



PROJETO CUSTOMIZAÇÕES

Gera Excel com dados do processo

Amanda Alvim

1. Objetivo

O objetivo dessa customização é todos dias 1 e 16 enviar um e-mail para todas as pessoas de um determinado grupo com um excel contendo todos os dados dos processos, linha por linha da grid.

O robô contendo essa implementação se encontra no Projeto Base no nosso git (<http://git.lecom.com.br/PSP/Projeto-Base-BPM>) com o nome de **RoboMontaRelatorio**, que está no package com.lecom.workflow.cadastrs.robexemplos, existe também o properties de mesmo nome RoboMontaRelatorio no diretório upload/cadastro/config do projeto e também no diretório do projeto há um arquivo como parâmetro de excel, seria uma cara a qual o arquivo vai começar e depois ganhar os valores que forem pegando do banco, esse arquivo chama Relatorio_Exemplo.xlsx e está no diretório /upload/cadastro/excelDefault/.

2. Como usar

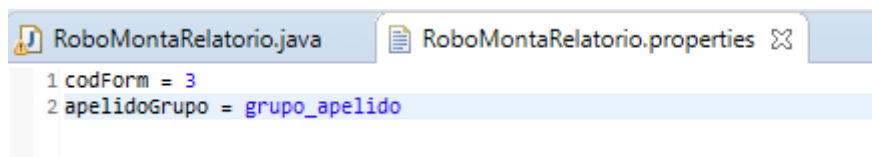
No momento em que for utilizar essa implementação será necessário onde tem informações da tabela de modelo, campos, grid para o seu modelo em questão.

Temos uma planilha modelo conforme mencionado acima, você precisará pegar ela alterar para as colunas que deseja ter e colocar posteriormente essa planilha no diretório do servidor do cliente podendo ser diretamente /opt/lecom/app/tomcat/webapps/bpm/upload/cadastrs para ambientes com o modo antigo de atualização das customizações ou no diretório /opt/lecom/custom/web-content para modo novo de atualização, assim você precisa criar o diretório excelDefault e colocar essa planilha modelo.

Deverá fazer a mesma coisa com o arquivo de properties colocando-o no servidor na pasta de config após alterar para as informações do seu processo, seja ele no modo antigo ou no modo novo de atualização das customizações.

3. Properties

O properties usado nessa customização é bem simples, mas facilita para que quando você for colocar na produção você altere apenas o properties e não tenha que gerar o fonte novamente com a informação atualizada.



```
RoboMontaRelatorio.java  RoboMontaRelatorio.properties ✕
1 codForm = 3
2 apelidoGrupo = grupo_apelido
```

Acima precisamos preencher no properties somente dois parâmetros:

- codForm: código do seu formulário, para que seja utilizado para encontrar as instancias.
- apelidoGrupo: irá informar o apelido do grupo de usuários cadastrados no admin que irá receber esse e-mail contendo o excel com todos os dados.

4. Fonte Java

Nesse arquivo temos exemplo de como pegar os dados do processo e montar um excel se baseando em um modelo com os títulos para apenas preencher o conteúdo.

```

RoboMontaRelatorio.java  RoboMontaRelatorio.properties
41 @RobotModule(value = "RoboMontaRelatorio")
42 @Version({1,0,0})
43 public class RoboMontaRelatorio implements WebServices{
44
45     private static final Logger logger = Logger.getLogger(RoboMontaRelatorio.class);
46     // variáveis que referenciam as colunas da folha do excel
47     private static String COL_PROCESSO = "A";
48     private static String COL_ABERTURA = "B";
49     private static String COL_ETAPA = "C";
50     private static String COL_STATUS = "D";
51     private static String COL_SOLIC = "E";
52     private static String COL_NOME = "F";
53     private static String COL_TIPO = "G";
54     private static String COL_PREVISAO = "H";
55     private static String COL_MATERIA = "I";
56     private static String COL_QUANTIDADE = "J";
57     private static String COL_UNIDADE = "K";
58     private static String codForm = "";
59     private static String apelidoGrupo = "";
60
61     private String configPath = Funcoes.getWRootDir() + "/upload/cadastros/";
62
63     @Execution
64     public void enviaEmail() {
65         try {
66             Map<String,String> parametros = Funcoes.getParametrosIntegracao(configPath + "config/RoboMontaRelatorio");
67
68             codForm = parametros.get("codForm");
69             apelidoGrupo = parametros.get("apelidoGrupo");
70
71             logger.debug("codForm: " + codForm);
72
73         } catch (Exception e) {
74             // TODO: handle exception
75             logger.error("Erro ao achar properties", e);
76         }
77
78         try {
79             logger.debug("Inicio do Robo -> Verificando se é dia de Enviar Relatório");
80
81             Calendar calendar = new GregorianCalendar();
82             LocalDate dataAtual = LocalDate.now();
83             int hora = calendar.get(Calendar.HOUR_OF_DAY);
84             logger.debug(hora);
85
86             // PEGA A O DIA DO MES ATUAL E COMPARA PARA VER SE É O DIA PRIMEIRO
87             if ((dataAtual.getDayOfMonth() == 1 || dataAtual.getDayOfMonth() == 16) && hora == 6) {
88                 logger.debug("hoje é dia de Relatório --> data: " + dataAtual + " Hora: " + hora);
89                 Date dataGravacao = new Date();
90                 SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
91                 SimpleDateFormat farquivo = new SimpleDateFormat("yyyy-MM-dd");
92                 // usa a função getPast, para pegar a pasta de um determinado caminho especificado na função
93                 String caminhoArquivo = getPast();
94                 String nomeArquivo = "Relatorio_" + farquivo.format(dataGravacao) + ".xlsx";
95
96                 logger.debug("Inicio escrevePlanilha ");
97                 String caminhoPlanilha = configPath + "excelDefault/Relatorio_Exemplo.xlsx";
98                 // busca template criado para este projeto
99                 FileInputStream inputFile = new FileInputStream(new File(caminhoPlanilha));
100                 // workbook classe da api do apache para se trabalhar com arquivo excel
101                 Workbook workbook = new XSSFWorkbook(); // XLSX
102                 // busca o template do excel para digitalizar seu "estilo"
103                 workbook = WorkbookFactory.create(inputFile);
104                 Sheet sheet = workbook.getSheetAt(0);
105
106                 // REFERENCIA DO LOCAL DAS CELULAS
107                 Cell referenceValorCell = getCell(sheet, getCellReference("A3"));
108
109                 Cell cellCodProcesso = getCell(sheet, getCellReference("A3"));
110                 Cell cellDTAbertura = getCell(sheet, getCellReference("B3"));
111                 Cell cellEtapa = getCell(sheet, getCellReference("C3"));
112                 Cell cellStatusProcesso = getCell(sheet, getCellReference("D3"));
113                 Cell cellNomSolic = getCell(sheet, getCellReference("E3"));
114                 Cell cellNome = getCell(sheet, getCellReference("F3"));
115                 Cell cellTipo = getCell(sheet, getCellReference("G3"));
116                 Cell cellPrevisao = getCell(sheet, getCellReference("H3"));
117                 Cell cellMaterial = getCell(sheet, getCellReference("I3"));
118                 Cell cellQuantidade = getCell(sheet, getCellReference("J3"));
119                 Cell cellUnidade = getCell(sheet, getCellReference("K3"));
120
121                 // DESIGN DAS CELULAS
122                 CellStyle estiloValor = referenceValorCell.getCellStyle();
123
124                 try {

```

```

124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168

StringBuilder sql = new StringBuilder();

sql.append(" SELECT DISTINCT pe.COD_PROCESSO, pe.IDE_STATUS, e.TITULO_ETAPA, sol.NOM_USUARIO AS SOLICITANTE, u.NOM_USUARIO AS EXECUTOR_ETAPA, ");
sql.append("     date.format(p.DAT_DATA, '%d/%m/%Y %h:%i') as DATA_ABERTURA, ");
sql.append("     fa.NOME, fa.TIPO_SOLICITACAO, DATE_FORMAT(TD.DT_PREVISAO, '%d/%m/%Y') as datPrevisao, ");
sql.append("     gs.MATERIAL, gs.QTD, gs.UNIDADE_MEDIDA, ");
sql.append("     FROM processo_etapa pe ");
sql.append(" INNER JOIN usuario u ON pe.COD_USUARIO_ETAPA = u.COD_USUARIO ");
sql.append(" INNER JOIN processo p ON pe.COD_PROCESSO = p.COD_PROCESSO AND p.COD_ETAPA_ATUAL = pe.COD_ETAPA_ATUAL AND p.COD_CICLO_ATUAL = pe.COD_CICLO ");
sql.append(" INNER JOIN etapa e ON e.COD_ETAPA = p.COD_ETAPA_ATUAL AND e.COD_FORM = p.COD_FORM AND e.COD_VERSAO = p.COD_VERSAO ");
sql.append(" INNER JOIN f_lic_m_comp fa ON p.COD_PROCESSO = fa.COD_PROCESSO_F AND fa.COD_CICLO_F = p.COD_CICLO_ATUAL AND p.COD_ETAPA_ATUAL = fa.COD_ETAPA_F ");
sql.append(" LEFT JOIN g_lic_m_comp MATERIAL gs ON p.COD_PROCESSO = gs.COD_PROCESSO AND gs.COD_CICLO = p.COD_CICLO_ATUAL AND p.COD_ETAPA_ATUAL = gs.COD_ETAPA ");
sql.append(" WHERE p.COD_FORM = ? ");
sql.append(" ORDER BY pe.COD_PROCESSO ");

//logger.debug(sql.toString());
try (Connection con = DBUtils.getConnection("workflow")) {
    //realiza a conexão com o banco
    try (PreparedStatement pst = con.prepareStatement(sql.toString())) {
        pst.setString(1, codForm);

        StringBuilder mensagem = new StringBuilder();
        mensagem.append("<div class='col-lg-6'>");
        mensagem.append("<h4>Segue em Anexo o Relatório.</h4></div>");
        mensagem.append("<div class='col-lg-6'>");
        mensagem.append("<h4>Data Atual: " + formato.format(datGravacao) + "</h4></div>");

        try (ResultSet rs = pst.executeQuery()) {
            //Passa o resultado de cada linha e manda para o select abaixo
            Sheet service = workbook.cloneSheet(0); // esta função clona a aba , ou seja todo seu template e até o nome da aba fica (nome da aba_1 ou nome da aba_1_1)
            int linha = 3;
            while (rs.next()) {
                String codProcesso = Funcoes.nulo(rs.getString("COD_PROCESSO"), "");
                String status = Funcoes.nulo(rs.getString("IDE_STATUS"), "");
                String nomesolic = Funcoes.nulo(rs.getString("SOLICITANTE"), "");
                String tituloetapa = Funcoes.nulo(rs.getString("TITULO_ETAPA"), "");
                String datainicial = Funcoes.nulo(rs.getString("DATA_ABERTURA"), "");
                String nome = Funcoes.nulo(rs.getString("NOME"), "");
                String tiposolicitacao = Funcoes.nulo(rs.getString("TIPO_SOLICITACAO"), "");
                String dataPrevisao = Funcoes.nulo(rs.getString("datPrevisao"), "");
                String material = Funcoes.nulo(rs.getString("MATERIAL"), "");
                String qtd = Funcoes.nulo(rs.getString("QTD"), "");
                String unidade = Funcoes.nulo(rs.getString("UNIDADE_MEDIDA"), "");
            }
        }
    }
}

```

```

170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213

if(status.equals("P")) {
    status = "Aprovado";
} else if(status.equals("C")) {
    status = "Cancelado";
} else if(status.equals("R")) {
    status = "Rejeitado";
} else if(status.equals("A")) {
    status = "Andamento";
}

cellCodProcesso = getCell(service, getCellReference(COL_PROCESSO + linha));
cellCodProcesso.setCellValue(codProcesso);
cellCodProcesso.setCellStyle(estiloValor);

cellDTAbertura = getCell(service, getCellReference(COL_ABERTURA + linha));
cellDTAbertura.setCellValue(datainicial);
cellDTAbertura.setCellStyle(estiloValor);

cellStatusProcesso = getCell(service, getCellReference(COL_STATUS + linha));
cellStatusProcesso.setCellValue(status);
cellStatusProcesso.setCellStyle(estiloValor);

cellEtapa = getCell(service, getCellReference(COL_ETAPA + linha));
cellEtapa.setCellValue(tituloEtapa);
cellEtapa.setCellStyle(estiloValor);

cellNomSolic = getCell(service, getCellReference(COL_SOLIC + linha));
cellNomSolic.setCellValue(nomesolic);
cellNomSolic.setCellStyle(estiloValor);

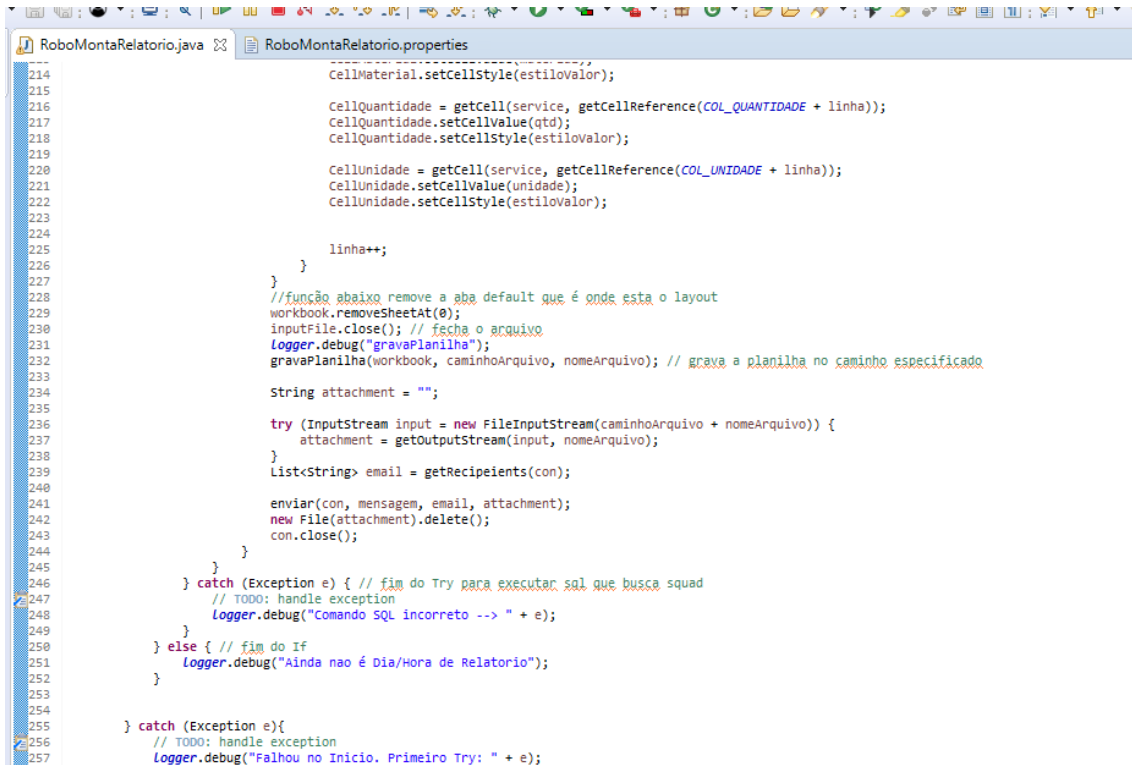
cellNome = getCell(service, getCellReference(COL_NOME + linha));
cellNome.setCellValue(nome);
cellNome.setCellStyle(estiloValor);

cellTipo = getCell(service, getCellReference(COL_TIPO + linha));
cellTipo.setCellValue(tipoSolicitacao);
cellTipo.setCellStyle(estiloValor);

cellPrevisao = getCell(service, getCellReference(COL_PREVISAO + linha));
cellPrevisao.setCellValue(dataPrevisao);
cellPrevisao.setCellStyle(estiloValor);

cellMaterial = getCell(service, getCellReference(COL_MATERIA + linha));
cellMaterial.setCellValue(material);

```



```
214 cellMaterial.setCellStyle(estiloValor);
215
216 cellQuantidade = getCell(service, getCellReference(COL_QUANTIDADE + linha));
217 cellQuantidade.setCellValue(qtd);
218 cellQuantidade.setCellStyle(estiloValor);
219
220 cellUnidade = getCell(service, getCellReference(COL_UNIDADE + linha));
221 cellUnidade.setCellValue(unidade);
222 cellUnidade.setCellStyle(estiloValor);
223
224 linha++;
225
226 }
227 }
228 //função abaixo remove a aba default que é onde está o layout
229 workbook.removeSheetAt(0);
230 inputFile.close(); // fecha o arquivo
231 logger.debug("gravaPlanilha");
232 gravaPlanilha(workbook, caminhoArquivo, nomeArquivo); // grava a planilha no caminho especificado
233
234 String attachment = "";
235
236 try (InputStream input = new FileInputStream(caminhoArquivo + nomeArquivo)) {
237     attachment = getOutputStream(input, nomeArquivo);
238 }
239 List<String> email = getRecipeints(con);
240
241 enviar(con, mensagem, email, attachment);
242 new File(attachment).delete();
243 con.close();
244
245 }
246 } catch (Exception e) { // fim do Try para executar sql que busca squad
247     // TODO: handle exception
248     logger.debug("Comando SQL incorreto --> " + e);
249 }
250 } else { // fim do If
251     logger.debug("Ainda nao é Dia/Hora de Relatorio");
252 }
253
254 } catch (Exception e){
255     // TODO: handle exception
256     logger.debug("Falhou no Inicio. Primeiro Try: " + e);
257 }
```

Nas imagens acima contém o método principal da execução desse robô, dele é chamado alguns outros que iremos ver na sequência após a explicação deste.

Nas **linhas 47 a 59** são criadas variáveis estáticas apenas para utilização dentro desse robô, variáveis essas que são as colunas do excel, para facilitar na hora de preencher a informação, e os campos dos properties que serão utilizados também no decorrer do arquivo.

Na **linha 66** conforme já vistos em outros documentos está sendo feita a transformação do arquivo properties em um Map para facilmente pegar os valores que deseja e assim nas **linhas 68 a 69** atribuímos os valores as variáveis estáticas.

Nas **linhas 81 a 86** estamos pegando a data e hora atual e validando se está no dia e hora de execução do robô, no caso está sendo validado se é dia 1 ou 16 e se a hora é 6, caso entre na condição fará a criação do excel e envio do e-mail.

Na **linha 92** está sendo retornado o caminho onde irá salvar temporariamente o excel gerado com os dados para após o envio do e-mail ele ser deletado, a **linha 93** está denominando um nome para esse arquivo final.

Nas **linhas 96 a 103** está sendo feito a inicialização da leitura do arquivo excel de modelo, utilizando as classes da lib poi para realizar essas iterações com criações de excel, por isso instanciamos o objeto Workbook e dele estamos fazendo um create (**linha 102**) a partir do modelo existente, e assim pegamos o objeto Sheet, a aba inicial do nosso excel e a partir dela que faremos o preenchimento do excel com os valores.

Nas **linhas 106 a 118** realizamos a inicialização das colunas, pegando a referência das que já existe no excel de modelo, dentro do while vamos ver que utilizaremos esses mesmos nomes para preencher os dados das colunas respectivas.

Na **linha 121** pegamos o estilo da primeira coluna para que ele mantenha esse estilo para todas colunas linhas que forem sendo preenchidas.

Nas **linhas 125 a 139** está sendo montado um select na tabela de processo_etapa, realizando um inner join com as tabelas de processo para pegar os processos do formulário que passarmos, tabela de usuário para pegar o nome do usuário iniciador, tabela de etapa para pegar o nome da atividade que está o processo, tabela do modelo em questão para trazer os dados específicos desse processo e realiza um left join com a tabela da grid específica do modelo para que assim vai trazer repetidamente as informações do processo pela quantidade de vezes que teve registro na tabela da grid.

Na **linha 145** é onde estamos informando qual parâmetro será usado no select, nesse caso só utilizamos o código de formulário como parâmetro.

Nas **linhas 147 a 151** estamos iniciando o conteúdo do e-mail que será enviado.

Na **linha 155** está sendo clonada a aba inicial apenas para conseguirmos escrever dentro dessa aba e caso desejasse criar outra aba dentro do documento.

Na **linha 156** iniciamos a variável linha que será a partir de qual linha irá começar a escrever no excel e irá incrementando essa variável na **linha 225** para ir escrevendo até o fim do retorno do select.

Nas **linhas 158 a 168** estamos atribuindo a variáveis os valores que vieram do retorno do select, pegando a partir do resultset e estamos validando utilizando a função **nulo** da classe **Funcoes** para verificar se vier nulo ele preenche com o valor em branco.

Nas **linhas 170 a 178** estamos validando qual status que veio para colocar o nome de fato.

Nas **linhas abaixo** o que temos é um conjunto de 3 linhas para cada coluna que vamos preencher onde eu digo em qual aba vamos escrever, e em qual coluna vamos escrever utilizando ai a variável estática e a linha que está sendo incrementada e assim nessa célula informamos o value e o estilo para assim manter um padrão em todas as linhas colunas.

Na **linha 230** está sendo feito o fechamento do arquivo para não ficar em memória o objeto que foi instanciando e não causando possíveis erros no sistema.

Na **linha 232** estamos chamando a função que fará a gravação desse workbook em um arquivo físico que definimos.

Nas **linhas 236 a 238** está sendo feito a leitura e escrita desse arquivo físico utilizando as classes do java de inputstream e outputstream para assim ser retornado o caminho absoluto do arquivo gerado e esse ser utilizado.

Na **linha 239** estamos chamando o método que nos retornará todos os e-mails dos usuários de um determinado grupo, grupo esse definido no properties.

Na **linha 241** estamos chamando o método que fará o envio do e-mail.

Nas **linhas 242 e 243** está sendo feito a deleção do arquivo gerado para não ficar com o arquivo no servidor e sim somente ser temporário até o envio do e-mail, e abaixo está sendo fechada a conexão com o banco de dados.

No restante do método principal são tratativas para caso de algum problema e ele loga no arquivo gerado para esse robô que estamos fazendo, sendo possível acessar esse log direto pela plataforma em Studio -> Logs do Ambiente.

```

RoboMontaRelatorio.java  RoboMontaRelatorio.properties
261
262
263 private void enviar(Connection con, StringBuilder mensagem, List<String> email, String anexo) {
264     String from = Funcoes.getFrom(con, Logger);
265
266     String subject = "Robô - Relatório Solicitações";
267
268     Logger.debug("entrou no enviar, até aqui ok");
269     EmailMessage enviaEmailMessage = new EmailMessage(subject, mensagem.toString(), from, email, true);
270     enviaEmailMessage.setAttached(anexo);
271     WfMail wfmail = new WfMail();
272     wfmail.enviaEmailMessage(enviaEmailMessage);
273     Logger.debug("[[[[ RbNotificarAtrasos - Inicio ENVIO: ]]]]");
274     Logger.debug("Msg: " + mensagem.toString());
275     Logger.debug("Subject: " + subject);
276     Logger.debug("Recipients: " + email);
277     Logger.debug("[[[[ RbNotificarAtrasos - Fim ENVIO: ]]]]");
278 }
279
280
281 private String getPasta() throws Exception {
282     String retorno = Funcoes.getWfRootDir() + "/upload/cadastros/planilha/";
283     File pasta = new File(retorno);
284
285     // Verifica se a pasta existe
286     if (!pasta.exists()) {
287         pasta.mkdirs();
288
289         // Cria o arquivo index.html para segurança da pasta no workflow
290         File index = new File(retorno + "index.html");
291         if (!index.exists()) {
292             index.createNewFile();
293         }
294     }
295     return retorno;
296 }

```

Continuando na classe RoboMontaRelatorio temos os métodos que dão subsídio para a execução do método principal, acima temos os seguintes métodos **enviar** e **getPasta**.

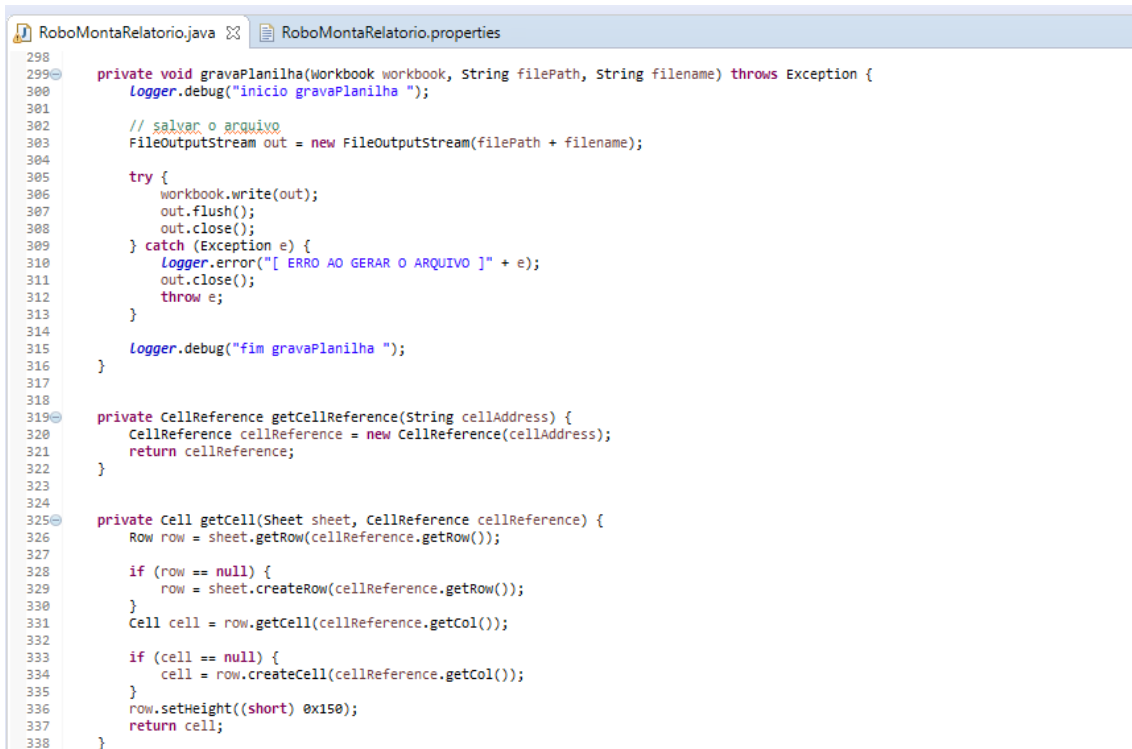
Na **linha 264** início do método **enviar** que recebe a conexão do banco de dados, o conteúdo do e-mail a ser enviado, a lista de e-mails dos usuários que receberam e o anexo que será encaminhado, estamos utilizando a chamada do método **getFrom** da classe útil **Funcoes** e ele nos retornará o e-mail cadastrado no admin como sendo o e-mail from de todos os envios que a plataforma fizer.

Nas **linhas 269 a 270** estamos utilizando a classe **EmailMessage** que o produto libera para uso nosso de envio de e-mail, então no construtor passamos o assunto do e-mail, conteúdo do e-mail, o e-mail from, a lista de e-mail que receberão, e o último parâmetro é se no momento do envio ele transformará o conteúdo em um html de verdade e na **linha 270** passamos o caminho do arquivo gerado para o método **setAttached**, para ser adicionado como um anexo do e-mail.

Nas **linhas 271 e 272** temos que instanciar outra classe que produto libera para enviar o e-mail de fato, como vimos em outro documento na integração conseguimos apenas chamar um método e ele faz o envio, dentro dos robôs precisamos instanciar essa classe **WfMail** e chamamos o método **enviaEmailMessage** passando o objeto que instanciamos e preenchemos nas linhas acima e assim o envio do e-mail é feito com sucesso.

Na **linha 282** é o início do método **getPasta** onde nele é chamado o método **getWfRootDir** da classe útil **Funcoes**, que retorna a raiz do bpm (/opt/lecom/app/tomcat/webapps/bpm) e concatena com o caminho que usamos como padrão upload/cadastros/planilha, a pasta planilha é apenas para guardar ali temporariamente os arquivos gerados.

Nas **linhas 286 a 294** está sendo feito utilizando a classe **File** do Java, e é verificado se existe a pasta se não existir ele cria, e aproveita para criar um arquivo vazio somente para garantir que consegue criar arquivos nesse diretório.



```

298
299 private void gravaPlanilha(Workbook workbook, String filePath, String filename) throws Exception {
300     Logger.debug("inicio gravaPlanilha ");
301
302     // salvar o arquivo
303     FileOutputStream out = new FileOutputStream(filePath + filename);
304
305     try {
306         workbook.write(out);
307         out.flush();
308         out.close();
309     } catch (Exception e) {
310         Logger.error("[ ERRO AO GERAR O ARQUIVO ]" + e);
311         out.close();
312         throw e;
313     }
314
315     Logger.debug("fim gravaPlanilha ");
316 }
317
318
319 private CellReference getCellReference(String cellAddress) {
320     CellReference cellReference = new CellReference(cellAddress);
321     return cellReference;
322 }
323
324
325 private Cell getCell(Sheet sheet, CellReference cellReference) {
326     Row row = sheet.getRow(cellReference.getRow());
327
328     if (row == null) {
329         row = sheet.createRow(cellReference.getRow());
330     }
331     Cell cell = row.getCell(cellReference.getCol());
332
333     if (cell == null) {
334         cell = row.createCell(cellReference.getCol());
335     }
336     row.setHeight((short) 0x150);
337     return cell;
338 }

```

Na imagem acima temos 3 métodos, **gravaPlanilha**, **getCellReference**, **getCell**.

Nas **linhas 303 a 315** estamos realizando a inicialização da classe **FileOutputStream** passando o caminho de onde será gerado o arquivo temporário, e a partir do objeto **workbook** é feito a escrita nesse objeto de saída, e depois fechado a variável da classe **FileOutputStream** para não ficar aberto em memória.

Nas **linhas 320 e 321** está realizando apenas um retorno da célula de referência, a partir da coluna que passamos ele retorna a célula.

Nas **linhas 326 a 338** estamos utilizando as classes **Row** e **Cell** da lib poi para retornarmos qual linha e qual célula estamos para realizar o preenchimento do valor da coluna corretamente. Por isso na linha 326 pegamos o **Row**, linha do excel, e se ela não existir é feito a criação dessa linha, e a mesma coisa para **Cell**, verifica se já existe se não cria e retorna a célula para ser feito o preenchimento do valor.


```

RoboMontaRelatorio.java RoboMontaRelatorio.properties
340
341 private String getOutputStream(InputStream input, String nomeAnexo) throws Exception {
342
343     try {
344
345         if (input != null && !input.equals("")) {
346             // Cria arquivos de anexo temporários
347             try (OutputStream output = new FileOutputStream(new File(nomeAnexo))); {
348                 int read = 0;
349                 byte[] bytes = new byte[1024];
350
351                 while ((read = input.read(bytes, 0, bytes.length)) > 0) {
352                     output.write(bytes, 0, read);
353                     output.flush();
354                 }
355             }
356         }
357     } catch (Exception e) {
358         e.printStackTrace();
359         Logger.error("ERRO: " + e);
360     }
361     return nomeAnexo;
362 }
363
364

```

Na imagem acima temos o método **getOutputStream**, que escreverá no arquivo a partir do **InputStream** enviado.

```

RoboMontaRelatorio.java RoboMontaRelatorio.properties
362
363 }
364
365
366 private List<String> getRecipeients(Connection con) {
367
368     List<String> recipients = new ArrayList<String>();
369
370     StringBuilder sql = new StringBuilder();
371     sql.append(" SELECT u.DES_EMAIL ");
372     sql.append(" FROM GRUPO g ");
373     sql.append(" INNER JOIN GRUPO_USUARIO gu ON gu.COD_GRUPO = g.COD_GRUPO ");
374     sql.append(" INNER JOIN USUARIO u ON u.COD_USUARIO = gu.COD_USUARIO ");
375     sql.append(" WHERE g.DES_COMANDO = ' '");
376     sql.append(apelidoGrupo);
377     sql.append("");
378
379     try(PreparedStatement pst = con.prepareStatement(sql.toString());){
380         try(ResultSet rs = pst.executeQuery();){
381             while(rs.next()){
382                 recipients.add(rs.getString("DES_EMAIL"));
383             }
384         }
385     } catch (Exception e) {
386         // TODO: handle exception
387         Logger.error("erro ao buscar recipients");
388     }
389
390     return recipients;
391 }
392
393 @Override
394 public List<WebServicesVO> getWebServices() {
395     // TODO Auto-generated method stub
396     return null;
397 }
398
399 @Override
400 public void setWebServices(WebServicesVO arg0) {
401     // TODO Auto-generated method stub
402
403 }
404
405 }

```

Na imagem acima temos o método **getRecipeients**, que retornará todos os e-mails dos usuários pertencentes ao grupo configurado.

Nas **linhas 370 a 377** estamos realizando o select na tabela de grupo realizando join com tabela de grupo_usuario e usuário, assim a partir do apelido do grupo retornamos todos os e-mails de todos os usuários presente naquele grupo.

5. Resultado

Após toda a explicação do que é feito nesse exemplo o que temos como resultado é um e-mail muito simples apenas com um texto conforme abaixo:

Assunto: Robô - Relatório Solicitações
Para: Mim ★

Segue em Anexo o Relatório.

Data Atual: 12/05/2020.

> 1 anexo: Relatório_2020_05_12.xlsx 7,2KB

O excel modelo contém o seguinte formato:

A	B	C	D	E	F	G	H	I	J	K	L
Processo	Data Abertura	Etapas Atual	Status	Solicitante	Nome	Tipo de Solicitação	Previsão da Necessidade	Material	Quantidade	Unidade de Medida	

O excel em anexo do email veio com o conteúdo preenchido corretamente nas linhas corretas.

A	B	C	D	E	F	G	H	I	J	K	L
Processo	Data Abertura	Etapas Atual	Status	Solicitante	Nome	Tipo de Solicitação	Previsão da Necessidade	Material	Quantidade	Unidade de Medida	
2119	06/01/2020 12:17	Efetuar Compras	Aprovado	Lecom - Marcos Santos	Marcos Santos	Material de Informática		Item 01	9		
2119	06/01/2020 12:17	Efetuar Compras	Aprovado	Lecom - Marcos Santos	Marcos Santos	Material de Informática		Item 02	20		
2373	09/01/2020 15:05	Solicitar Reembolso	Cancelado	Lecom - Marcos Santos							
4110	08/05/2020 13:31	Fim	Andamento	Administrador	Teste USER	Material de Informática	14/05/2020	material 1	2	Pacote	
4110	08/05/2020 13:31	Fim	Andamento	Administrador	Teste USER	Material de Informática	14/05/2020	material 2	3	Unidade	
4110	08/05/2020 13:31	Fim	Andamento	Administrador	Teste USER	Material de Informática	14/05/2020	material 3	4	Litros	
4110	08/05/2020 13:31	Fim	Andamento	Administrador	Teste USER	Material de Informática	14/05/2020	material 4	5	Outro	