Evan Doucett

Codes

C# JavaScript, JQUERY, CSS, HTML

[Images 2](#_Toc160780813)

[Images with HTML 2](#_Toc160780814)

[Open image in new window 2](#_Toc160780815)

[Iframe Image 2](#_Toc160780816)

[Busy Spinner 2](#_Toc160780817)

[Spinning Loading 2](#_Toc160780818)

[JS 2](#_Toc160780819)

[HTML 2](#_Toc160780820)

[Connection String 2](#_Toc160780821)

[IConfiguration 2](#_Toc160780822)

[String for the class 2](#_Toc160780823)

[String for the method 2](#_Toc160780824)

[DB C# Commands: 3](#_Toc160780825)

[Get from DB with Stored Procedure 3](#_Toc160780826)

[DB Command 3](#_Toc160780827)

[Controller 3](#_Toc160780828)

[Stored Procedure 3](#_Toc160780829)

[FireBird SQL Commands: 3](#_Toc160780830)

[Connect with Firebird directly 3](#_Toc160780831)

[Select returning one value 3](#_Toc160780832)

[Select returning table 4](#_Toc160780833)

[GetTable from DB 4](#_Toc160780834)

[TableCall 4](#_Toc160780835)

[Insert Into Table 4](#_Toc160780836)

[Connection Strings 5](#_Toc160780837)

[Appsettings.json 5](#_Toc160780838)

[JAVASCRIPT 6](#_Toc160780839)

[Print CSS 6](#_Toc160780840)

[Open Print 6](#_Toc160780841)

[Export to PDF 6](#_Toc160780842)

[Set above print area 6](#_Toc160780843)

[Export Table to CSV 6](#_Toc160780844)

[JQUERY to PDF 7](#_Toc160780845)

[SCRIPT TAGS 7](#_Toc160780846)

# Images

## Images with HTML

### Open image in new window

<td class="text-center">

<**a** href="~/Files/@(item.FileName)" target="\_blank">View File</**a**>

</td>

### Iframe Image

<td>

<iframe target="~/Files/@(item.FileName)" class="w-100 mh-100 embed-responsive-item modal-lg" width="1000"></iframe>

</td>

## Retrieve Image from Network

public IActionResult Download(string NAME, string PATH)

{

if (!string.IsNullOrEmpty(NAME) && !string.IsNullOrEmpty(PATH))

{

string sourceDir = PATH + "\\" + NAME;

byte[] FileBytes = System.IO.File.ReadAllBytes(sourceDir);

return File(FileBytes, "application/pdf");

}

else

{

TempData["TableMessage"] = "Missing File Information";

return View("GetTable");

}

}

## ASP.NET RAZOR HTML Layout

<div class="container ">

<div class="row">

<div class="col-3"></div>

<div class="col-md-6 boxShadow">

<**form** **asp-action**="Create" enctype="multipart/form-data" method="post">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="JOBNUMBER" class="control-label" hidden></**label**>

<**input** **asp-for**="@Model.JOBNUMBER" class="form-control" readonly hidden />

</div>

<div class="form-group">

<**label** **asp-for**="PATH" class="control-label" hidden></**label**>

<**input** **asp-for**="PATH" class="form-control" **value**="@ViewBag.PATH" readonly hidden/>

</div>

<div class="form-group">

<**label** **asp-for**="DOCTYPEID" class="control-label fw-bold"></**label**>

<**select** **asp-for**="DOCTYPEID" **name**="DOCTYPEID" class="form-control bg-yellow">

@foreach (JobModel a in ViewBag.DOCTYPE\_ID)

{

<**option** **value**="@a.DOCTYPEID" id="@a.DOCTYPEID">@a.DOCTYPE / @a.DOCTYPEID</**option**>

}

</**select**>

</div>

<div class="form-group">

<**label** **asp-for**="DOCFILE" class="control-label fw-bold"></**label**>

<**input** **asp-for**="DOCFILE" **type**="file" class="form-control bg-yellow " accept="application/pdf" required />

<**span** **asp-validation-for**="DOCFILE" class="text-danger"></**span**>

</div>

<div class="form-group m-2">

<input type="submit" value="Upload Document" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

</div>

<div>

</div>

## Delete an Image

public IActionResult Delete(JobModel jobModel)

{

ViewBag.JOBNUMBER = jobModel.JOBNUMBER;

ViewBag.ATTACHMENT\_ID = jobModel.ATTACHMENT\_ID;

string pathOld = @"\\morfsdc1\CustomerDocs\" + jobModel.JOBNUMBER + @"\RemovedOrDeleted";

string path = @"\\morfsdc1\CustomerDocs\" + jobModel.JOBNUMBER + @"\";

string fileNameWithPathOld = Path.Combine(path + "RemovedOrDeleted\\" + jobModel.NAME);

string fileNameWithPath = Path.Combine(path + jobModel.NAME);

if (System.IO.File.Exists(fileNameWithPath))

{

if (!Directory.Exists(pathOld))

{

Directory.CreateDirectory(pathOld);

}

System.IO.File.Copy(fileNameWithPath, fileNameWithPathOld, true);

System.IO.File.Delete(fileNameWithPath);

TempData["CreateMessage"] = "Delete successful";

return RedirectToAction("GetTable", new { JOBNUMBER = jobModel.JOBNUMBER, SITEADDRESS1 = jobModel.SITEADDRESS1 });

}

else

{

System.IO.File.Delete(fileNameWithPath);

TempData["CreateMessage"] = "Delete successful";

return RedirectToAction("GetTable", new { JOBNUMBER = jobModel.JOBNUMBER, SITEADDRESS1 = jobModel.SITEADDRESS1 });

}

//homeManager.DeleteDocPathAndDoc(Convert.ToInt32(jobModel.ATTACHMENT\_ID));

}

## Create a Folder

if (System.IO.File.Exists(fileNameWithPath))

{

if (!Directory.Exists(pathOld))

{

Directory.CreateDirectory(pathOld);

} }

# Busy Spinner

## Spinning Loading

### JS

function displayBusyIndicator() {

document.getElementById("loading").style.display = "block";

}

### HTML

<div class="d-flex justify-content-center ">

<div class="spinner-border text-primary mb-3"

style="width: 6rem; height: 6rem; display: none"

role="status" id="loading">

<br />

<span class="sr-only ms-2"> Loading...</span>

</div>

</div>

# Connection String

## IConfiguration

### String for the class

private IConfiguration connString = new ConfigurationBuilder().SetBasePath(Directory.GetCurrentDirectory()).AddJsonFile("appsettings.json").Build();

### String for the method

string config = connString.GetConnectionString("BMTWMS")!;

# DB C# Commands:

## Get from DB with Stored Procedure

### DB Command

public string GetLastEntryByUser(string username1)

{

string username = username1;

string config = OeDbConnection();

string? result = "";

using (SqlConnection connection = new SqlConnection(config))

{

SqlCommand command = connection.CreateCommand();

command.CommandType = CommandType.StoredProcedure;

command.CommandText = "spr\_LastEntry";

command.Parameters.AddWithValue("@RequestedBy", username);

command.Parameters.Add("@OutputMessage", SqlDbType.VarChar, 50).Direction = ParameterDirection.Output;

connection.Open();

command.ExecuteNonQuery();

result = command.Parameters["@OutputMessage"].Value.ToString();

connection.Close();

}

return result;

}

### Controller

var OElist = requestmanager.GetViewOERequest(id, username, abrv);

### Stored Procedure

Create or ALTER procedure [dbo].[spr\_LastEntry]

@RequestedBy varchar (50),

@OutputMessage varchar (50) output

as

begin

declare @rowcount int = 0,

@value varchar (15)= (select convert(varchar(10),max(RequestID)) from TblRequests where RequestedBy = @RequestedBy);

begin try

set @rowcount = (select count(1) from dbo.TblRequests where RequestedBy = @RequestedBy)

if(@rowcount > 0)

begin

begin tran

set @OutputMessage = @value

commit tran

end

else

begin

set @OutputMessage = 0

end

end try

begin catch

rollback tran

set @OutputMessage = error\_message()

end catch

end

# FireBird SQL Commands:

## Helpful Links

### <https://www.firebirdfaq.org/faq348/>

### <https://csharp.hotexamples.com/examples/FirebirdSql.Data.FirebirdClient/FbCommand/-/php-fbcommand-class-examples.html>

### <https://firebirdsql.org/file/documentation/html/en/refdocs/fbdevgd30/firebird-30-developers-guide.html#fbdevgd30-efw>

### NuGet Package: FirebirdSql.Data.FirebirdClient

## Connect with Firebird directly

### Select returning one value

public int FindMaxAttachmentNumber()

{

int value = 0;

string config = connString.GetConnectionString("BMTWMS")!;

List<JobModel> listOfOERequest = new List<JobModel>();

using (FbConnection connection = new FbConnection(config))

{

connection.Open();

using (FbCommand command = new FbCommand("SELECT MAX(ATTACHMENT\_ID) as ATTACHMENT\_ID FROM JOB\_ATTACHMENTS ", connection))

{

using (FbDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

value = reader.GetInt32(0);

return value;

}

return value;

}

}

}

}

## Select returning table

### GetTable from DB

Example One

public List<JobModel> GetJobNumberByAddress(string Address)

{

string varAddress = Address.ToUpper().Replace("\*","").Replace("$", "").Replace("^", "").Replace("(", "").Replace(")", "").Trim();

string config = connString.GetConnectionString("BMTWMS")!;

List<JobModel> listOfOERequest = new List<JobModel>();

DataTable dt = new DataTable();

FbDataAdapter da = new FbDataAdapter(

"select AREA.AREA\_DESC AS DESC, " +

"JOB.JOB\_NUMBER AS JOB\_NUMBER, " +

"JOB.SITEADDRESS1 AS SITEADDRESS1, " +

"JOB.HEADING1 AS OWNERNAME, " +

"JOB\_ATTACHMENTS.PATH, " +

"JOB\_ATTACHMENTS.NAME, " +

"DOCTYPES.DOCTYPE " +

"FROM JOB " +

"INNER JOIN AREA on AREA.AREA\_ID = JOB.AREA\_ID " +

"INNER JOIN JOB\_ATTACHMENTS on JOB\_ATTACHMENTS.JOB\_NUMBER = JOB.JOB\_NUMBER " +

"LEFT JOIN DOCTYPES on DOCTYPES.DOCTYPE\_ID = JOB\_ATTACHMENTS.DOCTYPE\_ID " +

"where UPPER(JOB.SITEADDRESS1) = '" + varAddress + "'"

, config);

da.Fill(dt);

da.Dispose();

foreach (DataRow dr in dt.Rows)

{

listOfOERequest.Add(new JobModel

{

JOBNUMBER = (dr["JOB\_NUMBER"] is not DBNull) ? dr["JOB\_NUMBER"].ToString() : null,

Ownername = (dr["OWNERNAME"] is not DBNull) ? dr["OWNERNAME"].ToString() : null,

Address = (dr["SITEADDRESS1"] is not DBNull) ? dr["SITEADDRESS1"].ToString() : null,

Community = (dr["DESC"] is not DBNull) ? dr["DESC"].ToString() : null,

PATH = (dr["PATH"] is not DBNull) ? dr["PATH"].ToString() : null,

NAME = (dr["NAME"] is not DBNull) ? dr["NAME"].ToString() : null,

DOCTYPE = (dr["DOCTYPE"] is not DBNull) ? dr["DOCTYPE"].ToString() : null,

DOCTYPE\_ID = Convert.ToInt32(dr["ATTACHMENT\_ID"]),

});

}

return listOfOERequest;

}

### Second Example

public List<JobModel> GetJobSearch(string Job)

{

string varJobs = Job.ToUpper().Replace(";", "").Replace("$", "").Replace("%", "").Replace("^", "")

.Replace("\*", "").Replace("(", "").Replace(")", "").Replace(" ", "").Replace("-", "").Replace("\_", "")

.Trim();

string config = connString.GetConnectionString("BMTWMS")!;

List<JobModel> listOfOERequest = new List<JobModel>();

DataTable dt = new DataTable();

FbDataAdapter da = new FbDataAdapter(

"select AREA.AREA\_DESC AS DESC, " +

"JOB.JOB\_NUMBER AS JOB\_NUMBER, " +

"JOB.SITEADDRESS1 AS SITEADDRESS1, " +

"JOB.HEADING1 AS OWNERNAME, " +

"JOB\_ATTACHMENTS.PATH, " +

"JOB\_ATTACHMENTS.NAME, " +

"DOCTYPES.DOCTYPE, " +

"JOB\_ATTACHMENTS.ATTACHMENT\_ID " +

"FROM JOB " +

"INNER JOIN AREA on AREA.AREA\_ID = JOB.AREA\_ID " +

"INNER JOIN JOB\_ATTACHMENTS on JOB\_ATTACHMENTS.JOB\_NUMBER = JOB.JOB\_NUMBER " +

"LEFT JOIN DOCTYPES on DOCTYPES.DOCTYPE\_ID = JOB\_ATTACHMENTS.DOCTYPE\_ID " +

"where JOB.JOB\_NUMBER = '" +varJobs+ "'"

, config);

da.Fill(dt);

da.Dispose();

foreach (DataRow dr in dt.Rows)

{

listOfOERequest.Add(new JobModel

{

JOBNUMBER = (dr["JOB\_NUMBER"] is not DBNull) ? dr["JOB\_NUMBER"].ToString() : null,

Ownername = (dr["OWNERNAME"] is not DBNull) ? dr["OWNERNAME"].ToString() : null,

Address = (dr["SITEADDRESS1"] is not DBNull) ? dr["SITEADDRESS1"].ToString() : null,

Community = (dr["DESC"] is not DBNull) ? dr["DESC"].ToString() : null,

PATH = (dr["PATH"] is not DBNull) ? dr["PATH"].ToString() : null,

NAME = (dr["NAME"] is not DBNull) ? dr["NAME"].ToString() : null,

DOCTYPE = (dr["DOCTYPE"] is not DBNull) ? dr["DOCTYPE"].ToString() : null,

ATTACHMENT\_ID = Convert.ToInt32(dr["ATTACHMENT\_ID"]),

});

}

return listOfOERequest;

}

## Creates a list from the GetJobSearch

public List<JobModel> GetTableSelection(string Jobs)

{

string varJobs = Jobs.ToUpper().Replace(";", "").Replace("$", "").Replace("%", "").Replace("^", "")

.Replace("\*", "").Replace("(", "").Replace(")", "").Replace(" ", "").Replace("-", "").Replace("\_", "")

.Trim();

List<JobModel> jobs = GetJobSearch(varJobs);

List<JobModel> jobList = (

from job in jobs

select new JobModel

{

JOBNUMBER = job.JOBNUMBER,

Address = job.Address,

Community = job.Community,

PATH = job.PATH,

NAME = job.NAME,

Ownername = job.Ownername,

DOCTYPE = job.DOCTYPE,

ATTACHMENT\_ID = job.ATTACHMENT\_ID,

}).ToList();

return jobList;

}

## Select Records

public List<JobModel> GetTableSelection(string Jobs)

{

string varJobs = Jobs.ToUpper().Replace(";", "").Replace("$", "").Replace("%", "").Replace("^", "")

.Replace("\*", "").Replace("(", "").Replace(")", "").Replace(" ", "").Replace("-", "").Replace("\_", "")

.Trim();

List<JobModel> jobs = GetJobSearch(varJobs);

List<JobModel> jobList = (

from job in jobs

select new JobModel

{

JOBNUMBER = job.JOBNUMBER,

Address = job.Address,

Community = job.Community,

PATH = job.PATH,

NAME = job.NAME,

Ownername = job.Ownername,

DOCTYPE = job.DOCTYPE,

ATTACHMENT\_ID = job.ATTACHMENT\_ID,

}).ToList();

return jobList;

}

## Insert Record

Firebird does not auto increment

public void InsertImageToTable(JobModel jobModel)

{

string? name = jobModel.NAME;

string? path = jobModel.PATH;

string? jobNumber = jobModel.JOBNUMBER;

string? doctype\_id = jobModel.DOCTYPE;

string config = connString.GetConnectionString("BMTWMS")!;

int ValueMax = FindMaxAttachmentNumber()+1;

FbCommand cmd = new FbCommand("INSERT into JOB\_ATTACHMENTS (ATTACHMENT\_ID,JOB\_NUMBER, PATH, NAME, DOCTYPE\_ID) values (@ATTACHMENT\_ID, @JOB\_NUMBER,@PATH, @NAME, @DOCTYPE\_ID);");

cmd.CommandType = CommandType.Text;

cmd.Parameters.Add("@ATTACHMENT\_ID", ValueMax);

cmd.Parameters.Add("@NAME", name);

cmd.Parameters.Add("@PATH", path);

cmd.Parameters.Add("@JOB\_NUMBER", jobNumber);

cmd.Parameters.Add("@DOCTYPE\_ID", doctype\_id);

using (cmd.Connection = new FbConnection(config))

{

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

}

}

## Update Table

public void InsertDocPathToTable(string JOBNUMBER)

{

string? path = @"\\morfsdc1\CustomerDocs2\" + JOBNUMBER + "\\";

//string? jobNumber = JOBNUMBER;

string config = connString.GetConnectionString("BMTWMS")!;

FbCommand cmd = new FbCommand("UPDATE job SET DOCS\_FOLDER = @PATH WHERE JOB\_NUMBER = @JOBNUMBER");

cmd.CommandType = CommandType.Text;

cmd.Parameters.Add("@PATH", path);

cmd.Parameters.Add("@JOBNUMBER", JOBNUMBER);

using (cmd.Connection = new FbConnection(config))

{

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

}

}

## Delete Record

public void DeleteDocPathAndDoc(int ATTACHMENT\_ID)

{

string config = connString.GetConnectionString("BMTWMS")!;

FbCommand cmd = new FbCommand("DELETE FROM JOB\_ATTACHMENTS WHERE ATTACHMENT\_ID = @ATTACHMENT\_ID; ");

cmd.CommandType = CommandType.Text;

cmd.Parameters.Add("@ATTACHMENT\_ID", ATTACHMENT\_ID);

using (cmd.Connection = new FbConnection(config))

{

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

}

}

# Connection Strings

## Appsettings.json

"AllowedHosts": "\*",

"ConnectionStrings": {

"OperatingExpenseConnString": "Data Source=MORSQL;Initial Catalog=OperatingExpense;User Id=bpm\_user;Password=resu\_mpb1; TrustServerCertificate=True",

"VmortConnString": "Data Source=VMORTL\\SQLEXPRESS;Initial Catalog=TimberlineLink;User Id=ITMain;Password=M0rr1s0n1961; TrustServerCertificate=True",

"SMTP": "10.0.0.26",

"PORT": "25",

"FROM": "oe@morrisonhomes.ca",

"ABGst": "5",

"Exportlocation": "C:/Users/edoucett/Desktop/"

}

# JAVASCRIPT

## Print CSS

@media print {

@page {

margin-left: 0.0in;

margin-right: 0.0in;

margin-top: 0;

margin-bottom: 0;

}

}

@media print {

.print-job {

display: none;

}

}

## Open Print

function printscreen() {

window.print();

}

<span class="float-end pt-2 print-job">

<button type="button" onclick="printscreen()" class=" col-2 m-3 btn-primary print-job">Print Job</button>

</span>

## Export to PDF

function generatePDF() {

// Choose the element id which you want to export.

var InvoiceId = document.getElementById('jobid').outerText;

var element = document.getElementById('ObjectToPrint');

// element.style.width = '816px';

// element.style.height = '1054px';

var opt = {

margin: 0.25,

filename: InvoiceId + '.pdf',

image: { type: 'jpeg', quality: 1 },

html2canvas: { scale: 1 },

jsPDF: { unit: 'in', format: 'legal', orientation: 'landscape', precision: '12' }

};

// choose the element and pass it to html2pdf() function and call the save() on it to save as pdf.

html2pdf().set(opt).from(element).save();

}

<span class="float-end pt-2 print-job">

<button type="button" onclick="generatePDF()" class="col-2 m-3 btn-danger print-job">Export PDF</button>

</span>

### Set above print area

<div id="ObjectToPrint"> </div>

### Remove element from export

<td class="print-job ignoreColumn ignore" data-html2canvas-ignore="true">

## Export Table to CSV

function downloadCSV(csv) {

var csvFile;

var filename = document.getElementById('jobid').outerText;

var downloadLink;

// CSV file

csvFile = new Blob([csv], { type: "text/csv" });

// Download link

downloadLink = document.createElement("a");

// File name

downloadLink.download = filename;

// Create a link to the file

downloadLink.href = window.URL.createObjectURL(csvFile);

// Hide download link

downloadLink.style.display = "none";

// Add the link to DOM

document.body.appendChild(downloadLink);

// Click download link

downloadLink.click();

}

function exportTableToCSV() {

var filename = document.getElementById('jobid').outerText;

var csv = [];

var rows = document.querySelectorAll("table tr");

for (var i = 0; i < rows.length; i++) {

var row = [], cols = rows[i].querySelectorAll("td, th");

for (var j = 0; j < cols.length; j++)

row.push(cols[j].innerText);

csv.push(row.join(","));

}

// Download CSV file

downloadCSV(csv.join("\n"), filename);

}

## 

## JQUERY to PDF

$("body").on("click", "#btnExport", function () {

html2canvas($('#ObjectToPrint')[0], {

onrendered: function (canvas) {

var data = canvas.toDataURL();

var docDefinition = {

content: [{

image: data,

width: 500

}]

};

pdfMake.createPdf(docDefinition).download("cutomer-details.pdf");

}

});

});

## 

## SCRIPT TAGS

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.22/pdfmake.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/0.4.1/html2canvas.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/1.5.3/jspdf.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.9.3/html2pdf.bundle.min.js"></script>

<**script** **src**="~/js/ExportToPdf.js" **asp-append-version**="true" type="text/javascript"></**script**>

<**script** **src**="~/js/ExportToCSV.js" **asp-append-version**="true" type="text/javascript"></**script**>

# ASP.NET

## TempData

### HTML

<div class="container-fluid boxShadow">

<div class="row">

<div class="col-8 ">

<h4 class="">Upload an attachment a Document for @ViewBag.JOBNUMBER</h4>

<**a** **asp-action**="GetTable" **asp-area**="" **asp-controller**="Home" **asp-route-JOBNUMBER**="@ViewBag.Job" **asp-route-SITEADDRESS1**="@ViewBag.Address" class="btn btn-primary m-1"><h5>Back to Images</h5></**a**>

</div>

<div class="col-4"></div>

</div>

</div>

@if (TempData["CreateMessage"] != null)

{

<div class="container-flex d-flex justify-content-center boxShadow bg-info">

<div class="row">

<div class="" id="top">

<h5 class="text-black">@TempData["CreateMessage"]</h5>

</div>

</div>

</div>

}

else

{

<div></div>

}

### Controller

TempData["CreateMessage "] = "Missing File Information";

### Model Annotation

[FileExtensions(Extensions = "pdf, PDF")]

[Required(ErrorMessage = "Please select file")]

[DisplayName("Image Upload (pdf only)")]

[NotMapped]

public IFormFile? DOCFILE { get; set; }

# Creating Files

## File Info

https://learn.microsoft.com/en-us/dotnet/api/system.io.fileinfo?view=net-8.0

FileInfo fileInfo = new FileInfo(jobModel.DOCFILE.FileName);

## System IO

string fileNameWithPath = Path.Combine(path + fileInfo);

System.IO.File.Exists(fileNameWithPath)

### Creating a folder and moving a file if exist

if (System.IO.File.Exists(fileNameWithPath))

{

if (!Directory.Exists(pathOld))

{

Directory.CreateDirectory(pathOld);

}

using (var stream = new FileStream(fileNameWithPath, FileMode.Create))

{

System.IO.File.Copy(fileNameWithPath, fileNameWithPathOld, true);

jobModel.DOCFILE.CopyTo(stream);

TempData["CreateMessage"] = "Upload successful";

return View("Create");

}

}

else

{

using (var stream = new FileStream(fileNameWithPath, FileMode.Create))

{

jobModel.DOCFILE.CopyTo(stream);

homeManager.InsertImageToTable(jobModel);

TempData["CreateMessage"] = "Upload successful";

return View("Create");

}

}