(idOrXpath,eleName,checked=="是")

elif "悬停"==actionName:

idOrXpath = pars[0]

eleName = pars[1]

self.moveTo(idOrXpath, eleName)

elif "双击" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

self.doubleClick(idOrXpath,eleName)

elif "右键" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

self.rightClick(idOrXpath,eleName)

elif "拖拽" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

idOrXpath2 = pars[2]

eleName2 = pars[3]

self.drag(idOrXpath,eleName,idOrXpath2,eleName2)

elif "输入" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

text = pars[2]

self.input(idOrXpath, eleName, text)

elif "选择by文本" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

text = pars[2]

self.selectByText(idOrXpath, eleName, text)

elif "选择by值" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

value = pars[2]

self.selectByValue (idOrXpath, eleName, value)

elif "选择by序号" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

index = pars[2]

self.selectByIndex(idOrXpath, eleName, index)

elif "验证是否存在" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

exist = pars[2]

timeout= int(pars[3])

if exist=="是":

self.exist( idOrXpath,eleName,timeout )

else:

self.unExist(idOrXpath,eleName,timeout)

elif "得到是否存在"== actionName:

idOrXpath = pars[0]

eleName = pars[1]

parName = pars[2]

isExist = self.isExist(idOrXpath,eleName)

self.setPar(parName,isExist)

elif "得到文本" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

parName = pars[2]

tmp = self.getText(idOrXpath,eleName)

self.setPar( parName,tmp)

elif "验证文本" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

expectText = pars[2]

tmp = self.getText(idOrXpath, eleName)

if tmp!=expectText:

raise Exception("控件[" + eleName + "]的实际文本是[" + tmp+"]，与期望文本["+ expectText + "]不符合")

elif "得到值" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

parName = pars[2]

tmp = self.getValue(idOrXpath, eleName)

self.setPar(parName, tmp)

elif "验证值" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

expectValue = pars[2]

tmp = self.getValue(idOrXpath, eleName)

if tmp != expectValue:

raise Exception("控件[" + eleName + "]的实际值是[" + tmp + "]，与期望值[" + expectValue + "]不符合")

elif "验证是否选中" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

expectStat = pars[2]

checked = self.isChecked(idOrXpath,eleName)

if "是"==expectStat:

if not checked:

raise Exception("控件[" + eleName + "]的实际选中状态是[" + str(checked) + "]，与期望值[" + expectStat + "]不符合")

else:

if checked:

raise Exception("控件[" + eleName + "]的实际选中状态是[" + str(checked) + "]，与期望值[" + expectStat + "]不符合")

elif "得到选中状态"==actionName:

idOrXpath = pars[0]

eleName = pars[1]

parName = pars[2]

checked = self.isChecked(idOrXpath, eleName)

self.setPar(parName,checked)

elif "得到属性" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

attrName = pars[2]

parName = pars[3]

tmp = self.getAttr(idOrXpath, eleName ,attrName)

self.setPar(parName, tmp)

elif "验证属性" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

attrName = pars[2]

expectAttrValue = pars[3]

tmp = self.getAttr(idOrXpath, eleName, attrName)

if tmp != expectAttrValue:

raise Exception("控件[" + eleName + "]的["+ attrName +"]属性实际值是[" + tmp + "]，与期望值[" + expectAttrValue + "]不符合")

elif "移除属性" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

attrName = pars[2]

self.removeAttr(idOrXpath,eleName,attrName)

elif "修改属性" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

attrName = pars[2]

newAttrValue = pars[3]

self.editAttr(idOrXpath,eleName,attrName,newAttrValue)

elif "强制可输入" == actionName:

idOrXpath = pars[0]

eleName = pars[1]

self.removeAttr(idOrXpath, eleName, "readonly")

elif "进入iframe" == actionName:

idOrXpath = pars[0]

iframeName =pars[1]

self.switchFrame(idOrXpath,iframeName)

elif "退出iframe" == actionName:

self.exitFrame()

elif "切换窗口" == actionName:

index = int(pars[0])

self.switchWindow(index)

elif "只保留第一窗口"== actionName:

self.closeAllwindowsExceptFirst()

elif "回退窗口" == actionName:

self.back()

elif "前进窗口" == actionName:

self.forword()

elif "刷新窗口" == actionName:

self.refresh()

elif "执行js" == actionName:

if len(pars)==1:

jsStr = pars[0]

self.exeJS(jsStr)

else:

jsStr = pars[0]

idOrXpath = pars[1]

eleName = pars[2]

self.exeJS(jsStr,idOrXpath,eleName)

elif "发送钉钉消息"==actionName:

dingUser= pars[0]

dingPassword = pars[1]

userList = pars[2].split(",")

content= pars[3]

content=self.replacePars(content)

DingSender(dingUser,dingPassword,self.logger).sendMsg(userList,content)

elif "得到相对日期"==actionName:

if len(pars)==2:

day = int(pars[0])

parName = pars[1]

elif len(pars)==1:

day= 0

parName = pars[0]

t = datetime.date.today() + datetime.timedelta(days=day)

self.setPar( parName,str(t) )

elif "得到相对时间" == actionName:

unit=pars[0]

num = int(pars[1])

parName = pars[2]

now = datetime.datetime.now()

if "秒"==unit:

t= now+datetime.timedelta(seconds=num)

elif "分"==unit:

t= now+datetime.timedelta(minutes=num)

elif "小时"==unit:

t = now+datetime.timedelta(hours=num)

elif "天"==unit or "日"==unit:

t = now+datetime.timedelta(days=num)

elif "星期"==unit or "周"==unit:

t = now+datetime.timedelta(weeks=num)

timeStr = t.strftime("%Y-%m-%d %H:%M:%S")

self.setPar(parName, timeStr)

elif "得到随机数"==actionName:

num = int(pars[0])

parName=pars[1]

randomStr= self.getRandom(num)

self.setPar(parName,randomStr)

elif "调试输出"==actionName:

text = pars[0]

print("调试输出\r\n[[[[" + text + "\r\n]]]]")

elif "点击js弹出框的确定"==actionName:

self.alertAccept()

elif "点击js弹出框的取消"==actionName:

self.alertCancel()

elif "得到子字符串"==actionName:

if len(pars) == 4:

textSource = pars[0]

indexStart = int(pars[1])

indexEnd = int(pars[2])

parName = pars[3]

subStr = textSource[indexStart:indexEnd]

self.setPar(parName,subStr)

elif len(pars) == 3:

textSource = pars[0]

indexStart = int(pars[1])

parName = pars[2]

subStr = textSource[indexStart:]

self.setPar(parName, subStr)

elif "替换字符串"==actionName:

textSource = pars[0]

oldText = pars[1]

newText = pars[2]

parName = pars[3]

tmp = textSource.replace(oldText,newText)

self.setPar(parName, tmp)

elif "去除字符串"==actionName:

textSource = pars[0]

oldText = pars[1]

parName = pars[2]

tmp = textSource.replace(oldText,"")

self.setPar(parName, tmp)

elif "分割后取第几个"==actionName:

textSource = pars[0]

splitStr = pars[1]

index = int(pars[2])

parName = pars[3]

tmps = textSource.split(splitStr)

tmp= tmps[index]

self.setPar(parName, tmp)

elif "mysql查询"==actionName:

ip= pars[0]

port = int( pars[1] )

dbName = pars[2]

username= pars[3]

password = pars[4]

sql = pars[5]

parName = pars[6]

text = MySqlConnect(ip,port,dbName,username,password,self.logger).query(sql)

self.setPar(parName, text)

elif "mysql执行"==actionName:

ip = pars[0]

port = int(pars[1])

dbName = pars[2]

username = pars[3]

password = pars[4]

sql = pars[5]

MySqlConnect(ip, port, dbName, username, password, self.logger).execute(sql)

elif "键盘单键输入"==actionName:

key=pars[0]

KeyboardAndMouse().keyboardClick(key)

elif "鼠标点击坐标"==actionName:

x=int(pars[0])

y = int(pars[1])

KeyboardAndMouse().mouseClickLeft(x,y)

elif "鼠标滚轮"==actionName:

value=int(pars[0])\*(-1)

KeyboardAndMouse().mouseScroll(value)

elif "鼠标拖拽" == actionName:

locates = pars[0]

KeyboardAndMouse().mouseDraw(locates)

elif "屏幕找图并点击"==actionName:

picPath=pars[0]

KeyboardAndMouse().mouseClickByPic(picPath)

elif "屏幕找图得到坐标"==actionName:

picPath = pars[0]

parName0 = pars[1]

parName1 = pars[2]

parName2 = pars[3]

parName3 = pars[4]

tmp = KeyboardAndMouse().getPicLocalOnScreen(picPath)

self.setPar(parName0,tmp[0])

self.setPar(parName1, tmp[1])

self.setPar(parName2, tmp[2])

self.setPar(parName3, tmp[3])

print("左上角x=" + str(tmp[0]))

print("左上角y=" + str(tmp[1]))

print("右上角x=" + str(tmp[2]))

print("右上角y=" + str(tmp[3]))

elif "屏幕区域截图并保存"==actionName:

x0 = int(pars[0])

y0 = int(pars[1])

x1 = int(pars[2])

y1 = int(pars[3])

picPath = pars[4]

KeyboardAndMouse().saveScreenshotByLocation(x0,y0,x1,y1,picPath)

elif "识别图片中的字母和数字"==actionName:

picPath = pars[0]

parName = pars[1]

code = IdentifyingCodeDistinguish().distinguish(picPath)

self.setPar(parName,code)

elif "chrome上传单个文件"==actionName:

filePath = pars[0]

UpLoadWindow().upload(filePath)

elif "最小化浏览器"==actionName:

# self.driver.minimize\_window()

autoit.win\_minimize\_all()

elif "最大化浏览器"==actionName:

self.driver.maximize\_window()

elif "设置浏览器位置和大小"==actionName:

x = int(pars[0])

y = int(pars[1])

w = int(pars[2])

h = int(pars[3])

self.driver.set\_window\_position(x,y)

self.driver.set\_window\_size(w,h)

elif "执行命令"==actionName:

command = pars[0]

parName = pars[1]

outText = os.popen(command).read()

self.setPar(parName,outText)

elif "读取多行数据到序列"==actionName:

excelPath=pars[0]

parName = pars[1]

self.getListFromCaseDateFile(excelPath,parName)

elif "读取电子表格某单元格文本"==actionName:

excelPath = pars[0]

sheetName= pars[1]

row = int(pars[2]) # 电子表格左侧的数字

col = int(pars[3]) #电子表格上面的字母对应的数字， A 对应1 B 对应2 ..

parName = pars[4]

self.getTextOfACell(excelPath,sheetName,row,col,parName)

elif ""==actionName:

pass

else:

raise Exception("单步方法[" + actionName + "]未定义" )

except Exception as e:

self.setFailed(e)

def find(self , idOrXpath,eleName ,timeout=10):

if "/" in idOrXpath:

element = EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).find(timeout)

else:

element = EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).find(timeout)

return element

#休眠

def sleep(self,second):

time.sleep(second)

#获取参数

def getPar(self,parName):

return self.pars.get(parName)

#设置参数

def setPar(self,parName,parValue):

self.pars.set(parName,parValue)

#增加字典

def addDic(self,dic):

self.pars.addDic()

#验证控件存在

def exist(self, idOrXpath,eleName ,timeout ):

if "/" in idOrXpath:

element = EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).find(timeout)

else:

element = EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).find(timeout)

def unExist(self, idOrXpath,eleName ,timeout):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).waitDisappear(timeout)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).waitDisappear(timeout)

def isExist(self, idOrXpath,eleName):

if "/" in idOrXpath:

return EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).isExist()

else:

return EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).isExist()

#点击

def click(self , idOrXpath,eleName):

if "/" in idOrXpath:

EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).click()

else:

EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).click()

#单选按钮和复选框的选中和不选中

def selectRadioOrCheckbox(self,idOrXpath,eleName,selected):

ele=None

if "/" in idOrXpath:

ele = EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).find()

else:

ele = EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).find()

if selected != self.driver.execute\_script("return arguments[0].checked", ele):

ele.click()

#悬停

def moveTo(self , idOrXpath,eleName):

if "/" in idOrXpath:

EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).moveTo()

else:

EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).moveTo()

#双击

def doubleClick(self ,idOrXpath,eleName):

if "/" in idOrXpath:

EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).doubleClick()

else:

EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).doubleClick()

# 右键

def rightClick(self, idOrXpath, eleName):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).rightClick()

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).rightClick()

#拖拽

def drag(self, idOrXpath,eleName, idOrXpath2,eleName2):

ele2 = self.find(idOrXpath2,eleName2)

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).drag(ele2,eleName2)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).drag(ele2,eleName2)

#输入

def input(self,idOrXpath,eleName ,text):

if "/" in idOrXpath:

EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).input(text)

else:

EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).input(text)

#得到文本

def getText(self, idOrXpath,eleName ):

if "/" in idOrXpath:

return EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger).text();

else:

return EleDescription(self.driver,"id",idOrXpath,eleName,self.logger).text()

#通过index选择

def selectByIndex(self , idOrXpath,eleName , index):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).selectByIndex(index)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).selectByIndex(index)

# 通过value选择

def selectByValue(self , idOrXpath,eleName ,value):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).selectByValue(value)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).selectByValue(value)

# 通过文本选择

def selectByText(self, idOrXpath,eleName , text):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).selectByText(text)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).selectByText(text)

#chrome上传文件

def upload(self,filePath):

filePath = filePath.replace("/", "\\")

UpLoadWindow().upload(filePath)

def alertAccept(self):

self.driver.switch\_to\_alert().accept()

self.logger.appendContent("点击了弹出框的确定按钮")

self.driver.switch\_to.default\_content()

def alertCancel(self):

self.driver.switch\_to\_alert().dismiss()

self.logger.appendContent("点击了弹出框的取消按钮")

self.driver.switch\_to.default\_content()

def getAttr(self, idOrXpath,eleName ,attrName ):

if "/" in idOrXpath:

return EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).getAttr(attrName)

else:

return EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).getAttr(attrName)

def removeAttr(self, idOrXpath,eleName ,attrName ):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).removeAttr(attrName)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).removeAttr(attrName)

def editAttr(self, idOrXpath,eleName , attrName, attrValue):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).editAttr(attrName,attrValue)

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).editAttr(attrName, attrValue)

def clearText(self,idOrXpath,eleName):

if "/" in idOrXpath:

EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).clearText()

else:

EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).clearText()

def getValue(self,idOrXpath,eleName):

if "/" in idOrXpath:

return EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).getValue()

else:

return EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).getValue()

def getSelecterTexts(self ,idOrXpath,eleName):

if "/" in idOrXpath:

return EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).getSlecterTexts()

else:

return EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).getSlecterTexts()

def isChecked(self,idOrXpath,eleName):

if "/" in idOrXpath:

return EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).isChecked()

else:

return EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).isChecked()

# num 位数

def getRandom(self,num):

b = 10\*\*num;

r = random.randint(0,b);

s= str(r).zfill(num)

return s

def switchFrame(self,idOrXpath,eleName):

if "/" in idOrXpath:

iframe = EleDescription(self.driver,"xpath",idOrXpath,eleName,self.logger)

else:

iframe = EleDescription(self.driver, "id", idOrXpath, eleName, self.logger)

FrameSwitcher(self.driver,iframe,self.logger).switchFrame()

def exitFrame(self):

self.driver.switch\_to.default\_content()

self.logger.appendContent("退出frame")

def switchWindow(self,index):

WindowSwitcher(self.driver,self.logger).switchWindow(index)

def closeAllwindowsExceptFirst(self):

WindowSwitcher(self.driver, self.logger).closeAllwindowsExceptFirst()

def back(self):

self.driver.back()

def forword(self):

self.driver.forward()

def refresh(self):

self.driver.refresh()

def exeJS(self, jsStr , idOrXpath=None,eleName=None):

if idOrXpath != None and eleName!=None:

if "/" in idOrXpath:

ele = EleDescription(self.driver, "xpath", idOrXpath, eleName, self.logger).find()

else:

ele = EleDescription(self.driver, "id", idOrXpath, eleName, self.logger).find()

self.driver.execute\_script( jsStr , ele)

else:

self.driver.execute\_script(jsStr)

def getTextOfACell(self, filePath, sheetName, row, col,parName):

filePath = self.getFileFullPath(filePath)

text = ExcelOperator().getTextOfACell(filePath,sheetName,row,col)

self.setPar(parName,text)

def getListFromCaseDateFile(self,filePath,parName):

filePath= self.getFileFullPath(filePath)

maps = ExcelOperator().getDicsFromXls(filePath)

self.setPar( parName,maps )

def getFileFullPath(self,filePath):

pwd = os.getcwd()

tmp = pwd.split("src")[0]

if (filePath[1:2] != ':'):

# 是相对路径 要到files下面去找

filesDir = os.path.abspath(tmp + os.path.sep + "..") + "\\files"

picPath = filePath.replace("/", "\\")

if (not filePath.startswith("\\")):

filePath = "\\" + filePath

filePath = filesDir + filePath

return filePath

if \_\_name\_\_ == "\_\_main\_\_":

pass

文件 EleDescription.py

#coding=utf-8

import datetime

import time

from selenium.webdriver.support.ui import Select

from selenium.webdriver.common.action\_chains import ActionChains

class EleDescription(object):

driver = None

findType = None

findValue = None

eleName = None

logger = None

def \_\_init\_\_(self, driver, findType, findValue, eleName, logger):

self.driver = driver

self.findType = findType

self.findValue = findValue

self.eleName = eleName

self.logger = logger

def getIndex(self):

ele = self.find(10)

if None!= ele:

brothers = ele.find\_elements\_by\_xpath("./../\*")

index=0

for brother in brothers:

if brother==ele:

return index

index = index + 1

return -1

return -1

#等待消失

def waitDisappear(self, timeout=10):

startTime = datetime.datetime.now()

if self.findType.lower() == "id":

while len(self.driver.find\_elements\_by\_id(self.findValue)) > 0 and (

datetime.datetime.now() - startTime).seconds < timeout:

time.sleep(1)

elelist = self.driver.find\_elements\_by\_id(self.findValue)

if len(elelist) == 0:

self.logger.appendContent("根据id[" + self.findValue + "]等待到控件[" + self.eleName + "]消失");

else:

self.logger.appendContent("根据id[" + self.findValue + "]没有等待到控件[" + self.eleName + "]消失");

raise Exception("根据id[" + self.findValue + "]没有等待到控件[" + self.eleName + "]消失")

elif self.findType.lower() == "xpath":

while len(self.driver.find\_elements\_by\_xpath(self.findValue)) > 0 and (

datetime.datetime.now() - startTime).seconds < timeout:

time.sleep(1)

elelist = self.driver.find\_elements\_by\_xpath(self.findValue)

if len(elelist) == 0:

self.logger.appendContent("根据xpath[" + self.findValue + "]等待到控件[" + self.eleName + "]消失");

else:

self.logger.appendContent("根据xpath[" + self.findValue + "]没有等待到控件[" + self.eleName + "]消失");

raise Exception("根据xpath[" + self.findValue + "]没有等待到控件[" + self.eleName + "]消失")

def isExist(self):

if self.findType.lower() == "id":

l = len(self.driver.find\_elements\_by\_id(self.findValue))

return l>0

else:

l = len( self.driver.find\_elements\_by\_xpath(self.findValue ))

return l>0

def find(self, timeout=10):

startTime = datetime.datetime.now()

if self.findType.lower() == "id":

while len(self.driver.find\_elements\_by\_id(self.findValue)) == 0 and (datetime.datetime.now() - startTime).seconds < timeout :

time.sleep(1)

elelist = self.driver.find\_elements\_by\_id(self.findValue)

if len(elelist) > 0:

self.logger.appendContent("根据id[" + self.findValue + "]找到控件[" + self.eleName + "]");

self.driver.execute\_script("arguments[0].style.border='2px solid red'", elelist[0])

return elelist[0]

else:

self.logger.appendContent("根据id[" + self.findValue + "]没有找到控件[" + self.eleName + "]");

raise Exception("根据id[" + self.findValue + "]没有找到控件[" + self.eleName + "]")

elif self.findType.lower() == "xpath":

while len( self.driver.find\_elements\_by\_xpath(self.findValue ) )==0 and (datetime.datetime.now() - startTime).seconds < timeout:

time.sleep(1)

elelist = self.driver.find\_elements\_by\_xpath(self.findValue)

if len( elelist)>0:

self.logger.appendContent("根据xpath[" + self.findValue + "]找到控件[" + self.eleName + "]");

self.driver.execute\_script("arguments[0].style.border='2px solid red'", elelist[0])

return elelist[0]

else:

self.logger.appendContent("根据xpath[" + self.findValue + "]没有找到控件[" + self.eleName + "]");

raise Exception("根据xpath[" + self.findValue + "]没有找到控件[" + self.eleName + "]")

def click(self):

ele = self.find(10)

ele.click()

self.logger.appendContent( "点击控件[" + self.eleName + "]")

def moveTo(self):

ele = self.find(10)

ActionChains(self.driver).move\_to\_element(ele).perform()

self.logger.appendContent("鼠标移动到控件[" + self.eleName + "]")

def doubleClick(self):

ele = self.find(10)

ActionChains(self.driver).double\_click(ele).perform()

self.logger.appendContent("鼠标双击控件[" + self.eleName + "]")

def rightClick(self):

ele = self.find(10)

ActionChains(self.driver).context\_click(ele).perform()

self.logger.appendContent("鼠标右键单击控件[" + self.eleName + "]")

def drag(self, targetEle ,targetEleName ):

ele = self.find(10)

ActionChains(self.driver).drag\_and\_drop(ele,targetEle).perform()

self.logger.appendContent("鼠标拖拽控件[" + self.eleName + "]到控件["+ targetEleName + "]" )

def text(self):

ele = self.find(10)

text = ele.text

self.logger.appendContent("得到控件[" + self.eleName + "]的文本为["+ text +"]")

return text

def input(self,text):

ele = self.find(10)

ele.clear()

ele.send\_keys(text)

self.logger.appendContent( "控件[" + self.eleName + "]中输入文本["+text+"]")

def selectByIndex(self,index):

ele = self.find(10)

Select(ele).select\_by\_index(index)

self.logger.appendContent("根据index[" + index + "]选择控件["+ self.eleName +"]")

def selectByValue(self,value):

ele = self.find(10)

Select(ele).select\_by\_value(value)

self.logger.appendContent("根据value[" + value + "]选择控件[" + self.eleName + "]")

def selectByText(self, text):

ele = self.find(10)

Select(ele).select\_by\_visible\_text(text)

self.logger.appendContent("根据text[" + text + "]选择控件[" + self.eleName + "]")

def removeReadOnly(self):

ele = self.find(10)

self.driver.execute\_script("arguments[0].readOnly=false", ele)

self.logger.appendContent("将控件[" + self.eleName + "]的只读属性去掉")

#得到属性

def getAttr(self,attrName):

ele = self.find(10)

attrValue = ele.get\_attribute(attrName)

self.logger.appendContent("得到控件[" + self.eleName + "]的属性[" + attrName + "]的值为[" + attrValue + "]" )

return attrValue

#移除属性

def removeAttr(self,attrName):

ele = self.find(10)

self.driver.execute\_script("arguments[0].removeAttribute('"+ attrName +"')",ele)

#编辑属性

def editAttr(self,attrName,attrValue):

ele = self.find(10)

self.driver.execute\_script("arguments[0]."+ attrName +"='"+ attrValue +"';" ,ele)

self.logger.appendContent("将控件[" + self.eleName + "]的属性[" + attrName + "]的值修改为[" + attrValue + "]" )

#清除文本

def clearText(self):

ele = self.find(10)

ele.clear()

self.logger.appendContent("清除控件["+ self.eleName+"]的文本")

#得到值

def getValue(self):

ele=self.find(10)

value = self.driver.execute\_script("return arguments[0].value", ele)

self.logger.appendContent("得到控件[" + self.eleName + "]的值为[" + value + "]")

return value

#得到select 的 option的值的列表

def getSlecterTexts(self):

ele = self.find(10)

options = ele.find\_elements\_by\_xpath("./option");

texts = []

for option in options:

texts.append(option.text)

self.logger.appendContent("得到控件[" + self.eleName + "]的所有下拉选项为数组[ " + str(texts) + " ]")

return texts

#单选或者复选是否选中的状态

def isChecked(self):

ele = self.find(10)

return self.driver.execute\_script("return arguments[0].checked", ele)