Week 1
CSIT 2024
github/TanPingZhi

pom.xml
• Add this dependency into the pom.xml file to use Apache Tika

```
<dependency>
    <groupId>org.apache.tika</groupId>
    <artifactId>tika-parsers</artifactId>
    <version>1.17</version>
</dependency>
```

index.html
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Upload PDF</title>
</head>
<body>
<h1>Upload a PDF file</h1>
<form method="POST" enctype="multipart/form-data" action="/upload">
    <input type="file" name="file" accept="application/pdf"/>
    <button type="submit">Upload</button>
</form>

<div>
    <h2>PDF Content</h2>
    <pre th:text="${pdfContent}"></pre>
</div>
</body>
</html>
```

• xlmns:th is a namespace declaration for Thymeleaf
  • th any attribute that starts with th: is a Thymeleaf attribute
  • Thymeleaf is a modern server-side Java template engine for both web and standalone environments.

PdfController.java
```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

@Controller
public class PdfController {

    @Autowired
    private PdfService pdfService;

    @GetMapping("/")
    public String index() {
        return "index";
    }

    @PostMapping("/upload")
    public String uploadPdf(@RequestParam("file") MultipartFile file, Model model) {
        try {
            String pdfContent = pdfService.parsePdf(file);
            model.addAttribute("pdfContent", pdfContent);
        } catch (IllegalArgumentException e) {
            model.addAttribute("pdfContent", e.getMessage());
        } catch (Exception e) {
            model.addAttribute("pdfContent", "Error reading PDF: " + e.getMessage());
        }
        return "index";
    }
}
```

• @Autowired annotation is used for automatic dependency injection
  • Spring automatically injects the Tika object through the @Bean. Spring registers this bean into the application context.
  • There is actually no need for the @Autowired annotation since the Tika object is not edited in the AppConfig.java file.
• @GetMapping and @PostMapping are used to map HTTP GET and POST requests to the specified methods.
• @Controller annotation is used to indicate that the class is a controller.
  • Controller classes are responsible for processing incoming requests and returning responses to the client.
• @RequestParam annotation is used to bind a web request parameter to a method parameter.
  • @RequestParam("file") binds to the name="file" in the index.html input type="file" name="file" accept="application/pdf"/
• model.addAttribute is used to add attributes to the model.
  • pdfContent variable is dynamically loaded into the index.html file.
  • when its value is loaded, the page will then load. Spring is sync
• MultipartFile is an interface to handle file uploads.
  • when user uploads the file, its converted to binary data and stored in the file variable.

PdfService.java
```
package com.example.demo;

import org.apache.tika.Tika;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

@Service
public class PdfService {

    @Autowired
    private Tika tika;

    public String parsePdf(MultipartFile file) throws Exception {
        if (!"application/pdf".equalsIgnoreCase(file.getContentType())) {
            throw new IllegalArgumentException("The uploaded file is not a PDF.");
        }
        return tika.parseToString(file.getInputStream());
    }
}
```

• Service classes are used to write business logic in a different layer, separated from the controller.
• Makes it easier to test the business logic.

PdfServiceTest.java
Snippet
```
@Test
void testParsePdfSuccess() throws Exception {
    MultipartFile file = mock(MultipartFile.class);
```

```
    when(file.getContentType()).thenReturn("application/pdf");
    when(file.getInputStream()).thenReturn(new ByteArrayInputStream("PDF content".getBytes()));
    when(tika.parseToString((InputStream) any())).thenReturn("Parsed PDF content");

    String result = pdfService.parsePdf(file);

    assertEquals("Parsed PDF content", result);
}
```

- org.mockito.Mockito is used to mock objects.
- mock() will intercept the function call of the objects so that we can control the return value.
- so we can intercept things like when(file.getContentType()).thenReturn("application/pdf");
  - This will return "application/pdf" when file.getContentType() is called.

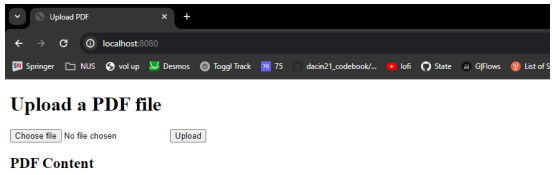
AppConfig.java

```
package com.example.demo;

import org.apache.tika.Tika;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public Tika tika() {
        return new Tika();
    }
}
```

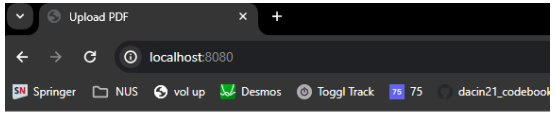
- This is class is needed since we are using the @Autowired annotation in the PdfController.java file.
- If we do not use the @Autowired annotation, we can remove this class.

Behavior

Original page



Uploading



Uploaded