DOCUMENTATION

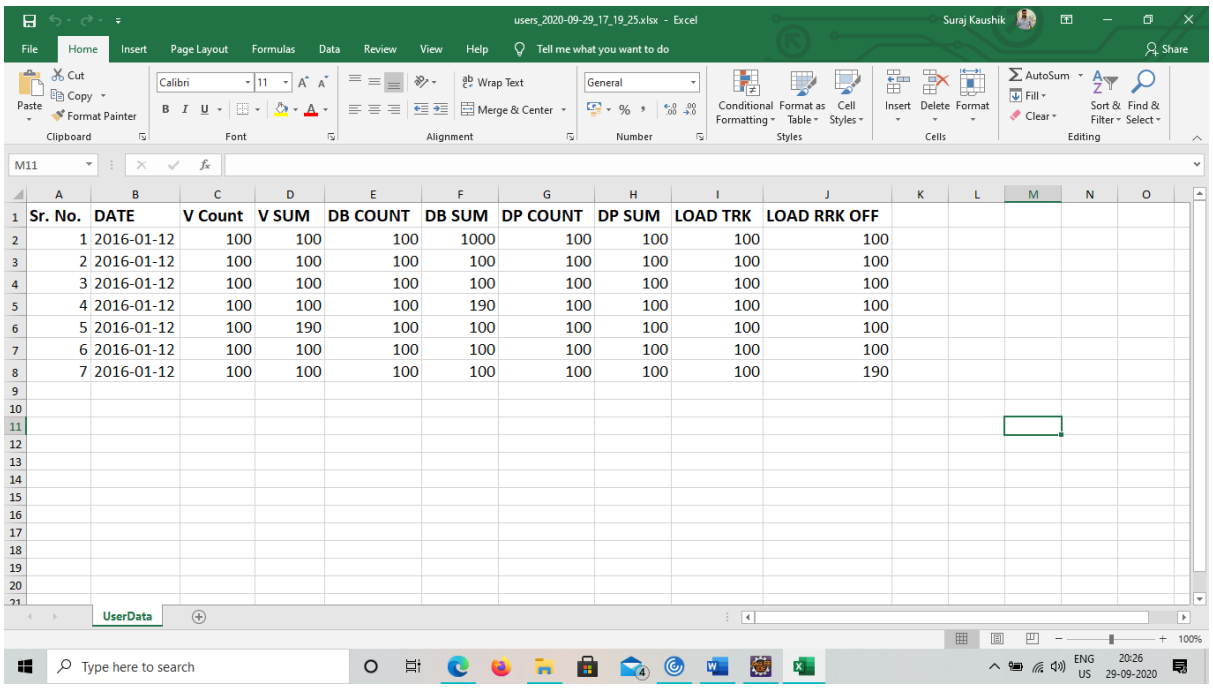
Oracle DB Data to Excel File Using Springboot

Assume that we have a **Datasheet** table in Oracle database like this:

Status | Result1

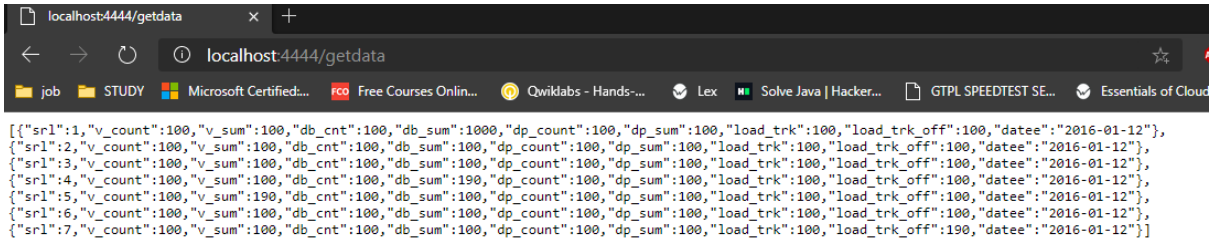
	SRL	DATEE	V_COUNT	V_SUM	DB_CNT	DB_SUM	DP_COUNT	DP_SUM	LOAD_TRK	LOAD_TRK_OFF
1	1	2016...	100	100	100	1000	100	100	100	100
2	2	2016...	100	100	100	100	100	100	100	100
3	3	2016...	100	100	100	100	100	100	100	100
4	4	2016...	100	100	100	190	100	100	100	100
5	5	2016...	100	190	100	100	100	100	100	100
6	6	2016...	100	100	100	100	100	100	100	100
7	7	2016...	100	100	100	100	100	100	100	190

We’re gonna create a Spring Boot Application that provides APIs for downloading MySQL table data as Excel file with following content:

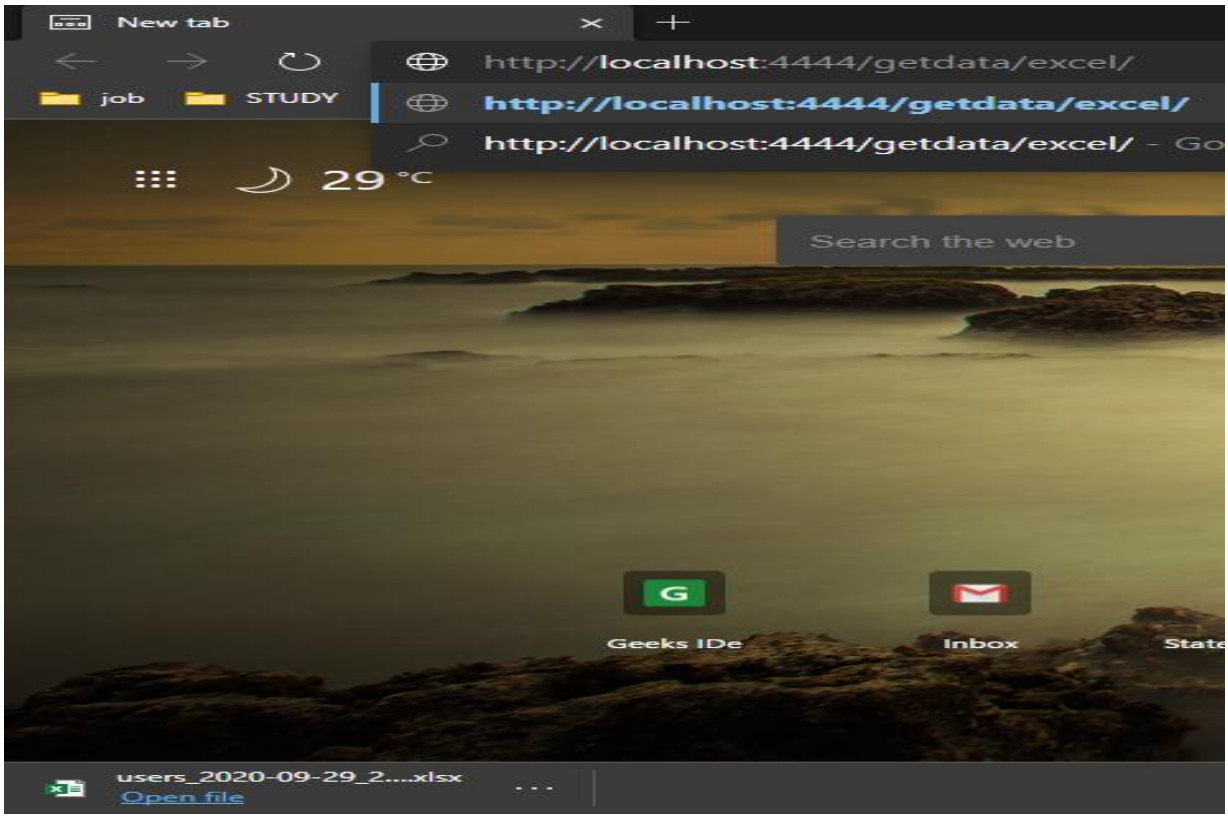


	A	B	C	D	E	F	G	H	I	J
1	Sr. No.	DATE	V Count	V SUM	DB COUNT	DB SUM	DP COUNT	DP SUM	LOAD TRK	LOAD RRK OFF
2	1	2016-01-12	100	100	100	1000	100	100	100	100
3	2	2016-01-12	100	100	100	100	100	100	100	100
4	3	2016-01-12	100	100	100	100	100	100	100	100
5	4	2016-01-12	100	100	100	190	100	100	100	100
6	5	2016-01-12	100	190	100	100	100	100	100	100
7	6	2016-01-12	100	100	100	100	100	100	100	100
8	7	2016-01-12	100	100	100	100	100	100	100	190

If you send request to http://localhost:4444/getdata/, the server will return a response on the webpage that contains data in Oracle table.



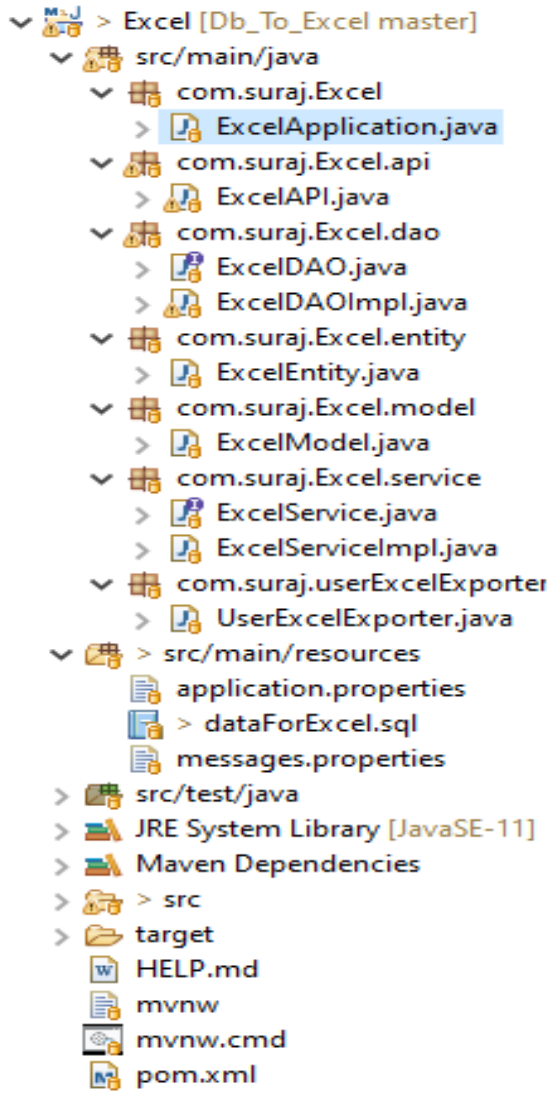
If you send request to http://localhost:4444/getdata/excel ,the server will return a response with an Excel file **users_(Data and Time).xlsx** that contains data in Oracle table.



TECHNOLOGY

- Java 11
- Spring Boot
- Maven
- Apache POI
- Oracle Database

PROJECT STRUCTURE



- **ExcelApplication.java –** This is the Main SpringbootApplication Class, which will initiate the application.
- **ExcelApi.java –** This class carrying API for exporting Excel sheet and viewing on web browser.
- **DAO Package–** This package will be functioning for fetching data from the database.
- **SERVICE Package –** This package will be functioning for fetching data from DAO and providing it to API for the use.
- **ExcelEntity.java –** This class is an Entity class used for interaction with Database.
- **ExcelModel.java –** This class is a Model class used for interaction with Java.
- **UserExcelExporter.java –** This class is dealing with creation of excel file.

REQUIRED DEPENDENCIES

In **pom.xml** and add these dependencies:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
        <dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.1.0</version>
        </dependency>
    <dependency>
        <groupId>com.oracle.database.jdbc</groupId>
        <artifactId>ojdbc8</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

CONFIGURE SPRING DATASOURCE

Under **src/main/resources** folder, open ***application.properties*** and write these lines.

application.properties

```
1
2 server.port=4444
3
4 # Oracle settings
5 spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
6 spring.datasource.username=system
7 spring.datasource.password=root
8
```

FUNCTIONS WITH COMMENTS

API CLASS

- **ExportToExcel Function**

```
//URL to be used for downloading Excel file: http://localhost:4444/getdata/excel
@GetMapping("/getdata/excel")
public void exportToExcel(HttpServletResponse response) throws IOException {
    response.setContentType("application/octet-stream");
    //Set date and time format for file name.
    DateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-dd_HH:mm:ss");
    //Got the current Date and Timestamp.
    String currentDateTime = dateFormatter.format(new Date());
    String headerKey = "Content-Disposition";
    //Set the File name as users_ then date and time with xlsx file extension.
    String headerValue = "attachment; filename=users_" + currentDateTime + ".xlsx";
    response.setHeader(headerKey, headerValue);
  //this statement is fetching data from database using service class.
    List<ExcelModel> excelData = service.getAllData();
    UserExcelExporter excelExporter = new UserExcelExporter(excelData);
    //Called UserExcelExporter class for downloading Excel file.
    excelExporter.export(response);
}
```

- **GetAllData Function**

```
//URL to be used for view data on webpage: http://localhost:4444/getdata/
@GetMapping(value = "/getdata")
public ResponseEntity<List<ExcelModel>> getAllData() throws Exception{
    try {
    //this statement is fetching data from database using service class.
    List<ExcelModel> allData=service.getAllData();
    //this statement is returning the Database Data to the webpage.
    return new ResponseEntity<List<ExcelModel>>(allData, HttpStatus.OK);
    }
    catch (Exception e) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage());
    }

}
```

ENTITY CLASS

- **ExcelEntity Function**

```
@Entity
@Table (name="Datasheet")
//Used To interact with Database.
public class ExcelEntity {
    @Id
    private Integer srl;
    private LocalDate Datee;
    private Integer v_count;
    private Integer v_sum;
    private Integer db_cnt;
    private Integer db_sum;
    private Integer dp_count;
    private Integer dp_sum;
    private Integer load_trk;
    private Integer load_trk_off;
```

MODEL CLASS

- **ExcelModel Function**

```java
//Used To interact with java classes.
public class ExcelModel {
    private Integer srl;
    private LocalDate Datee;
    private Integer v_count;
    private Integer v_sum;
    private Integer db_cnt;
    private Integer db_sum;
    private Integer dp_count;
    private Integer dp_sum;
    private Integer load_trk;
    private Integer load_trk_off;
```

DAO CLASS

- **GetAllData Function**

```java
@Override
public List<ExcelModel> getAllData() {
    //creating an Empty list
    List<ExcelModel> listOfData= new ArrayList<ExcelModel>();
    //JPA query for fetching Oracle database data
    Query q=entityManager.createQuery("SELECT p FROM ExcelEntity p");
    //Storing the fetching Entity results in ReceviedList
    List<ExcelEntity> receivedList=q.getResultList();
    //Iterating over the Entity(receivedList) and converting it to model
    for (ExcelEntity excelEntity : receivedList) {
        //Object as Model class is created
        ExcelModel excel = new ExcelModel();
        //Setting data in model class from Entity Class
        excel.setSrl(excelEntity.getSrl());
        excel.setDatee(excelEntity.getDatee());
        excel.setDb_cnt(excelEntity.getDb_cnt());
        excel.setDb_sum(excelEntity.getDb_sum());
        excel.setDp_count(excelEntity.getDp_count());
        excel.setDp_sum(excelEntity.getDp_sum());
        excel.setLoad_trk(excelEntity.getLoad_trk());
        excel.setLoad_trk_off(excelEntity.getLoad_trk_off());
        excel.setV_count(excelEntity.getV_count());
        excel.setV_sum(excelEntity.getV_sum());
        //Adding all the Model class data in the list.
        listOfData.add(excel);
    }
    //Returning the fetched data from backend.
    return listOfData;
}
```

SERVICE CLASS

- **GetAllData Function**

```java
@Override
//This getAllData Function will retrieve all the ExcelModel data which is fetched by DAO.
public List<ExcelModel> getAllData() {
    //Calling of DAO function
    return excelDAO.getAllData();
}
```

UserExcelExporter CLASS

- **WriteHeaderLine Function**

```
// for the Header row in Excel file
private void writeHeaderLine() {
    //Assigned the worksheet name
     sheet = workbook.createSheet("UserData");
     //Created row in sheet
     Row row = sheet.createRow(0);
     CellStyle style = workbook.createCellStyle();
     //Set the font for the row 0 with bold and size 16
     XSSFFont font = workbook.createFont();
     font.setBold(true);
     font.setFontHeight(16);
     style.setFont(font);

     //Created all the Title for the required database in excel file
     createCell(row, 0, "Sr. No.", style);
     createCell(row, 1, "DATE", style);
     createCell(row, 2, "V Count", style);
     createCell(row, 3, "V SUM ", style);
     createCell(row, 4, "DB COUNT", style);
     createCell(row, 5, "DB SUM", style);
     createCell(row, 6, "DP COUNT", style);
     createCell(row, 7, "DP SUM", style);
     createCell(row, 8, "LOAD TRK", style);
     createCell(row, 9, "LOAD RRK OFF", style);

}
```

- **WriteDataLines Function**

```
//this function will write the data part in the excel sheet
private void writeDataLines() {
    //set the rowCOunt as 1
    int rowCount = 1;
    CellStyle style = workbook.createCellStyle();
    //Created font for the data part and gave font size as 14
    XSSFFont font = workbook.createFont();
    font.setFontHeight(14);
    style.setFont(font);
    //Iterating over the backend data and writing it to the Excel sheet
    for (ExcelModel user : excelData) {
        Row row = sheet.createRow(rowCount++);
        int columnCount = 0;
         //Setting data for excel in each column
        createCell(row, columnCount++, user.getSrl(), style);
        createCell(row, columnCount++, user.getDatee().toString(), style);
        createCell(row, columnCount++, user.getV_count(), style);
        createCell(row, columnCount++, user.getV_sum(), style);
        createCell(row, columnCount++, user.getDb_cnt(), style);
        createCell(row, columnCount++, user.getDb_sum(), style);
        createCell(row, columnCount++, user.getDp_count(), style);
        createCell(row, columnCount++, user.getDp_sum(), style);
        createCell(row, columnCount++, user.getLoad_trk(), style);
        createCell(row, columnCount++, user.getLoad_trk_off(), style);
    }
}
```

- **Export Function**

```
//This is the main function for calling Header and Data lines.
public void export(HttpServletResponse response) throws IOException {
    //writeHeaderLine function invoked.
    writeHeaderLine();
    //writeDataLines function invoked.
    writeDataLines();
     //Returns a ServletOutputStream suitable for writing binary data in the response.
    //The servlet container does not encode the binary data.
    ServletOutputStream outputStream = response.getOutputStream();
    //Data written in Excel file.
    workbook.write(outputStream);
    //Closed the excel file.
    workbook.close();
    //Closed output stream.
    outputStream.close();

}
```