フレームワーク基礎実習

11. ファイルアップロード

株式会社ジードライブ

今回学ぶこと

- ファイルをアップロードする方法
- ファイル操作
  - ファイルの取得と削除

ファイルアップロードの実装

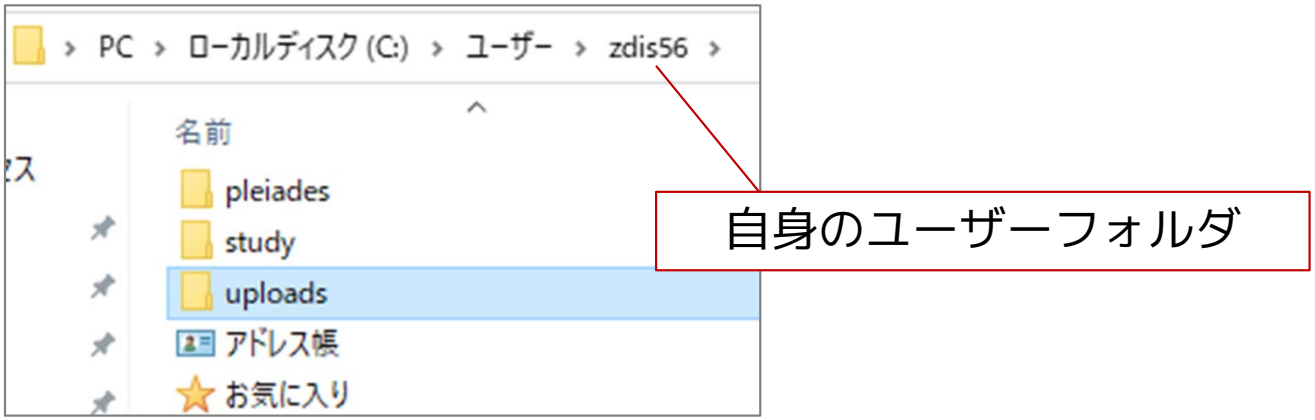
ファイルアップロード実装に必要な作業

- アップロード用ディレクトリの作成と有効化
- アップロード可能なファイルサイズの設定
- フォームの作成
- `MultipartFile`オブジェクトの操作

アップロード用ディレクトリの作成

- 任意の場所にアップロード用のディレクトリ(フォルダ)を作成する
  - ディレクトリ名は任意
  - アップロード先のディレクトリは、static内には配置できないので注意する

以降は、自身のユーザーフォルダ内に「uploads」というディレクトリを作成した場合の例を元に説明する



アップロード用ディレクトリの有効化

- WebMvcConfigurerを実装する設定ファイルを作成し、アップロード用ディレクトリを参照できるようにする
  - addResourceHandlers()をオーバーライドする

localhost:8080/uploads/ でユーザーフォルダ内のuploadsディレクトリを参照するための記述例

```
@Configuration
public class ApplicationConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/uploads/**")
            .addResourceLocations("file:///C:/Users/zdXXXX/uploads/");
    }

}
```

addResouceHandler(): 外部リソースフォルダへアクセス時のパスを指定
addResourceLocations()： 外部リソースフォルダーの追加

ファイルサイズの設定

- application.properties でアップロード可能なファイルサイズ等の設定を行うことができる

プロパティ	説明
spring.servlet.multipart.max-file-size	アップロード可能なファイルサイズの最大値（1ファイルあたりのサイズ） 初期値：1MB
spring.servlet.multipart.max-request-size	１回のリクエストで送信できるデータの最大値 初期値：10MB

設定例

```
spring.servlet.multipart.max-file-size=20MB
spring.servlet.multipart.max-request-size=100MB
```

フォームの作成

- formタグのenctype属性にmultipart/form-dataを設定する
- アップロード用のフォーム部品は、<input type="file">で作成する

```
<form action="" method="post" enctype="multipart/form-data">
    <p>イメージ画像：<input type="file" name="upfile">
    <input type="submit" value="アップロード">
</form>
```

name属性の代わりに
th:fieldを使用してもよい

MultipartFileオブジェクト

- アップロードされたファイルのデータは、MultipartFileオブジェクトとして取得することができる

MultipartFileインターフェースの主なメソッド

メソッド	説明
boolean isEmpty()	ファイルが空の場合trueを返す
String getName()	パラメータ名(フォーム部品のname属性値)を取得する
long getSize()	ファイルサイズを取得する（単位：byte)
String getContentType()	ファイルのコンテンツタイプを取得する
String getOriginalFilename()	ファイル名を取得する
void transferTo(File dest)	ファイルをディスクに書き込む(保存する)

MultipartFileオブジェクトの操作

コントローラーメソッドの記述例

```
@PostMapping("/upload")
public String post(@RequestParam MultipartFile upfile)
                                                    throws IOException {
    // ファイルが選択されている場合の処理
    if(!upfile.isEmpty()) {
        // アップロードされたファイルの名前を取得
        String fileName = upfile.getOriginalFilename();
        // 保存先のファイルパスの設定(uploadsフォルダ内に保存する)
        File dest = new File("C:/Users/zdXXXX/uploads/" + fileName);
        // ファイルを保存する
        upfile.transferTo(dest);
    }

    ...
```

ファイル一覧の取得

- あるディレクトリ内のファイルは、File#listFiles()メソッドで取得することができる

例：アップロードされたファイルを取得するためのコントローラーメソッド

```
@GetMapping("/filelist")
public String showFileList(Model model) {
    // アップロードされているファイルのリストの取得
    File uploadsDirectory = new File("C:/Users/zdxxxx/uploads");
    File[] fileList = uploadsDirectory.listFiles();

    // ファイルリストからファイル名のリストを作成し、Modelに保存
    List<String> fileNames =
            Arrays.stream(fileList).map(file -> file.getName()).toList();
    model.addAttribute("fileNames", fileNames);
    return "fileList";
}
```

ファイルの削除

- ファイルはFiles#delete()メソッドで削除することができる

例：削除用のコントローラーメソッド

```
@GetMapping("/delete/{fileName}")
public String delete(@PathVariable String fileName)
                                              throws IOException {
   Path path = Paths.get("C:/Users/zdxxxx/uploads/" + fileName);
   Files.delete(path);
   return "redirect:/filelist";
}
```

練習

- 練習11-1
- 練習11-2