using System;

using System.Linq;

using System.Web;

using System.Configuration;

using System.Collections.Generic;

using System.Web.UI;

using Microsoft.Reporting.WebForms;

using System.Web.UI.WebControls;

using System.IO;

using System.ComponentModel;

using System.Drawing.Printing;

public partial class \_Default : System.Web.UI.Page , ICallbackEventHandler

{

private string \_reportName;

private string \_sessionPDFFileName;

protected string iFrameURL;

protected void Page\_Load(object sender, EventArgs e)

{

\_reportName = "ReportName";

\_sessionPDFFileName = Session.SessionID.ToString() + ".pdf";

//Attach pdf to the iframe

string url = HttpContext.Current.Request.Url.Scheme + "://" + HttpContext.Current.Request.Url.Authority + HttpContext.Current.Request.ApplicationPath + \_sessionPDFFileName;

iFrameURL = url;

RegisterClientsCallBackReference();

RegisterClientCallBackAfterPrint();

this.Page.Title = \_reportName;

if ((!Page.IsPostBack))

{

rv1.ProcessingMode = Microsoft.Reporting.WebForms.ProcessingMode.Remote;

rv1.ServerReport.ReportServerUrl =

new Uri("http://Krishna-pc/reportserver");

rv1.ServerReport.ReportPath = "/Employees";

rv1.ShowPrintButton = true;

rv1.ShowRefreshButton = true;

rv1.ShowZoomControl = true;

rv1.ShowToolBar = true;

}

}

protected void ShowPrintButton()

{

string script = "<SCRIPT LANGUAGE='JavaScript'> ";

script += "showPrintButton()";

script += "</SCRIPT>";

ScriptManager.RegisterStartupScript(this, this.GetType(), "ShowStatus", "javascript:showPrintButton();showDatePicker();", true);

}

protected void SavePDF()

{

string \_reportPath = ConfigurationManager.AppSettings["ReportPath"];

Warning[] warnings;

string[] streamids;

string mimeType;

string encoding;

string extension;

rv1.LocalReport.ReportPath = \_reportPath + \_reportName;

byte[] bytes = rv1.ServerReport.Render("PDF", "", out mimeType, out encoding, out extension, out streamids, out warnings);

//save the pdf byte to the folder

if (!File.Exists(Server.MapPath(\_sessionPDFFileName)))

{

using (StreamWriter sw = new StreamWriter(File.Create(Server.MapPath(\_sessionPDFFileName))))

{

sw.Write("");

}

}

FileStream fs = new FileStream(Server.MapPath(\_sessionPDFFileName), FileMode.Open);

byte[] data = new byte[fs.Length];

fs.Write(bytes, 0, bytes.Length);

fs.Close();

}

protected void rrv1\_ReportRefresh(object sender, CancelEventArgs e)

{

ShowPrintButton();

}

public string AjaxCall(string name)

{

//var ajaxcall = Request.Form["ajaxcall"];

if (name != null)

{

if (File.Exists(Server.MapPath(\_sessionPDFFileName)))

{

frmPrint.Attributes["src"] = "";

File.Delete(Server.MapPath(\_sessionPDFFileName));

}

else

{

SavePDF();

}

}

return name;

}

string returnValue;

string ICallbackEventHandler.GetCallbackResult()

{

return returnValue;

}

void ICallbackEventHandler.RaiseCallbackEvent(string eventArgument)

{

returnValue = AjaxCall(eventArgument);

}

private void RegisterClientsCallBackReference()

{

String myClientsCallBack = Page.ClientScript.GetCallbackEventReference(this, "arg", "ServerCallSucceeded", "context", "ServerCallFailed", true);

//Could also call this wtihout the callback succeeded or failed methods:

String myCompleteClientFunction = @"function CallServerMethodBeforePrint(arg, context)

{ " +

myClientsCallBack + @";

}";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "TheScriptToCallServer", myCompleteClientFunction, true);

}

private void RegisterClientCallBackAfterPrint()

{

String myClientAfterPrintCallBack = Page.ClientScript.GetCallbackEventReference(this, "arg", "ServerCallSucceededAfterPrint", "context", "ServerCallFailed", true);

String myAfterPrintCompleteClientFunction = @"function CallServerAfterPrint(arg, context)

{ " +

myClientAfterPrintCallBack + @";

}";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "TheScriptToCallServerAfterPrint", myAfterPrintCompleteClientFunction, true);

}

protected void Button1\_Click(object sender, EventArgs e)

{

$('#printReport').click(function () {

printReport('rv1');};

}

}