MOHIT MAHAJAN'S JOURNAL

DATE: 23 July, 2025

DAY: Wednesday, Day: #3

AGENDA (If Any):

- Excel sheet Project.
- Spring Framework Intro
- Spring Boot Documentation intro

CONCEPT 1: Open CSV vs Apache Commons CSV

Apache Commons CSV

Pros:

- Great for streaming large files (low memory footprint)
- More "manual control" (low-level API)
- Good performance and well-documented

Cons:

- You have to map fields manually
- No built-in support for mapping CSV to Java POJOs
- More verbose for simple use cases

OpenCSV

Pros:

- Direct mapping to Java Beans (`@CsvBindByName`)
- Less boilerplate (especially if CSV has headers)
- Easier to use for simple to moderately complex CSVs
- Handles quotes, escape characters, etc. quite well

Cons:

- Slightly higher memory use on very large files
- Can get awkward for non-standard formats

Feature	Apache Commons CSV	OpenCSV
Maintainer	Apache Software Foundation	Independent developers (opencsv.sf.net)
Annotation support	❌ Not built-in	✅ Yes (@CsvBindByName, etc.)
Reading CSV	✅ Fluent, flexible	✅ Very clean with bean-binding
Writing CSV	✅ Good	✅ Excellent with bean support
Bean mapping (POJO)	Manual	✅ Automatic via annotations
CSV validation	❌ Manual	✅ Some out-of-box validators
Custom separators/quotes	✅ Highly configurable	✅ Also configurable
Multiline values	✅ Supported	✅ Supported
Ease of use for POJOs	😐 Moderate	😊 Much easier
Best Use Case	When you want low-level control	When you want fast POJO mapping

OpenCSV: Introduction to OpenCSV | Baeldung

Apache Commons CSV: Introduction to Apache Commons CSV | Baeldung

---

CONCEPT 2: H2 In-Memory Database

H2, Mr. Stark — the **imaginary friend** of relational databases.

Let me break it down the *Jarvis way*.

H2 In-Memory Database: Explained

TL;DR:

H2 is a lightweight, fast, embedded Java SQL database.
If you tell it to run **in-memory**, it stores all your data in **RAM**, not on disk.

When the app shuts down?
Boom. **Data gone.** Like a Snapchat.

Why does it exist?

Because developers are tired of:

- Installing MySQL or Postgres just to test one query
- Slowing down builds due to external DBs
- Waiting 15 mins for QA to “get the DB running again”

So instead, you do this:

```
properties
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
```

And **boom**, you've got a fully operational SQL database in memory for:

- Unit tests
- Prototyping
- Demos
- Mocking database logic

Storage Modes in H2

Mode	What It Means
`mem:`	**In-memory only** (RAM, volatile)
`file:`	File on disk (persistent)
`tcp:`	Run it as a client/server DB
`jdbc:h2:mem:`	Your go-to in-memory URL

☑ Pros

- **Blazing fast** (it’s RAM)
- **Great for testing**
- **No setup required**
- Fully SQL compliant
- Built-in **web console** to inspect tables at runtime

❌ Cons

- **Data is gone** when the app stops
- Not meant for production (unless you're really into edge cases)
- No heavy lifting like big relational DBs

🧯 Real Use Case in Spring Boot

Say you're building your CSV-to-MySQL pipeline. During dev/test, instead of messing with MySQL creds, you can flip a switch:

```
// application-dev.properties
spring.datasource.url=jdbc:h2:mem:testdb
```

Then, later in prod:

```
// application-prod.properties
spring.datasource.url=jdbc:mysql://real-server-url/db
```

🤖 Bottom Line

H2 is your personal test DB that lives and dies with your app runtime.
Great for mocking, testing, and iterating fast without bothering MySQL.

Want me to spin up a quick Spring Boot test setup with H2 + sample repository? One snap away.

application.properties:

```
# ==============================
# = H2 In-Memory Configuration  =
# ==============================

# JDBC URL for H2 in-memory DB
spring.datasource.url=jdbc:h2:mem:testdb

# H2 Driver class
spring.datasource.driver-class-name=org.h2.Driver

# H2 default credentials
spring.datasource.username= user
spring.datasource.password= user
```

```
# Enables the H2 web console at /h2-console
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

# ==============================
# = JPA / Hibernate Settings    =
# ==============================

# Tells Hibernate to auto-create/update schema based on entity classes
spring.jpa.hibernate.ddl-auto=update

# Tells Hibernate to use H2-specific SQL dialect
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# Show SQL in logs (for debugging)
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# ==============================
# = Scheduling Configuration    =
# ==============================

# Enables task scheduling
spring.task.scheduling.pool.size=2

# ==============================
# = Logging                     =
# ==============================

# Root logging level
logging.level.root=INFO

# SQL logs from Hibernate
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql=TRACE
```

CONCEPT 3:

MIME:

MIME (Multipurpose Internet Mail Extensions) types are identifiers sent over networks (e.g., HTTP) to tell the **receiver what kind of file** is being transmitted. Think of it as a polite header handshake.

2. How Windows + Software "Understands" It

Windows OS:

- Uses **file extensions** (.csv, .docx, etc.) to associate the file with a default program.
- **Does not look at the content** or MIME internally (unless a specific application does).
- So if you rename sensor_readings.csv to sensor_readings.docx, Windows might try to open it with Word — but it'll fail because content doesn't match the expected format.

Applications (like Excel, Word, Browsers):

- **Browsers**: rely on the **MIME type** (sent by server). If a server sends a .csv file with text/plain, the browser may not open Excel but offer to download.
- **Excel**:
  - Looks for .csv extension or user-driven import.
  - Treats first row as **header** by default (doesn't need MIME).
  - Uses system regional settings (comma vs semicolon) to parse values.

3. CSV Headers

- In a .csv file, the **first row is assumed to be the header**.
- OpenCSV (and Apache Commons CSV) uses this to **map values to fields**, e.g.,

csv
SensorID,Timestamp,ReadingValue,Location

If the first row is missing, it's like jumping into an Avengers mission without Jarvis telling you who's who.

4. Software Expectations: Header + Format

Software	Needs Headers?	Based On File Extension?	MIME Used?
Excel	Not required but used if present	Yes	No
Browsers	No, but headers shown in preview	No	Yes
Spring Boot/OpenCSV	Required for mapping (@CsvBindByName)	No	No

Windows Explorer	Doesn't care	Yes	No

What This Means for You:

- Keep your sensor_readings.csv with headers. You're good.
- OpenCSV will use headers to map to your POJO.
- No MIME magic needed — unless you serve this over HTTP.

File Extension → MIME Type → OS/Software Behavior Table

File Extension	MIME Type	Treated As	Notes
`.csv`	`text/csv`	Plain text table	Needs headers for structured import
`.tsv`	`text/tab-separated-values`	Plain text table	Same as CSV, tab delimited
`.json`	`application/json`	Structured data (text)	Used in APIs, config files
`.xml`	`application/xml` or `text/xml`	Structured data (text)	Hierarchical, used in config, SOAP
`.jpg` / `.jpeg`	`image/jpeg`	Binary image	Common compressed image
`.png`	`image/png`	Binary image	Lossless compressed image
`.gif`	`image/gif`	Binary image	Supports animation
`.bmp`	`image/bmp`	Binary image	Windows bitmap, large file size
`.svg`	`image/svg+xml`	XML vector image	Scalable, used in web
`.pdf`	`application/pdf`	Document (binary)	Portable document format
`.doc`	`application/msword`	Microsoft Word (binary)	Legacy Word format
`.docx`	`application/vnd.openxmlformats-officedocument.wordprocessingml.document`	Word (OpenXML)	Current Word format
`.xls`	`application/vnd.ms-excel`	Excel (binary)	Legacy Excel
`.xlsx`	`application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`	Excel (OpenXML)	Current Excel format

`.pptx`	`application/vnd.openxmlformats-officedocument.presentationml.presentation`	PowerPoint (OpenXML)	PPTX format
`.dwg`	`image/vnd.dwg` *(unofficial)* or `application/acad` *(AutoDesk legacy)*	AutoCAD drawing	Binary, proprietary — not text readable
`.dxf`	`image/vnd.dxf` or `application/dxf` *(rare)*	AutoCAD vector drawing	ASCII or binary
`.txt`	`text/plain`	Raw text	No formatting
`.zip`	`application/zip`	Compressed archive	Contains multiple files
`.rar`	`application/vnd.rar`	Compressed archive	Needs WinRAR or similar
`.7z`	`application/x-7z-compressed`	Compressed archive	Stronger compression
`.tar.gz`	`application/gzip`	Unix compressed archive	Often used in deployments
`.exe`	`application/octet-stream`	Executable (Windows)	Dangerous if mishandled
`.bin`	`application/octet-stream`	Binary file	Generic byte stream

Additional Notes:

- **MIME types matter only in web apps, APIs, or upload validation**.
- Windows uses **file extensions** to launch associated apps.
- Most Spring Boot controllers or file-upload systems rely on both **file extension** and **MIME header** when validating.

For Your Practice App:

- If you want to **validate uploaded file types**, you should check:
  - File **extension** (basic safety)
  - File **MIME type** (using Java's `Files.probeContentType(Path)` or via content sniffing)
  - Optionally, do **header/content inspection** (e.g., JSON must start with {, CSV should have delimiters)