Bonjour,

Je vous joints à travers ce document le code ajouté dans le projet afin de faire répondre à l’énoncé que vous nous avez transmis.

Tout est fonctionnel (le PDF de l’article aussi).

En parallèle je vous joins le lien du fork GitHub du projet (vous pourrez certainement plus facilement vous y retrouvez grâce au commit et les fichiers modifiés car dans un Word ça me parait illisible) et un zip du projet en pièce jointe dans le courriel, car il y a eu pas mal de modification notamment quelques-unes dans les DTO (j’ai souhaité les utiliser dans la mesure du possible) et d’ajout.

Par peur d’oublié de vous transmettre une partie de code ajouté je préfère vous partager le projet complet.

[lien GitHub](https://github.com/Fabrice-F/21-DIGIT-JAVA250/tree/live-coding-cdev)

ExportArticleController.

package com.example.demo.controller.export;  
  
import com.example.demo.service.export.ArticleExportCVSService;  
import com.example.demo.service.export.ArticleExportXLSXService;  
import com.example.demo.service.export.ClientExportCVSService;  
import com.example.demo.service.export.ExportPDFITextService;  
import com.itextpdf.text.Document;  
import com.itextpdf.text.DocumentException;  
import com.itextpdf.text.Paragraph;  
import com.itextpdf.text.pdf.PdfWriter;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
import org.apache.poi.ss.usermodel.Workbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
  
*/\*\*  
 \* Controller pour réaliser export des articles.  
 \*/*@Controller  
@RequestMapping("export")  
public class ExportArticleController {  
  
 @Autowired  
 private ArticleExportCVSService articleExportCVSService;  
  
 @Autowired  
 private ArticleExportXLSXService articleExportXLSXService;  
  
 @Autowired  
 private ExportPDFITextService exportPDFITextService;  
  
  
 */\*\*  
 \* Export des articles au format CSV.  
 \*/* @GetMapping("/articles/csv")  
 public void articlesCSV(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 response.setContentType("text/csv");  
 response.setHeader("Content-Disposition", "attachment; filename=\"export-articles.csv\"");  
 PrintWriter writer = response.getWriter();  
 articleExportCVSService.export(writer);  
 }  
  
 @GetMapping("/articles/xlsx")  
 public void articlesXLSX(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 response.setContentType("application/vnd.ms-excel");  
 response.setHeader("Content-Disposition", "attachment; filename=\"export-articles.xlsx\"");  
 OutputStream outputStream = response.getOutputStream();  
 articleExportXLSXService.export(outputStream);  
 }  
  
 */\*\* Méthode utilisée pour l'export facture (point n°5) \*/* @GetMapping("/clients/{id}/factures/xlsx")  
 public void clientGetFacturesXLSX(@PathVariable Long id, HttpServletResponse response) throws IOException {  
 response.setContentType("application/vnd.ms-excel");  
 response.setHeader("Content-Disposition", "attachment; filename=\"client-" + id + "-factures.xlsx\"");  
 OutputStream outputStream = response.getOutputStream();  
 // *TODO* Workbook wb = new HSSFWorkbook();  
 wb.write(outputStream);  
 }  
  
 */\*\*  
 \* Export de la liste des articles au format PDF.  
 \*/* @GetMapping("/articles/pdf")  
 public void facturesPDF(HttpServletRequest request, HttpServletResponse response) throws IOException, DocumentException {  
 response.setContentType("application/pdf");  
 response.setHeader("Content-Disposition", "attachment; filename=\"test-factures.pdf\"");  
 OutputStream outputStream = response.getOutputStream();  
 exportPDFITextService.export(outputStream);  
 }  
  
}

ExportClientController

package com.example.demo.controller.export;  
  
import com.example.demo.service.export.ArticleExportCVSService;  
import com.example.demo.service.export.ArticleExportXLSXService;  
import com.example.demo.service.export.ClientExportCVSService;  
import com.example.demo.service.export.ClientExportXLSXService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
  
*/\*\*  
 \* Controller pour réaliser export des articles.  
 \*/*@Controller  
@RequestMapping("export")  
public class ExportClientController {  
  
 @Autowired  
 private ClientExportCVSService clientExportCVSService;  
 @Autowired  
 private ClientExportXLSXService clientExportXLSXService;  
  
 */\*\* Méthode utilisée pour l'export des clients en CSV \*/* @GetMapping("/clients/csv")  
 public void clientCSV(HttpServletResponse response) throws IOException {  
 response.setContentType("text/csv");  
 response.setHeader("Content-Disposition", "attachment; filename=\"export-clients.csv\"");  
 PrintWriter writer = response.getWriter();  
 clientExportCVSService.export(writer);  
 }  
  
 */\*\* Méthode utilisée pour l'export des clients en XLSX \*/* @GetMapping("/clients/xlsx")  
 public void clientXLSX(HttpServletResponse response) throws IOException {  
 response.setContentType("application/vnd.ms-excel");  
 response.setHeader("Content-Disposition", "attachment; filename=\"export-clients.xlsx\"");  
 OutputStream outputStream = response.getOutputStream();  
 clientExportXLSXService.export(outputStream);  
 }  
  
  
  
}

ClientDto :

package com.example.demo.dto;  
  
import java.time.LocalDate;  
import java.time.Period;  
import java.time.format.DateTimeFormatter;  
  
*/\*\*  
 \* Classe permettant d'exposer des données au format JSON au client.  
 \*/*public class ClientDto {  
 private Long id;  
 private String nom;  
 private String prenom;  
 private LocalDate dateNaissance;  
 private Integer age;  
  
 public ClientDto(Long id, String nom, String prenom, LocalDate date) {  
 this.id = id;  
 this.nom = nom;  
 this.prenom = prenom;  
 this.dateNaissance=date;  
 this.age = Period.*between*(date, LocalDate.*now*()).getYears();  
 }  
  
 public String getDateNaissance() {  
 DateTimeFormatter dateFormat = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");  
 return dateNaissance.format(dateFormat);  
 }  
  
 public void setDateNaissance(LocalDate dateNaissance) {  
 this.dateNaissance = dateNaissance;  
 }  
  
 public Integer getAge() {  
 return age;  
 }  
  
 public void setAge(Integer age) {  
 this.age = age;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getNom() {  
 return nom;  
 }  
  
 public void setNom(String nom) {  
 this.nom = nom;  
 }  
  
 public String getPrenom() {  
 return prenom;  
 }  
  
 public void setPrenom(String prenom) {  
 this.prenom = prenom;  
 }  
}

Article :

package com.example.demo.entity;  
  
import javax.persistence.\*;  
  
*/\*\*  
 \* Entity représentant un article.  
 \*/*@Entity  
public class Article {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 @Column  
 private String libelle;  
  
 @Column  
 private double prix;  
  
 @Column  
 private int stock;  
  
 @Column  
 private String Description;  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public void setDescription(String description) {  
 Description = description;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getLibelle() {  
 return libelle;  
 }  
  
 public void setLibelle(String libelle) {  
 this.libelle = libelle;  
 }  
  
 public double getPrix() {  
 return prix;  
 }  
  
 public void setPrix(double prix) {  
 this.prix = prix;  
 }  
  
 public int getStock() {  
 return stock;  
 }  
  
 public void setStock(int stock) {  
 this.stock = stock;  
 }  
}

Client

package com.example.demo.entity;  
  
import javax.persistence.\*;  
import java.time.LocalDate;  
import java.util.HashSet;  
import java.util.Set;  
  
*/\*\*  
 \* Entity représentant un client.  
 \*/*@Entity  
public class Client {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 @Column  
 private String nom;  
  
 @Column  
 private String prenom;  
  
 @OneToMany(mappedBy = "client")  
 private Set<Facture> factures = new HashSet<>();  
  
 @Column  
 private LocalDate dateNaisance;  
  
 public LocalDate getDateNaisance() {  
 return dateNaisance;  
 }  
  
 public void setDateNaisance(LocalDate dateNaisance) {  
 this.dateNaisance = dateNaisance;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getNom() {  
 return nom;  
 }  
  
 public void setNom(String nom) {  
 this.nom = nom;  
 }  
  
 public String getPrenom() {  
 return prenom;  
 }  
  
 public void setPrenom(String prenom) {  
 this.prenom = prenom;  
 }  
  
 public Set<Facture> getFactures() {  
 return factures;  
 }  
  
 public void setFactures(Set<Facture> factures) {  
 this.factures = factures;  
 }  
}

InitData

package com.example.demo.service;  
  
import com.example.demo.entity.Article;  
import com.example.demo.entity.Client;  
import com.example.demo.entity.Facture;  
import com.example.demo.entity.LigneFacture;  
import com.example.demo.repository.ClientRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.context.event.ApplicationReadyEvent;  
import org.springframework.context.ApplicationListener;  
import org.springframework.data.domain.Example;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import javax.persistence.EntityManager;  
import java.time.LocalDate;  
import java.util.HashSet;  
  
*/\*\*  
 \* Classe permettant d'insérer des données dans l'application.  
 \*/*@Service  
@Transactional  
public class InitData implements ApplicationListener<ApplicationReadyEvent> {  
  
 private EntityManager entityManager;  
  
 @Autowired  
 ClientRepository clientRepository;  
  
 public InitData(EntityManager entityManager) {  
 this.entityManager = entityManager;  
 }  
  
 @Override  
 public void onApplicationEvent(ApplicationReadyEvent applicationReadyEvent) {  
 insertTestData();  
 }  
  
 private void insertTestData() {  
 Article a1 = createArticle("Chargeurs de téléphones Portables", 22.98, 9,"Samsung EP-P1100. Type de chargeur: Intérieur;\n" + "Type de source d'alimentation: Secteur;\n" + "Compatibilité de chargeur: Smartphone;\n" + "Charge rapide. Couleur du produit: Noir");  
 Article a2 = createArticle("Playmobil Hydravion de Police", 14.39, 2,"L'intérieur de l'avion peut contenir deux personnages et une valise.");  
 Article a3 = createArticle("Distributeur de croquettes pour chien", 12.99, 0,"Distributeur de nourriture croquettes, biscuits ou snacks pour chats et chiens Plastique robuste avec couvercle amovible");  
  
 Client cl1 = createClient("John", "Doe",LocalDate.*parse*("1987-08-30"));  
 Client cl2 = createClient("Jeff", "Bezos",LocalDate.*parse*("1964-01-12"));  
 Client cl3 = createClient("James", "Gosling",LocalDate.*parse*("1955-05-19"));  
 Client cl4 = createClient("Rosa", "Parks",LocalDate.*parse*("1913-01-04"));  
  
  
 Facture fac1 = createFacture(cl1);  
 LigneFacture lf1 = createLigneFacture(a1,2,fac1);  
 LigneFacture lf2 = createLigneFacture(a2,4,fac1);  
  
  
  
 Facture fac2 = createFacture(cl2);  
 LigneFacture lf3 = createLigneFacture(a3,1,fac2);  
  
 Facture fac3 = createFacture(cl3);  
 LigneFacture lf4 = createLigneFacture(a2,1,fac3);  
 LigneFacture lf5 = createLigneFacture(a1,2,fac3);  
 LigneFacture lf6 = createLigneFacture(a3,3,fac3);  
  
 Facture fac4 = createFacture(cl4);  
 LigneFacture lf7 = createLigneFacture(a1,8,fac4);  
 LigneFacture lf8 = createLigneFacture(a2,3,fac4);  
  
 Facture fac5 = createFacture(cl1);  
 LigneFacture lf9 = createLigneFacture(a3,1,fac5);  
 LigneFacture lf10 = createLigneFacture(a1,14,fac5);  
  
 Facture fac6 = createFacture(cl2);  
 LigneFacture lf12 = createLigneFacture(a1,4,fac6);  
  
 Facture fac7 = createFacture(cl4);  
 LigneFacture lf13 = createLigneFacture(a1,1,fac7);  
 LigneFacture lf14 = createLigneFacture(a2,1,fac7);  
 LigneFacture lf15 = createLigneFacture(a3,1,fac7);  
  
 Facture fac8 = createFacture(cl4);  
 LigneFacture lf17 = createLigneFacture(a2,8,fac8);  
 }  
  
 private Client createClient(String prenom, String nom, LocalDate date) {  
 Client client = new Client();  
 client.setPrenom(prenom);  
 client.setNom(nom);  
 client.setDateNaisance(date);  
 entityManager.persist(client);  
 return client;  
 }  
  
 private Article createArticle(String libelle, double prix, int stock,String description) {  
 Article a1 = new Article();  
 a1.setLibelle(libelle);  
 a1.setPrix(prix);  
 a1.setStock(stock);  
 a1.setDescription(description);  
 entityManager.persist(a1);  
 return a1;  
 }  
  
 private Facture createFacture(Client client){  
 Facture facture = new Facture();  
 facture.setClient(client);  
 entityManager.persist(facture);  
 return facture;  
 }  
  
 private LigneFacture createLigneFacture(Article article,int quantite,Facture facture){  
 LigneFacture lf = new LigneFacture();  
 lf.setArticle(article);  
 lf.setQuantite(quantite);  
 lf.setFacture(facture);  
 entityManager.persist(lf);  
 return lf;  
 }  
}

ArticleExportCVSService :

package com.example.demo.service.export;  
  
import com.example.demo.entity.Article;  
import com.example.demo.repository.ArticleRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.OutputStreamWriter;  
import java.io.PrintWriter;  
  
@Service  
public class ArticleExportCVSService {  
  
 @Autowired  
 private ArticleRepository articleRepository;  
  
 public void export(PrintWriter writer) {  
 writer.println("Libellé;Prix;Description produit");  
  
  
 for (Article article :articleRepository.findAll() ) {  
 writer.println(String.*format*("\"%s\";\"%s\";\"%s\"",article.getLibelle(),article.getPrix(),article.getDescription()));  
  
 }  
  
 }  
  
  
}

ArticleExportXLSXService

package com.example.demo.service.export;  
  
import com.example.demo.entity.Article;  
import com.example.demo.repository.ArticleRepository;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.time.LocalDate;  
import java.util.Calendar;  
import java.util.List;  
  
@Service  
public class ArticleExportXLSXService {  
  
 @Autowired  
 private ArticleRepository articleRepository;  
  
 public void export(OutputStream outputSteam) {  
 try {  
 // Apache POI  
 Workbook wb = new XSSFWorkbook();  
 // *TODO* // création de la feuille  
 Sheet sheet = wb.createSheet("Feuille 1");  
  
 // création de l'entête  
 Row row = sheet.createRow(0);  
 row.createCell(0).setCellValue("Libellé");  
 row.createCell(1).setCellValue("Description"); // largeur taille 169.14  
 row.createCell(2).setCellValue("Prix");  
 row.createCell(3).setCellValue("Stock");  
  
  
 List<Article> articleList = articleRepository.findAll();  
 for (int i = 0; i < articleList.size(); i++) {  
 row = sheet.createRow(i+1);  
 row.createCell(0).setCellValue(articleList.get(i).getLibelle());  
 row.createCell(1).setCellValue(articleList.get(i).getDescription());  
 row.createCell(2).setCellValue(articleList.get(i).getPrix());  
 row.createCell(3).setCellValue(articleList.get(i).getStock());  
 }  
 wb.write(outputSteam);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

ClientExportCVSService.

package com.example.demo.service.export;  
  
import com.example.demo.entity.Client;  
import com.example.demo.repository.ArticleRepository;  
import com.example.demo.repository.ClientRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.PrintWriter;  
import java.time.LocalDate;  
import java.time.Period;  
  
@Service  
public class ClientExportCVSService {  
  
 @Autowired  
 private ClientRepository clientRepository;  
  
 public void export(PrintWriter writer) {  
  
 writer.println("NOM;PRENOM;AGE");  
 for (Client client :clientRepository.findAll() ) {  
 int age = Period.*between*(client.getDateNaisance(), LocalDate.*now*()).getYears();  
 writer.println( String.*format*("\"%s\";\"%s\";\"%s\" ans",client.getNom(),client.getPrenom(),age));  
 }  
  
 }  
  
  
}

ClientExportXLSXService

package com.example.demo.service.export;  
  
import com.example.demo.dto.ClientDto;  
import com.example.demo.entity.Article;  
import com.example.demo.entity.Client;  
import com.example.demo.repository.ArticleRepository;  
import com.example.demo.repository.ClientRepository;  
import com.example.demo.service.ClientService;  
import org.apache.poi.hssf.util.HSSFColor;  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFColor;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.time.Period;  
import java.util.List;  
  
@Service  
public class ClientExportXLSXService {  
  
 @Autowired  
 private ClientService clientService;  
 Workbook wb;  
 public void export(OutputStream outputSteam) {  
 try {  
 // Apache POI  
 wb = new XSSFWorkbook();  
 // *TODO* // création de la feuille  
 Sheet sheet = wb.createSheet("Feuille 1");  
  
 // création de l'entête  
 Row row = sheet.createRow(0);  
 CellStyle style = wb.createCellStyle();  
  
 // application du style et de la valeurs  
 CellStyle(row.createCell(0),"NOM",true);  
 CellStyle(row.createCell(1),"PRENOM",true);  
 CellStyle(row.createCell(2),"AGE",true);  
  
 List<ClientDto> clientDtoList = clientService.findAll();  
 for (int i = 0; i < clientDtoList.size(); i++) {  
 row = sheet.createRow(i+1);  
 CellStyle(row.createCell(0),clientDtoList.get(i).getNom(),false);  
 CellStyle(row.createCell(1),clientDtoList.get(i).getPrenom(),false);  
 CellStyle(row.createCell(2),clientDtoList.get(i).getAge().toString() + " ans",false);  
 }  
 wb.write(outputSteam);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void CellStyle (Cell cell,String value,Boolean Header){  
 CellStyle style = wb.createCellStyle();  
 style.setBorderBottom(BorderStyle.*THICK*);  
 style.setBottomBorderColor(IndexedColors.*BLUE*.getIndex());  
 style.setBorderLeft(BorderStyle.*THICK*);  
 style.setLeftBorderColor(IndexedColors.*BLUE*.getIndex());  
 style.setBorderRight(BorderStyle.*THICK*);  
 style.setRightBorderColor(IndexedColors.*BLUE*.getIndex());  
 style.setBorderTop(BorderStyle.*THICK*);  
 style.setTopBorderColor(IndexedColors.*BLUE*.getIndex());  
 cell.setCellStyle(style);  
 cell.setCellValue(value);  
 if (Header)  
 {  
 Font font = wb.createFont();  
 font.setBold(true);  
 font.setColor(IndexedColors.*PINK*.getIndex());  
 style.setFont(font);  
 }  
 }  
  
}

ClientServiceImpl

package com.example.demo.service.impl;  
  
import com.example.demo.dto.ClientDto;  
import com.example.demo.entity.Client;  
import com.example.demo.repository.ClientRepository;  
import com.example.demo.service.ClientService;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Service contenant les actions métiers liées aux articles.  
 \*/*@Service  
@Transactional  
public class ClientServiceImpl implements ClientService {  
  
 private ClientRepository clientRepository;  
  
 public ClientServiceImpl(ClientRepository clientRepository) {  
 this.clientRepository = clientRepository;  
 }  
  
 @Override  
 public List<ClientDto> findAll() {  
 List<Client> clients = clientRepository.findAll();  
  
 List<ClientDto> dtos = new ArrayList<>();  
 for (Client client : clients) {  
 ClientDto clientDto = new ClientDto(client.getId(), client.getNom(), client.getPrenom(),client.getDateNaisance());  
 dtos.add(clientDto);  
 }  
 return dtos;  
 }  
  
}

ClientMapper

package com.example.demo.service.mapper;  
  
import com.example.demo.dto.ClientDto;  
import com.example.demo.entity.Client;  
import org.springframework.stereotype.Component;  
  
@Component  
public class ClientMapper {  
 public ClientDto clientDto(Client client) {  
 return new ClientDto(client.getId(), client.getNom(), client.getPrenom(),client.getDateNaisance());  
 }  
}

clients.mustache

<table border="1" class="table">  
 <thead>  
 <tr>  
 <th>Nom</th>  
 <th>Prénom</th>  
 <th class="text-right">Age</th>  
 <th>Date de naissance</th>  
 <th>Ses factures</th>  
 </tr>  
 </thead>  
 <tbody>  
 {{#clients}}  
 <tr>  
 <td>{{nom}}</td>  
 <td>{{prenom}}</td>  
 <td class="text-right">{{age}}</td>  
 <td>{{dateNaissance}}</td>  
 <td><a href="export/clients/{{id}}/factures/xlsx">Télécharger toutes ses factures</a></td>  
 </tr>  
 {{/clients}}  
 </tbody>  
</table>

ExportFactureController.

package com.example.demo.controller.export;  
  
import com.example.demo.service.export.ArticleExportCVSService;  
import com.example.demo.service.export.ArticleExportXLSXService;  
import com.example.demo.service.export.FactureExportXLSXService;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
import org.apache.poi.ss.usermodel.Workbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
  
*/\*\*  
 \* Controller pour réaliser export des articles.  
 \*/*@Controller  
@RequestMapping("export")  
public class ExportFactureController {  
  
  
 @Autowired  
 private FactureExportXLSXService factureExportXLSXService;  
 */\*\*  
 \* Export des articles au format CSV.  
 \*/* @GetMapping("factures/xlsx")  
 public void articlesCSV(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 response.setContentType("application/vnd.ms-excel");  
 response.setHeader("Content-Disposition", "attachment; filename=\"export-factures.xlsx\"");  
 OutputStream outputStream = response.getOutputStream();  
 factureExportXLSXService.export(outputStream);  
 }  
  
  
  
  
  
}

FactureExportXLSXService

package com.example.demo.service.export;  
  
import com.example.demo.dto.ClientDto;  
import com.example.demo.dto.FactureDto;  
import com.example.demo.entity.Client;  
import com.example.demo.entity.Facture;  
import com.example.demo.entity.LigneFacture;  
import com.example.demo.repository.ClientRepository;  
import com.example.demo.service.ClientService;  
import com.example.demo.service.FactureService;  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Set;  
  
@Service  
public class FactureExportXLSXService {  
  
 @Autowired  
 private FactureService factureService;  
  
 @Autowired  
 private ClientRepository clientRepository;  
  
 Workbook wb;  
 public void export(OutputStream outputSteam) {  
 try {  
 // Apache POI  
 wb = new XSSFWorkbook();  
 // *TODO* List<Client> clientList = clientRepository.findAll();  
 for( Client client : clientList) {  
 Sheet sheet = wb.createSheet(client.getNom() + " " + client.getPrenom());  
 createCellClient(sheet.createRow(0),"Nom : ",client.getNom());  
 createCellClient(sheet.createRow(1),"Prénom : ",client.getPrenom());  
 createCellClient(sheet.createRow(2),"Année de naissance : ",client.getDateNaisance().getYear());  
 createCellClientTotalFacture(sheet.createRow(3),client);  
  
 sheet.autoSizeColumn(0,true);  
  
 // creation feuille(s) facture:  
 Set<Facture> factureSet = client.getFactures();  
 for (Facture facture : factureSet) {  
 Sheet sheetFacture = wb.createSheet("Facture N° "+facture.getId().toString());  
 createHeaderCellLigneFacture(sheetFacture.createRow(0));  
  
  
 // create des ligne factures:  
 Set<LigneFacture> ligneFactureSet = facture.getLigneFactures();  
 List<LigneFacture> ligneFactureList = new ArrayList<LigneFacture>(ligneFactureSet);  
 int RowTotalPrice = ligneFactureList.size()+1;  
 double priceTotal=0;  
 for (int i = 0; i < ligneFactureList.size(); i++) {  
 priceTotal += createCellLigneFacture(sheetFacture.createRow(i+1),ligneFactureList.get(i));  
 }  
 createRowTotalLigneFacture(sheetFacture.createRow(RowTotalPrice),priceTotal);  
  
 sheetFacture.autoSizeColumn(0,true);  
 sheetFacture.autoSizeColumn(1,true);  
 sheetFacture.autoSizeColumn(2,true);  
 }  
 }  
 wb.write(outputSteam);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void createCellClient(Row row,String Name,Object value){  
 Cell cellOne = row.createCell(0);  
 cellOne.setCellValue(Name);  
 Cell cellTwo = row.createCell(1);  
 cellTwo.setCellValue(value.toString());  
 }  
  
 private void createCellClientTotalFacture(Row row,Client client){  
 CellStyle style = wb.createCellStyle();  
 Font font = wb.createFont();  
 font.setBold(true);  
 style.setFont(font);  
  
 Cell cellOne = row.createCell(0);  
 cellOne.setCellValue(client.getFactures().size() +" facture(s)");  
 cellOne.setCellStyle(style);  
 for (int i = 0; i < client.getFactures().size(); i++) {  
 Cell cell= row.createCell(i+1);  
 Set<Facture> factureSet = client.getFactures();  
 List<Facture> factures = new ArrayList<Facture>(factureSet);  
 cell.setCellValue(factures.get(i).getId());  
 cell.setCellStyle(style);  
 }  
 }  
  
 private void createHeaderCellLigneFacture(Row row){  
 CellStyle style = wb.createCellStyle();  
 Font font = wb.createFont();  
 font.setBold(true);  
 style.setFont(font);  
  
 Cell cell1 = row.createCell(0);  
 cell1 .setCellValue("Désignation");  
 cell1 .setCellStyle(style);  
  
 Cell cell2 = row.createCell(1);  
 cell2.setCellValue("Quantité");  
 cell2 .setCellStyle(style);  
  
 Cell cell3 = row.createCell(2);  
 cell3.setCellValue("Prix unitaire");  
 cell3 .setCellStyle(style);  
 }  
  
 private double createCellLigneFacture(Row row , LigneFacture ligneFacture){  
 Cell cell1 = row.createCell(0);  
 cell1.setCellValue(ligneFacture.getArticle().getLibelle());  
  
  
 Cell cell2 = row.createCell(1);  
 cell2.setCellValue(ligneFacture.getQuantite());  
  
  
 Cell cell3 = row.createCell(2);  
 cell3.setCellValue(ligneFacture.getArticle().getPrix());  
  
 return ligneFacture.getQuantite() \* ligneFacture.getArticle().getPrix();  
 }  
  
 private void createRowTotalLigneFacture(Row row,double price){  
 CellStyle style = wb.createCellStyle();  
 Font font = wb.createFont();  
 font.setBold(true);  
 style.setFont(font);  
  
 Cell cell2 = row.createCell(1);  
 cell2.setCellValue("Total : ");  
 cell2.setCellStyle(style);  
  
  
 Cell cell3 = row.createCell(2);  
 cell3.setCellValue(price);  
 cell3.setCellStyle(style);  
  
 }  
}

ExportPDFITextService

package com.example.demo.service.export;  
  
import com.example.demo.entity.Article;  
import com.example.demo.repository.ArticleRepository;  
import com.itextpdf.text.\*;  
import com.itextpdf.text.pdf.PdfPCell;  
import com.itextpdf.text.pdf.PdfPTable;  
import com.itextpdf.text.pdf.PdfWriter;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.OutputStream;  
import java.io.PrintWriter;  
  
@Service  
public class ExportPDFITextService {  
  
 @Autowired  
 private ArticleRepository articleRepository;  
  
 public void export(OutputStream outputSteam) throws DocumentException {  
  
 Document document = new Document();  
 PdfWriter.*getInstance*(document, outputSteam);  
 document.open();  
  
 PdfPTable table = new PdfPTable(3); // 3 columns.  
 table.setWidthPercentage(100); //Width 100%  
 table.setSpacingBefore(10f); //Space before table  
 table.setSpacingAfter(10f); //Space after table  
  
 //Set Column widths  
 float[] columnWidths = {1f, 1f, 1f};  
 table.setWidths(columnWidths);  
  
 PdfPCell cell1 = new PdfPCell(new Paragraph("LIBELLÉ"));  
 cell1.setBackgroundColor(BaseColor.*LIGHT\_GRAY*);  
 cell1.setPaddingLeft(10);  
 cell1.setHorizontalAlignment(Element.*ALIGN\_CENTER*);  
 cell1.setVerticalAlignment(Element.*ALIGN\_MIDDLE*);  
  
 PdfPCell cell2 = new PdfPCell(new Paragraph("PRIX"));  
 cell2.setBackgroundColor(BaseColor.*LIGHT\_GRAY*);  
 cell2.setPaddingLeft(10);  
 cell2.setHorizontalAlignment(Element.*ALIGN\_CENTER*);  
 cell2.setVerticalAlignment(Element.*ALIGN\_MIDDLE*);  
  
 PdfPCell cell3 = new PdfPCell(new Paragraph("DESCRIPTION"));  
 cell3.setBackgroundColor(BaseColor.*LIGHT\_GRAY*);  
 cell3.setPaddingLeft(10);  
 cell3.setHorizontalAlignment(Element.*ALIGN\_CENTER*);  
 cell3.setVerticalAlignment(Element.*ALIGN\_MIDDLE*);  
  
  
 table.addCell(cell1);  
 table.addCell(cell2);  
 table.addCell(cell3);  
  
 createCellArticle(table,document);  
  
 // step 5  
 document.close();  
  
 }  
  
 public void createCellArticle(PdfPTable table , Document document) throws DocumentException {  
  
 for (Article article : articleRepository.findAll()) {  
 PdfPCell cellLibellé = new PdfPCell(new Paragraph(article.getLibelle()));  
 Double prix = article.getPrix();  
 PdfPCell cellPrix= new PdfPCell(new Paragraph(prix.toString()));  
 cellPrix.setHorizontalAlignment(Element.*ALIGN\_RIGHT*);  
 PdfPCell cellDescription = new PdfPCell(new Paragraph(article.getDescription()));  
 table.addCell(cellLibellé);  
 table.addCell(cellPrix);  
 table.addCell(cellDescription);  
 }  
 document.add(table);  
 }  
  
}