**Report Generation Service**

Table of Contents

[**PDF Generation Service** 1](#_Toc157956192)

[**CSV Generation Service** 4](#_Toc157956193)

[**Text Generation Service** 6](#_Toc157956194)

[**RTF Report Generation** 8](#_Toc157956195)

[**Excel Report Generation** 10](#_Toc157956196)

## **PDF Generation Service**

The PDFGenerationService orchestrates the retrieval of data and utilizes the PdfExporter to create PDF reports with tabular data, providing clients with downloadable PDF files.

#### **Project Structure**

    my-spring-boot-app

    |-- src

    |   |-- main

    |       |-- java

    |           |-- com.example

    |               |-- MySpringBootApplication.java

    |               |-- model

    |                   |-- PersonalInfo.java

    |                   |-- ProfessionalInfo.java

    |               |-- repository

    |                   |-- PersonalInfoRepository.java

    |                   |-- ProfessionalInfoRepository.java

    |               |-- service

    |                   |-- InfoService.java

    |                   |-- PDFGenerationService.java

    |                   |-- PDFExporter.java

    |               |-- controller

    |                   |-- InfoController.java

    |                   |-- PDFGenerationController.java

    |-- src

    |   |-- main

    |       |-- resources

    |           |-- application.properties

    |-- pom.xml

1. **MySpringBootApplication.java:**
   * Main class with the main method to run the Spring Boot application.
2. **model Package:**
   * Contains entity classes representing personal and professional information.
3. **repository Package:**
   * Spring Data JPA repositories for personal and professional information.
4. **service Package:**
   * InfoService: Service interface for retrieving information.
   * ReportGenerationService: Service interface for generating PDF reports.
   * PDFExporter: Component responsible for PDF generation using Apache PDFBox.
5. **controller Package:**
   * InfoController: Controller for handling requests related to information retrieval.
   * ReportGenerationController: Controller for handling requests related to PDF generation.
6. **resources Directory:**
   * application.properties: Configuration properties file.
7. **pom.xml:**
   * Maven Project Object Model file specifying project dependencies, plugins, and configurations.

The PDFExporter component in the service package is responsible for generating PDF reports using Apache PDFBox. The controllers (**InfoController** and **ReportGenerationController**) handle HTTP requests related to information retrieval and PDF generation, respectively.

Ensure to add the Apache PDFBox dependency to your **pom.xml**:

<dependency>

    <groupId>org.apache.pdfbox</groupId>

    <artifactId>pdfbox</artifactId>

    <version>2.0.31</version> <!-- Use the latest version available -->

</dependency>

#### **Detailed Flow for PDF Generation (using PDFBox):**

1. **Client Sends Request:**
   * The client sends an HTTP request to the ReportGenerationController to initiate the PDF generation process.
2. **ReportGenerationController:**
   * The ReportGenerationController receives the request and delegates the request to the ReportGenerationService.
3. **ReportGenerationService:**
   * The ReportGenerationService is responsible for orchestrating the PDF generation process.
   * It interacts with the InfoService to retrieve necessary data (personal and professional information).
4. **InfoService:**
   * The InfoService communicates with the InfoRepository to fetch personal and professional information from the database.
   * It returns the retrieved data to the ReportGenerationService.
5. **ReportGenerationService (contd.):**
   * The ReportGenerationService has obtained the required data.
   * It invokes the PDFExporter component to generate the PDF report.
6. **PDFExporter:**
   * The PDFExporter is a specialized component responsible for PDF generation using Apache PDFBox.
   * It creates a PDF document, adds content, and saves it to a byte array.
7. **PDFExporter (contd.):**
   * The PDFExporter utilizes Apache PDFBox to create a PDF document.
   * It adds content to the document based on the retrieved data from the InfoService.
   * The PDF content is then saved to a byte array.
8. **ReportGenerationService (contd.):**
   * The ReportGenerationService receives the generated PDF content from the PDFExporter.
9. **ReportGenerationController (contd.):**
   * The ReportGenerationController receives the generated PDF content and prepares the HTTP response.
10. **HTTP Response:**
    * The ReportGenerationController sends an HTTP response to the client.
    * The response includes the generated PDF content as a downloadable file.
11. **Client Receives PDF:**
    * The client receives the HTTP response with the PDF file.
    * The client can download and view the PDF document.

#### **Notes**

* **PDF Exporter (using PDFBox):**
  + Apache PDFBox is a Java library for working with PDF documents.
  + In the PDFExporter component, you can use PDFBox to create a PDDocument, add content, and convert it to a byte array.
  + Ensure to handle exceptions and cleanup resources properly.
* **PDF Content Structure:**
  + The structure of the PDF content, such as headers, tables, or other elements, is defined within the PDFExporter component based on your requirements.

## **CSV Generation Service**

The **CSVGenerationService** fetches data through the InfoService and employs the **CSVExporter** to generate CSV reports, facilitating the creation of downloadable CSV files for clients.

#### **Project Structure**

    my-spring-boot-app

|-- src

|   |-- main

|       |-- java

|           |-- com.example

|               |-- MySpringBootApplication.java

|               |-- model

|                   |-- PersonalInfo.java

|                   |-- ProfessionalInfo.java

|               |-- repository

|                   |-- PersonalInfoRepository.java

|                   |-- ProfessionalInfoRepository.java

|               |-- service

|                   |-- InfoService.java

|                   |-- CSVGenerationService.java

|                   |-- CSVExporter.java

|               |-- controller

|                   |-- InfoController.java

|                   |-- CSVGenerationController.java

|-- src

|   |-- main

|       |-- resources

|           |-- application.properties

|-- pom.xml

#### **Detailed Flow for CSV Report Generation:**

1. **Client Sends Request:**
   * The client sends an HTTP request to the ReportGenerationController to initiate the CSV report generation process.
2. **ReportGenerationController:**
   * The ReportGenerationController receives the request and delegates the request to the ReportGenerationService.
3. **ReportGenerationService:**
   * The ReportGenerationService is responsible for orchestrating the CSV report generation process.
   * It interacts with the InfoService to retrieve necessary data (personal and professional information).
4. **InfoService:**
   * The InfoService communicates with the InfoRepository to fetch personal and professional information from the database.
   * It returns the retrieved data to the ReportGenerationService.
5. **ReportGenerationService (contd.):**
   * The ReportGenerationService has obtained the required data.
   * It invokes the CSVExporter component to generate the CSV report.
6. **CSVExporter:**
   * The CSVExporter is a specialized component responsible for CSV generation.
   * It creates a CSV file and adds content based on the retrieved data.
7. **CSVExporter (contd.):**
   * The CSVExporter utilizes a library (e.g., **Apache Commons CSV**) to create a CSV file.
   * It adds content to the CSV file based on the retrieved data.
8. **ReportGenerationService (contd.):**
   * The ReportGenerationService receives the generated CSV content from the CSVExporter.
9. **ReportGenerationController (contd.):**
   * The ReportGenerationController receives the generated CSV content and prepares the HTTP response.
10. **HTTP Response:**
    * The ReportGenerationController sends an HTTP response to the client.
    * The response includes the generated CSV content as a downloadable file.
11. **Client Receives CSV Report:**
    * The client receives the HTTP response with the CSV file.
    * The client can download and view the CSV document.

#### **CSV Generation Description:**

* **ReportGenerationService:**
  + Orchestrates the CSV report generation process.
  + Retrieves data from the InfoService.
  + Utilizes CSVExporter to create and write the CSV content.
* **CSVExporter:**
  + The CSV generation logic is embedded within the ReportGenerationService.
  + Creates a CSV file and adds content based on the retrieved data.
* **Flow Description:**
  + The ReportGenerationController handles the HTTP request for CSV report generation.
  + It delegates the request to the ReportGenerationService.
  + The ReportGenerationService interacts with the InfoService to retrieve data.
  + CSV generation is performed within the CSVExporter logic.
  + The generated CSV content is returned to the client as an attachment in the HTTP response.

This structure and flow reflect a simplified approach where CSV generation is handled within the ReportGenerationService.

## **Text Generation Service**

The **TextGenerationService** uses the **TextExporter** to generate text reports by creating plain text files based on retrieved data. Clients can download these text files for easy viewing and analysis.

Project Structure

    my-spring-boot-app

|-- src

|   |-- main

|       |-- java

|           |-- com.example

|               |-- MySpringBootApplication.java

|               |-- model

|                   |-- PersonalInfo.java

|                   |-- ProfessionalInfo.java

|               |-- repository

|                   |-- PersonalInfoRepository.java

|                   |-- ProfessionalInfoRepository.java

|               |-- service

|                   |-- InfoService.java

|                   |-- TextGenerationService.java

|                   |-- TextExporter.java

|               |-- controller

|                   |-- InfoController.java

|                   |-- TextGenerationController.java

|-- src

|   |-- main

|       |-- resources

|           |-- application.properties

|-- pom.xml

#### **Detailed Flow for Text Report Generation:**

1. **Client Sends Request:**
   * The client sends an HTTP request to the ReportGenerationController to initiate the text report generation process.
2. **ReportGenerationController:**
   * The ReportGenerationController receives the request and delegates the request to the ReportGenerationService.
3. **ReportGenerationService:**
   * The ReportGenerationService is responsible for orchestrating the text report generation process.
   * It interacts with the InfoService to retrieve necessary data (personal and professional information).
4. **InfoService:**
   * The InfoService communicates with the InfoRepository to fetch personal and professional information from the database.
   * It returns the retrieved data to the ReportGenerationService.
5. **ReportGenerationService (contd.):**
   * The ReportGenerationService has obtained the required data.
   * It invokes the TextExporter component to generate the text report.
6. **TextExporter:**
   * The TextExporter is a specialized component responsible for text generation.
   * It creates a text file and adds content based on the retrieved data.
7. **TextExporter (contd.):**
   * The TextExporter writes content to a text file based on the retrieved data.
8. **ReportGenerationService (contd.):**
   * The ReportGenerationService receives the generated text content from the TextExporter.
9. **ReportGenerationController (contd.):**
   * The ReportGenerationController receives the generated text content and prepares the HTTP response.
10. **HTTP Response:**
    * The ReportGenerationController sends an HTTP response to the client.
    * The response includes the generated text content as a downloadable file.
11. **Client Receives Text Report:**
    * The client receives the HTTP response with the text file.
    * The client can download and view the text document.

#### **Text Generation Description:**

* **ReportGenerationService:**
  + Orchestrates the text report generation process.
  + Retrieves data from the InfoService.
  + Utilizes TextExporter to create and write the text content.
* **TextExporter:**
  + The text generation logic is embedded within the ReportGenerationService.
  + Creates a text file and adds content based on the retrieved data.
* **Flow Description:**
  + The ReportGenerationController handles the HTTP request for text report generation.
  + It delegates the request to the ReportGenerationService.
  + The ReportGenerationService interacts with the InfoService to retrieve data.
  + Text generation is performed within the TextExporter logic.
  + The generated text content is returned to the client as an attachment in the HTTP response.

This structure and flow reflect a simplified approach where text generation is handled within the ReportGenerationService

## **RTF Report Generation**

The **RTFGenerationService** interacts with the **RTFExporter** to create rich-text format (RTF) reports. This service converts retrieved data into formatted RTF documents, providing clients with downloadable files.

#### **Project Structure**

    my-spring-boot-app

    |-- src

    |   |-- main

    |       |-- java

    |           |-- com.example

    |               |-- MySpringBootApplication.java

    |               |-- model

    |                   |-- PersonalInfo.java

    |                   |-- ProfessionalInfo.java

    |               |-- repository

    |                   |-- PersonalInfoRepository.java

    |                   |-- ProfessionalInfoRepository.java

    |               |-- service

    |                   |-- InfoService.java

    |                   |-- RTFGenerationService.java

    |                   |-- RTFExporter.java

    |               |-- controller

    |                   |-- InfoController.java

    |                   |-- RTFGenerationController.java

    |-- src

    |   |-- main

    |       |-- resources

    |           |-- application.properties

    |-- pom.xml

#### **Detailed Flow for RTF Report Generation:**

1. **Client Sends Request:**
   * The client sends an HTTP request to the ReportGenerationController to initiate the RTF report generation process.
2. **ReportGenerationController:**
   * The ReportGenerationController receives the request and delegates the request to the ReportGenerationService.
3. **ReportGenerationService:**
   * The ReportGenerationService is responsible for orchestrating the RTF report generation process.
   * It interacts with the InfoService to retrieve necessary data (personal and professional information).
4. **InfoService:**
   * The InfoService communicates with the InfoRepository to fetch personal and professional information from the database.
   * It returns the retrieved data to the ReportGenerationService.
5. **ReportGenerationService (contd.):**
   * The ReportGenerationService has obtained the required data.
   * It invokes the RTFExporter component to generate the RTF report.
6. **RTFExporter:**
   * The RTF Exporter is a specialized component responsible for RTF generation.
   * It creates an RTF document and adds formatted content based on the retrieved data.
7. **RTFExporter (contd.):**
   * The RTFExporter utilizes a library (e.g., Apache POI HWPF) to create an RTF document.
   * It adds formatted content to the RTF document based on the retrieved data.
8. **ReportGenerationService (contd.):**
   * The ReportGenerationService receives the generated RTF content from the RTFExporter.
9. **ReportGenerationController (contd.):**
   * The ReportGenerationController receives the generated RTF content and prepares the HTTP response.
10. **HTTP Response:**
    * The ReportGenerationController sends an HTTP response to the client.
    * The response includes the generated RTF content as a downloadable file.
11. **Client Receives RTF Report:**
    * The client receives the HTTP response with the RTF file.
    * The client can download and view the RTF document.

#### **RTF Generation Description:**

* **ReportGenerationService:**
  + Orchestrates the RTF report generation process.
  + Retrieves data from the InfoService.
  + Utilizes RTFExporter to create and write the RTF content.
* **RTFExporter:**
  + The RTF generation logic is embedded within the ReportGenerationService.
  + Creates an RTF document and adds formatted content based on the retrieved data.
* **Flow Description:**
  + The ReportGenerationController handles the HTTP request for RTF report generation.
  + It delegates the request to the ReportGenerationService.
  + The ReportGenerationService interacts with the InfoService to retrieve data.
  + RTF generation is performed within the RTFExporter logic.
  + The generated RTF content is returned to the client as an attachment in the HTTP response.

This structure and flow reflect a simplified approach where RTF generation is handled within the ReportGenerationService.

## **Excel Report Generation**

The **ExcelGenerationService** coordinates with the **ExcelExporter** to create Excel reports. Data from the InfoService is structured into sheets within the Excel workbook, offering clients downloadable Excel files.

#### **Project Structure**

    my-spring-boot-app

    |-- src

    |   |-- main

    |       |-- java

    |           |-- com.example

    |               |-- MySpringBootApplication.java

    |               |-- model

    |                   |-- PersonalInfo.java

    |                   |-- ProfessionalInfo.java

    |               |-- repository

    |                   |-- PersonalInfoRepository.java

    |                   |-- ProfessionalInfoRepository.java

    |               |-- service

    |                   |-- InfoService.java

    |                   |-- ExcelGenerationService.java

    |                   |-- ExcelExporter.java

    |               |-- controller

    |                   |-- InfoController.java

    |                   |-- ExcelGenerationController.java

    |-- src

    |   |-- main

    |       |-- resources

    |           |-- application.properties

    |-- pom.xml

#### **Detailed Flow for Excel Report Generation:**

1. **Client Sends Request:**
   * The client sends an HTTP request to the ReportGenerationController to initiate the Excel report generation process.
2. **ReportGenerationController:**
   * The ReportGenerationController receives the request and delegates the request to the ReportGenerationService.
3. **ReportGenerationService:**
   * The ReportGenerationService is responsible for orchestrating the Excel report generation process.
   * It interacts with the InfoService to retrieve necessary data (personal and professional information).
4. **InfoService:**
   * The InfoService communicates with the InfoRepository to fetch personal and professional information from the database.
   * It returns the retrieved data to the ReportGenerationService.
5. **ReportGenerationService (contd.):**
   * The ReportGenerationService has obtained the required data.
   * It invokes the ExcelExporter component to generate the Excel report.
6. **ExcelExporter:**
   * The ExcelExporter is a specialized component responsible for Excel generation.
   * It creates an Excel workbook and adds data to different sheets based on the retrieved data.
7. **ExcelExporter (contd.):**
   * The ExcelExporter utilizes a library (e.g., Apache POI) to create an Excel workbook.
   * It adds data to different sheets within the workbook based on the retrieved data.
8. **ReportGenerationService (contd.):**
   * The ReportGenerationService receives the generated Excel content from the ExcelExporter.
9. **ReportGenerationController (contd.):**
   * The ReportGenerationController receives the generated Excel content and prepares the HTTP response.
10. **HTTP Response:**
    * The ReportGenerationController sends an HTTP response to the client.
    * The response includes the generated Excel content as a downloadable file.
11. **Client Receives Excel Report:**
    * The client receives the HTTP response with the Excel file.
    * The client can download and view the Excel document.

#### **Excel Generation Description:**

* **ReportGenerationService:**
  + Orchestrates the Excel report generation process.
  + Retrieves data from the InfoService.
  + Utilizes ExcelExporter to create and write the Excel content.
* **ExcelExporter:**
  + The Excel generation logic is embedded within the ReportGenerationService.
  + Creates an Excel workbook and adds data to different sheets based on the retrieved data.
* **Flow Description:**
  + The ReportGenerationController handles the HTTP request for Excel report generation.
  + It delegates the request to the ReportGenerationService.
  + The ReportGenerationService interacts with the InfoService to retrieve data.
  + Excel generation is performed within the ExcelExporter logic.
  + The generated Excel content is returned to the client as an attachment in the HTTP response.

This structure and flow reflect a simplified approach where Excel generation is handled within the ReportGenerationService.