## Objective

The objective of this workshop is to write a simple address book. The information from the address book will be saved in a local filesystem as text file using Java IO package.

## Setup

a. Create and clone a remote Git repository
b. Generate a SpringBoot application from Spring Initializr. Add the following dependencies
   i. Spring Boot Dev Tools
   ii. Spring Web
   iii. Thymeleaf
c. Unpack the generated SpringBoot application in the Git repository

## Workshop

### Task 1

The application should accept a command line parameter call `--dataDir`. This options references a directory on your local computer eg.

```
java myapp.jar --dataDir /opt/tmp/data
```

Refers to the directory `/opt/tmp/data`

If the directory does not exist, the application should create it.

If the `--dataDir` option is not specified, print an error message and stop.

See Javadocs on `java.nio.files.Paths` and `java.nio.files.Files`.

**Task 2**

Write a HTML form that collects the following information: name, email , phone number and date of birth.

Validate the above information according to the following

| Property | Constraints |
|---|---|
| Name | Mandatory, the length must be between 3 and 64 |
| Email | Mandatory, valid email |
| Phone number | Mandatory, must contain at least 7 digits |
| Date of birth | Mandatory, must be in the past. Cannot be younger than 10 years old and older than 100 years |

This information should be sent to `/contact` resources with the HTTP `POST` method.

**Task 3**

The controller mapped to `/contact` process the data according to the following steps

a. Randomly generate an 8 character long hex string (eg `abcd1234`); this hex string will be used as the id for the data
b. Create a file with the above generated hex string in the directory specified by `--dataDir` option eg. you will create a file in `/opt/tmp/data/abcd1234`
c. Write the data into the file (`abcd1234`) as text (UTF-8) one field per line

Once the controller has completed the above steps, the controller should return the 'created' HTTP code with an appropriate message.

Important: All methods that manages the data directory should be handle by a single class and not in the controller. Call this class `Contacts`.

**Task 4**

Write a controller to handle a `GET` to `/contact/<id>` where `<id>` is 8 character long hex digit.

The controller will look into the data directory for a file with the corresponding `<id>` in the resource. Display the contents of the file in a HTML document.

If the `<id>` does not exist in the data directory, then return a not found status code with an appropriate message.

**Task 5**

Write a HTML page called contacts.html to generate a list of HTML links for all your contacts from the `dataDir` directory. The following is an example of the HTML link that should be generated

```
<a href=”/contact/abcd1234”> Fred </a>
```

where `abcd1234` is Fred’s id.

Note: we will not be deploying this application to Railway because applications deployed to Railway cannot access local the file system

## Submission

When you have completed the workshop, commit and push your code to your Github repository.