Spring Boot Generate Excel with React fetch blob.

To create a Spring Boot backend that allows you to fetch an Excel file and then fetch that file from a React frontend, you can follow these steps:

You can use Spring Initializer or your preferred method to create a new Spring Boot project with the necessary dependencies. You'll need dependencies like Spring Web, Spring Data JPA (if you want to interact with a database), and any other dependencies specific to your application.

2. **Create a Controller to Generate Excel:**

Create a controller that generates the Excel file. You can use a library like Apache POI to create the Excel file. Here's a basic example:

```java
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.ByteArrayOutputStream;
import java.io.IOException;

@RestController
public class ExcelController {

    @GetMapping("/download-excel")
    public ResponseEntity<byte[]> downloadExcel() throws IOException {
        // Create a new Excel workbook and sheet
        Workbook workbook = new XSSFWorkbook();
        Sheet sheet = workbook.createSheet("SampleSheet");

        // Create sample data (you can replace this with your own data)
        Row row = sheet.createRow(0);
        row.createCell(0).setCellValue("Name");
        row.createCell(1).setCellValue("Age");
        row = sheet.createRow(1);
        row.createCell(0).setCellValue("Meduim");
        row.createCell(1).setCellValue(30);

        // Write the workbook to a ByteArrayOutputStream
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        workbook.write(stream);

        // Set response headers
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.parseMediaType("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"));
        headers.setContentDispositionFormData("attachment", "sample.xlsx");

        return ResponseEntity.ok()
                .headers(headers)
                .body(stream.toByteArray());
    }
}
```

3. **Configure CORS (if needed):**

If your frontend is hosted on a different domain, configure CORS (Cross-Origin Resource Sharing) in your Spring Boot application to allow cross-origin requests.

4. **Run Your Spring Boot Application:**

Start your Spring Boot application. You should be able to access the Excel file at `http://localhost:8080/download-excel`

React Frontend:

- **Create a React Project:**

Create a new React project using Create React App or your preferred method.

2. **Fetch the Excel File:**

In your React component, you can use the `fetch` API to fetch the Excel file from your Spring Boot backend and convert it to a blob.

```
import React, { useEffect } from 'react';

function App() {
    const downloadExcel = () => {
        // Fetch the Excel file with .xls extension
        fetch('http://localhost:8080/download-excel-xls', {
            method: 'GET',
            responseType: 'blob', // This is not a standard option; you might need to handle it differently
        })
            .then((response) => response.blob())
            .then((blob) => {
                // Create a blob URL and create a link element to trigger the download
                const blobUrl = window.URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.style.display = 'none';
                a.href = blobUrl;
                a.download = 'sample.xls'; // Set the desired file name with .xls extension
                document.body.appendChild(a);
                a.click();
                window.URL.revokeObjectURL(blobUrl);
            })
            .catch((error) => {
                console.error('Error fetching Excel file:', error);
            });
    };

    useEffect(() => {
        downloadExcel();
    }, []);

    return (
        <div className="App">
            <h1>Fetch Excel File Example</h1>
        </div>
    );
}

export default App;
```

3. We fetch the Excel file from the server using the URL `http://localhost:8080/download-excel-xls`. Ensure that your Spring Boot backend has a new endpoint (`/download-excel-xls`) that generates and returns an Excel file with the `.xls` extension.

4. After fetching the blob, we create a blob URL using `window.URL.createObjectURL(blob)`.

5. We create a hidden `<a>` element and set its `href` to the blob URL, specify the `download` attribute with the desired file name including the `.xls` extension, and trigger a click event to initiate the download.

6. Finally, we revoke the blob URL to release resources.

Remember to update your Spring Boot backend to generate and serve the Excel file with the `.xls` extension when requested from the `/download-excel-xls` endpoint. You'll need to use a library like Apache POI with the `.xls` format for this purpose.