**function** *generateAssetGatePassReport*(singleChallanId) {  
 **if** (*$*(**"#organization"**).val()) {  
 **var** organization = *$*(**"#organization option:selected"**).val();  
 }  
 **if** (*$*(**"#office"**).val()) {  
 **var** office = *$*(**"#office option:selected"**).val();  
 }  
 **if** (*$*(**"#project"**).val()) {  
 **var** project = *$*(**"#project option:selected"**).val();  
 }  
 **if** (*$*(**"#item"**).val()) {  
 **var** item = *$*(**"#item option:selected"**).val();  
 }  
  
 **if** (*$*(**"#transferDate"**).val()) {  
 **var** transferDate = *$*(**"#transferDate"**).val();  
 }  
 **if** (*$*(**"#branchId"**).val()) {  
 **var** organizationBranchId = *$*(**"#branchId"**).val()  
 }  
 **var** data = **""**;  
 **if** (organizationBranchId) {  
 data = data + **"&organizationBranchId="** + organizationBranchId  
 }  
 **if** (organization) {  
 data = data + **"&organizationId="** + organization  
 }  
 **if** (office) {  
 data = data + **"&office="** + office  
 }  
 **if** (project) {  
 data = data + **"&project="** + project  
 }  
 data = data + **"&challan="** + singleChallanId;  
 **if** (transferDate) {  
 data = data + **"&transferDate="** + transferDate  
 }  
 **if** (item) {  
 data = data + **"&item="** + item  
 }  
 **var** singleChallan = 1;  
 data = data + **'&shipmentChallanOrSingleChallan='** + singleChallan;  
 **var** url = **"${**createLink(**controller**: **'faAssetReport'**, **action**: **'generateAssetGatePassReport'**)**}?format=PDF&radDocx=2"** + data;  
 ***window***.open(url);  
}

**def generateAssetGatePassReport** = {  
 ByteArrayOutputStream pdfStream = **null**;  
 Map assetRegisterReport = (Map) assetGatePassReportActionService.execute(params)  
 pdfStream = assetRegisterReport.**pdfStream** response.setHeader(**"Content-Disposition"**, **"inline;filename=AssetGatePass\_**${DateUtil.*now*()}**.docx"**);  
 render(**file**: pdfStream.toByteArray(), **contentType**: **'application/msword'**)  
}

**package** com.bits.gerp.fixedasset.action.report  
  
**import** com.bits.gerp.admin.BaseService  
**import** com.docu.security.ApplicationUser  
**import** com.docu.commons.Organization  
**import** com.bits.gerp.common.ActionInterface  
**import** com.bits.gerp.utility.DateUtility  
**import** grails.util.Holders  
**import** org.codehaus.groovy.grails.web.servlet.mvc.GrailsParameterMap  
**import** net.sf.jasperreports.engine.JasperFillManager  
**import** net.sf.jasperreports.engine.JasperPrint  
**import** net.sf.jasperreports.engine.export.JRPdfExporter  
**import** net.sf.jasperreports.engine.export.JRXlsExporter  
**import** net.sf.jasperreports.engine.export.ooxml.JRDocxExporter  
**import** net.sf.jasperreports.export.SimpleDocxExporterConfiguration  
**import** net.sf.jasperreports.export.SimpleExporterInput  
**import** net.sf.jasperreports.export.SimpleOutputStreamExporterOutput  
**import** net.sf.jasperreports.export.SimplePdfExporterConfiguration  
**import** net.sf.jasperreports.export.SimpleXlsExporterConfiguration  
**import** net.sf.jasperreports.export.SimpleXlsReportConfiguration  
**import** org.apache.log4j.Logger  
**import** org.springframework.transaction.annotation.Transactional  
  
  
**class** AssetGatePassReportActionService **extends** BaseService **implements** ActionInterface {  
  
 **private** Logger **log** = Logger.*getLogger*(getClass())  
 **private static final** String ***PDF\_STREAM*** = **"pdfStream"** @Override  
 Map executePreCondition(Map parameters) {  
 **return** parameters  
 }  
  
 @Transactional  
 Map execute(Map previousResult) {  
 String transferDate = **''**, toDate\_1, jrxmlFileName, subReportFilePath, dotJasperFileName  
 Long office, project, organizationId, challan, asset  
 ByteArrayOutputStream pdfStream = **null**;  
 Map<String, Objects> reportParam = **new** HashMap<String, Objects>()  
  
 **try** {  
 GrailsParameterMap parameterMap = (GrailsParameterMap) previousResult  
 Organization organization = appSessionManagementService.getOrganization();  
 ApplicationUser applicationUser = appSessionManagementService.getApplicationUser()  
  
 String condition = **""  
  
 if** (parameterMap.**shipmentChallanOrSingleChallan**.equals(**'0'**)) {  
 jrxmlFileName = **"rptAssetGatePass"** } **else if** (parameterMap.**shipmentChallanOrSingleChallan**.equals(**'1'**)) {  
  
 jrxmlFileName = **"rptAssetSingleChallanGatePass"** }  
  
 **if** (parameterMap.**organizationId**) {  
 organizationId = Long.*parseLong*(parameterMap.**organizationId**)  
 }  
 **if** (parameterMap.**office**) {  
 office = Long.*parseLong*(parameterMap.**office**)  
 }  
 **if** (parameterMap.**project**) {  
 project = Long.*parseLong*(parameterMap.**project**)  
 }  
 **if** (parameterMap.**item**) {  
 asset = Long.*parseLong*(parameterMap.**item**)  
 }  
 **if** (parameterMap.**challan**) {  
 challan = Long.*parseLong*(parameterMap.**challan**)  
 }  
 **if** (parameterMap.**transferDate**) {  
 transferDate = DateUtility.*getDateFromString*(parameterMap.**transferDate**)  
 }  
  
 subReportFilePath = Holders.*grailsApplication*.mainContext.getResource(**'reports/common/'**).file.getAbsoluteFile()  
 *//Report Path* dotJasperFileName = Holders.*grailsApplication*.mainContext.getResource(**'reports/fams/'** + jrxmlFileName + **'.jasper'**).file.getAbsoluteFile()  
  
 *// Report parameter* JasperPrint print;  
  
 reportParam.put(**"organizationId"**, organizationId)  
*// reportParam.put("office", office)* reportParam.put(**"project"**, project)  
 reportParam.put(**"challan"**, challan)  
 reportParam.put(**"transferDate"**, transferDate)  
 reportParam.put(**"asset"**, asset)  
 reportParam.put(**"SUBREPORT\_DIR"**, subReportFilePath + **"\\"**)  
 reportParam.put(**"organization"**, organization.id)  
 reportParam.put(**"organizationBranch"**, Long.*parseLong*(parameterMap.**organizationBranchId**))  
 reportParam.put(**"subReportOrgId"**, organization.id)  
 String baseURL = *grailsLinkGenerator*.serverBaseURL  
 String defaultImagePath = baseURL + **"/images/brac\_logo.jpg"** reportParam.put(**"defaultImagePath"**, defaultImagePath)  
  
 print = JasperFillManager.*fillReport*(dotJasperFileName, reportParam, dataSource.getConnection())  
  
  
 pdfStream = **new** ByteArrayOutputStream();  
  
 **if** ((parameterMap.**radPdf**).equals(**"1"**)) {*// exports report to pdf* JRPdfExporter exporter = **new** JRPdfExporter();  
 exporter.setExporterInput(**new** SimpleExporterInput(print));  
 exporter.setExporterOutput(**new** SimpleOutputStreamExporterOutput(pdfStream));  
 SimplePdfExporterConfiguration configuration = **new** SimplePdfExporterConfiguration();  
 exporter.setConfiguration(configuration);  
 exporter.exportReport();  
 }  
  
 **if** ((parameterMap.**radXls**).equals(**"0"**)) {*// exports report to excel* JRXlsExporter xlsExporter = **new** JRXlsExporter();  
 xlsExporter.setExporterInput(**new** SimpleExporterInput(print));  
 xlsExporter.setExporterOutput(**new** SimpleOutputStreamExporterOutput(pdfStream));  
 SimpleXlsReportConfiguration xlsReportConfiguration = **new** SimpleXlsReportConfiguration();  
 SimpleXlsExporterConfiguration xlsExporterConfiguration = **new** SimpleXlsExporterConfiguration();  
 xlsReportConfiguration.setOnePagePerSheet(**false**);  
 xlsReportConfiguration.setRemoveEmptySpaceBetweenRows(**true**);  
 xlsReportConfiguration.setDetectCellType(**true**);  
 xlsReportConfiguration.setWhitePageBackground(**false**);  
 xlsExporter.setConfiguration(xlsReportConfiguration);  
 xlsExporter.exportReport();  
 }  
 **if**(((parameterMap.**radDocx**).equals(**"2"**))){  
 JRDocxExporter exporter = **new** JRDocxExporter();  
  
 exporter.setExporterInput(**new** SimpleExporterInput(print));  
 exporter.setExporterOutput(**new** SimpleOutputStreamExporterOutput(pdfStream));  
 SimpleDocxExporterConfiguration configuration = **new** SimpleDocxExporterConfiguration();  
 exporter.setConfiguration(configuration);  
 exporter.exportReport();  
 }  
 previousResult.put(***PDF\_STREAM***, pdfStream)  
  
 } **catch** (Exception ex) {  
 **log**.error(ex.getMessage())  
 **throw new** RuntimeException(ex)  
 }  
 **finally** {  
 pdfStream.close();  
 }  
  
 **return** previousResult  
 }  
  
 @Override  
 Map executePostCondition(Map previousResult) {  
 **return** previousResult  
 }  
  
 @Override  
 Map buildSuccessResult(Map executeResult) {  
 **return** executeResult  
 }  
  
 @Override  
 Map buildFailureResult(Map executeResult) {  
 **return** executeResult  
 }  
}