*・ユーザー検索画面のメール送信ボタンが押された時*

FindUserController：

＠RequestMapping(value="/SendMail", method = RequestMethod.GET)に飛ぶ。

そのあと下に順繰り降りていく。

＊if(session.getAttribute("session1")==null) {で、session.getAttribute("session1")がnullなら、

SessionTimeout.jspに行き、セッションがタイムアウトしましたとエラーが出る。

何事もなければFolderPathを通って、SendMail.jspに返す。

別ウィンドウでメール送信画面が表示される。

**修正箇所のレビュー**

・3つ目の「添付ファイル3」ボタンを追加

SendMail.jspの「添付ファイル2」のHTMLの下に、

<div class="row 添付row">

<input type="file" id="file3" style="display:none;" onchange="$('#fake\_input\_file3').val($(this).prop('files')[0].name)" name="file3">

<input type="button" class="col-xs-3 button1" value="添付ファイル3" id="id添付ファイル3btn" onClick="$('#file3').click()" />

<div class="input添付div" id="divfile3">

<input id="fake\_input\_file3" class="input添付" readonly type="text" value="" id="添付ファイル名file3" >　←ファイルをインプットしてくれる＆添付ファイルが選択されると、「onchange」イベントの処理が実行され、「fake\_input\_file3」のIDを持つテキストボックスに選択されたファイル名を設定してファイル名を画面に表示される

</div>

<div class="input添付div">

<input type="button" class="添付ファイル3 btnCancel" value="X" id="btnCancelfile3" onclick="clearValue('file3');" />　←ファイル名横の×ボタンのコード。×ボタンを押すとファイルをキャンセルできる。

</div>

</div>

を追加。これで画面に「添付ファイル3」を表示できる。

*・添付ファイルのボタン処理の流れ*

１．クライアント）添付ファイルのボタンをクリック　:SendMail.jsp

<input type="file" id="file3" style="display:none;" onchange="$('#fake\_input\_file3').val($(this).prop('files')[0].name)" name="file3">

<input type="button" class="col-xs-3 button1" value="添付ファイル3" id="id添付ファイル3btn" onClick="$('#file3').click()" />

2．クライアント）添付ファイルに挿入したファイル名と「X」ボタンを表示　:SendMail.jsp

\*添付ファイルが選択されると、「onchange」イベントの処理が実行され、「fake\_input\_file3」のIDを持つテキストボックスに選択されたファイル名を設定してファイル名を画面に表示される

<div class="input添付div">

<a herf="" id="fake\_input\_file3" class="input添付"></a>

</div>

<div class="input添付div">

<input type="button" class="添付ファイル3 btnCancel" value="X" id="btnCancelfile3" onclick="clearValue('file3');" />

</div>

</div>

3．クライアント）サーバーにアップロードするメソッド

Jqueryで登録された「change」イベントの処理が実行される　:SendMail.jsp

$(function(){

$("#file1,#file2,#file3").change(proc);

function proc() {

//alert($(this).attr("id"));

fileSetted($(this).attr("id"))　←「fileSetted」メソッドを呼んで、「✖」ボタンを表示される

if (window.FormData){ //　FormDataにブラウザが対応しているかチェック

var fileObj = $('#'+ $(this).attr("id"))[0].files[0]; // ファイルオブジェクトの取り出し

if ( fileObj != null ){

fileupload(fileObj,$(this).attr("id"));　←「fileupload」メソッドを呼んで、サーバーに添付ファイルをアップロードする

}

}else{

alert("このブラウザはFormDataに対応していません。");

return null;

}

}

4.クライアント）サーバーにリクエストを送る　:SendMail.jsp

function fileupload(fileObj,number){

var fd = new FormData(); // FormDataオブジェクト生成　←仮想フォーム(<form>)を作成

fd.append('file', fileObj);　←仮想フォームに添付ファイルを入れる

＊fdはFormDataの略。Appendはデータの追加。

　fd.append('number',number);　←仮想フォームにファイル番号を入れる

$.ajax({

url : "${pageContext.request.contextPath}/SendMailFileUpload",

↑サーバーの「SendMailFileUpLoad」にリクエストと仮想フォームを送信する。

　type : "post",

　data : fd,

　cache: false,　←キャッシュを無効化

　contentType: false,　←インターネットでファイルのダウンロードを行う時、「これから返すファイルの種類」をContent-Typeによって示すことで、ファイルの種類を受け取る側が知ることができる。

　processData : false　←Dataに余計な手を加えずそのまま送る（クエリ文字列化しない）

})

何事もなく成功していれば↓

.done( function(text){

console.log(text);

});

に行き、添付されたファイルについて、エクリプス側のコンソールにテキストが打ち込まれる。

C:\Users\user\Documents\画面作成\2.資料-20240408T072147Z-001\2.資料\10.OfficeSystem\_Hibernate\_改修元\10.OfficeSystem\_Hibernate\_改修元\src\main\webapp\upload\202405091111\file1

org.springframework.web.multipart.commons.CommonsMultipartFile@14335179

5．サーバー）リクエストに対応するメソッドを実行　:FindUserController

@RequestMapping(value="/SendMailFileUpload", method = RequestMethod.POST)

public String SendMailFileUpload(HttpServletRequest request, HttpSession session, Model model

, @RequestParam(required = false) MultipartFile file

, @RequestParam(required = false) String number) {

//System.out.println(System.getProperty("file.encoding"));

6.サーバー）クライアントから届いた添付ファイルを指定したフォルダーに保存

//Webアプリフォルダーを取得

ServletContext context = request.getServletContext();

String appPath =context.getRealPath("");

// ファイル種類から決まる値をセットする

StringBuffer filePath = new StringBuffer("C:＼Users＼USER＼Desktop＼a＼");

↑Cドライブにファイルが保存される。

…

}

7.サーバー）リクエストに対応するメソッドの実行結果をクライアントに送る

return "SendMail";

}

8.クライアント）サーバーの処理の実行結果に応じて処理

成功したら、

.done( function(text){

console.log(text);

});

に行き、エクリプス側にコンソールでテキストが打ち込まれる

＊４と同じ

以上で、添付ファイルボタンを押して、ファイルをインプットする処理が整った。

*・添付ファイルのダウンロード*

1. クライアント）ファイル名、ボタンまたはリンクをクリックする

<div class="row 添付row">

<input type="file" id="file3" style="display:none;" onchange="setLink('file3',$(this).prop('files')[0].name)" required>

<input type="button" class="col-xs-3 button1" value="添付ファイル3" id="id添付ファイル3btn" onClick="$('#file3').click()" />

</div>

2．クライアント）サーバーにダウンロードリクエストを送信　:SendMail.jsp

ブラウザ内で、ファイル名を右クリックし、「リンクのアドレスをコピー」

　貼り付けたリンクをクリックすると、サーバーにリクエストを送る。

多分） /\*リンクを設定する\*/

function setLink(id, filename){

//ファイル名をリンクの表示内容として設定

$("#fake\_input\_" + id).html(filename);

//リンクを設定

$("#fake\_input\_" + id).attr("href",

"${pageContext.request.contextPath}/DownloadFile?id="+id+"&name="+filename

);

↑("#fake\_input\_" + id)に"${pageContext.request.contextPath}/

DownloadFile?id="+id+"&name="+filename　を上書き。

}

3．サーバー）ダウンロードリクエストを対応するメソッドを実行　:FindUserController

@RequestMapping(value="/DownloadFile", method = RequestMethod.GET)

public void DownloadFile(

HttpServletRequest request

,HttpServletResponse response

,@RequestParam(required = false) String id

,@RequestParam(required = false) String name) throws Exception{

File file = null;

switch(id) {

case("file1"):

file = new File(path1);

break;

case("file2"):

file = new File(path2);

break;

case("file3"):

file = new File(path3);

break;

↑Swichで条件分岐。

IdがFile1の場合、 file = new File(path1);でFileオブジェクトを作成する。

}

//ファイル名をUTF-8の文字コードに変換する

String fileName = URLEncoder.encode(file.getName(),"UTF-8");

//ファイルのタイプを取得

String mimeType = URLConnection.guessContentTypeFromName(fileName);

//サーバーからダウンロードされるファイルのタイプをクライアントに教える

response.setContentType(mimeType);

//クライアントでダウンロードされるファイル名を指定する

response.setHeader("Content-Disposition","attachment;filename\*=utf-8''" + fileName);

response.setContentLength((int)FileUtils.sizeOf(file));

InputStream inputStream = new FileInputStream(file);

4．サーバー）クライアントにファイルを送る

//ファイルの部分ごとをクライアントに送信する

FileCopyUtils.copy(inputStream,response.getOutputStream());

//ダウンロード処理を完了させる

response.flushBuffer();

　　　＊バッファをフラッシュする（バッファをディスクに書き出すこと）

inputStream.close();

}

}

5．クライアント）ダウンロードしたファイルを保存

*・送信ボタンの処理*

1．クライアント）送信ボタンをクリック

$("#btn送信").click(function (){

2．クライアント）データ内容チェック（宛名、件名、内容）

　if ($("#destination").val() == "") {

alert("宛先を入力してください");

return false;

}

　if ($("#subject").val() == "") {

alert("件名を入力してください");

return false;

}

　if ($("#mailContent").val() == "") {

alert("件名を入力してください");

return false;

}

3．クライアント）メール送信リクエストとデータをWebサーバーに送信

　$.ajax({

url : "${pageContext.request.contextPath}/SendMail",

type : "post",

data : formdata,

cache : false,

contentType: false,

processData : false

})

↑FormDateはこのAjaxの上にあるVar　form＝～Var　formdata＝～でデータを取得している。

4．サーバー）メール送信リクエストを対応するメソッドを実行

@RequestMapping(value="/SendMail", method = RequestMethod.POST)

public String SendMail(HttpSession session, Model model

, @RequestBody @RequestParam(required = false) String destination

, @RequestBody @RequestParam(required = false) String subject

, @RequestParam(required = false) String mailContent

, @RequestParam(required = false) String address) {

System.out.println(address);

↑メール送信ボタンをクリックすると、System.out.println(address);に行く

デバッグでSystem.out.println(address);を通り過ぎるとnullと出るため、

Addressはnullだとわかる。

↓クライアントのリクエストを対応するメソッド↓

5．サーバー）メールサーバー情報を指定し、アカウントとの情報(メールサーバー名、ポート番号、セキュリティ有無、ユーザー名、パスワード)とメール情報(送信元、宛先、件名、内容、添付ファイル)を設定

MultiPartEmail mail = new MultiPartEmail();

//ここでGmailを利用しているが、他のメールを利用しても問題ない↓

mail.setHostName("smtp.gmail.com");

mail.setSmtpPort(587);

mail.setStartTLSEnabled(true);

6．サーバー）指定されたメールサーバー情報を利用して、アカウント情報とメール情報をメール送信リクエストと一緒にメールサーバーに送信

//アカウント情報の設定

mail.setAuthentication("toyamakunn0930", "cfsd jfqk vlqc psgi");

//メール情報の設定

mail.setFrom("toyamakunn0930@gmail.com");←宛先

mail.setSubject(subject);←件名

mail.setMsg(mailContent);←内容

mail.setCharset("UTF-8");←文字コード

String[] dest = destination.split(";",0);

for (int i = 0; i < dest.length; i++) {

mail.addTo(dest[i]);

宛先１：～＠yahoo.co.jp\n宛先２：～＠yahoo.co.jpのような形式で保存で処理したのを、メールのToに加える。

}

…

　　//メールサーバーにリクエストを送る

mail.send();

7．メールサーバー）アカウント情報をチェックし、メールシステムにログインさせてメール情報を利用して、メールを送信

多分Google側？

8．メール送信結果を返す

多分Google？

9.メール送信結果を加工して、加工した結果を返す

//メールサーバーから届いたエラー結果を処理して、クライアントにエラーメッセージを表示させるようにする

} catch (IOException e) {

} catch (EmailException e) {

// TODO 自動生成された catch ブロック

e.printStackTrace();

JFrame frame = new JFrame("タイトル");

frame.setBounds(100, 100, 300, 250);

frame.setLocationRelativeTo(null);

//frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setAlwaysOnTop(true);

JPanel p = new JPanel();

p.setSize(200,150);

JLabel label = new JLabel();

label.setPreferredSize(new Dimension(200,150));

label.setText("EmailExceptionが発生しました");

label.setHorizontalAlignment(JLabel.CENTER);

label.setVerticalAlignment(JLabel.CENTER);

label.setFont(new Font("ＭＳ ゴシック", Font.BOLD, 24));

JButton button = new JButton();

button.setBounds(50, 50, 50, 30);

button.setText("OK");

frame.add(p);

p.add(label);

//p.add(button);

frame.setVisible(true);

return "";

｝

//メールサーバーの処理が成功した場合、クライアントにメール送信完了画面を返す

return "SendMailFinish";

}

10.メール送信結果を表示する

SendMail.jspのWebサーバーにPOSTリクエストを送った後の場所に移る。

/\*メール送信結果は成功の結果なら"SentMailFinish"の画面に遷移。間違っていれば、エラーメッセージを表示\*/

.done(function(data, textStatus, jqXHR){

console.error(jqXHR.responseText);

window.location.href = "SendMailFinish";

↑SendMailFinish.jspにジャンプ（だから別ウィンドウ？？）

//alert(data);

})

.fail(function(jqXHR, textStatus, errorThrown){

console.error(jqXHR.responseText);

alert(jqXHR.responseText);

});

↑間違ったメールアドレスを入力した場合、エラーメッセージがエクリプスのコンソールに打ち込まれる。

Caused by: javax.mail.internet.AddressException: Domain ends with dot in string ``toyamakunn0930@gmail.''

at javax.mail.internet.InternetAddress.checkAddress(InternetAddress.java:1333)

at javax.mail.internet.InternetAddress.parse(InternetAddress.java:1116)

at javax.mail.internet.InternetAddress.parse(InternetAddress.java:655)

at javax.mail.internet.InternetAddress.<init>(InternetAddress.java:115)

at org.apache.commons.mail.Email.createInternetAddress(Email.java:1949)

... 59 more

});

*・想定できる質問*

もし、ファイルを新たに増やすならどうする？

添付ファイル３を増やした時のようにSendMail.jspとFindUserControllerのファイル3の部分をコピペして適宜貼り付ける。

もし、ボタンを増やせと言われたらどうする？

どのような機能を持つボタンかにもよるが、

ボタンの表示はSendMail.jspを中心に入力し、例えばログイン画面に戻る仕組みなら、SendMail.jspにリクエスト先を入力し、Controllerにリクエストマッピングを追加して処理を行えば、できそうではある。