**Movie Application**- Java 8, Spring Boot, and Pagination

* Sorted
* Limit
* Pagination

**Github**: https://github.com/djrabi007/springboot-web-swagger-chatgpt

**Author**: Rabi

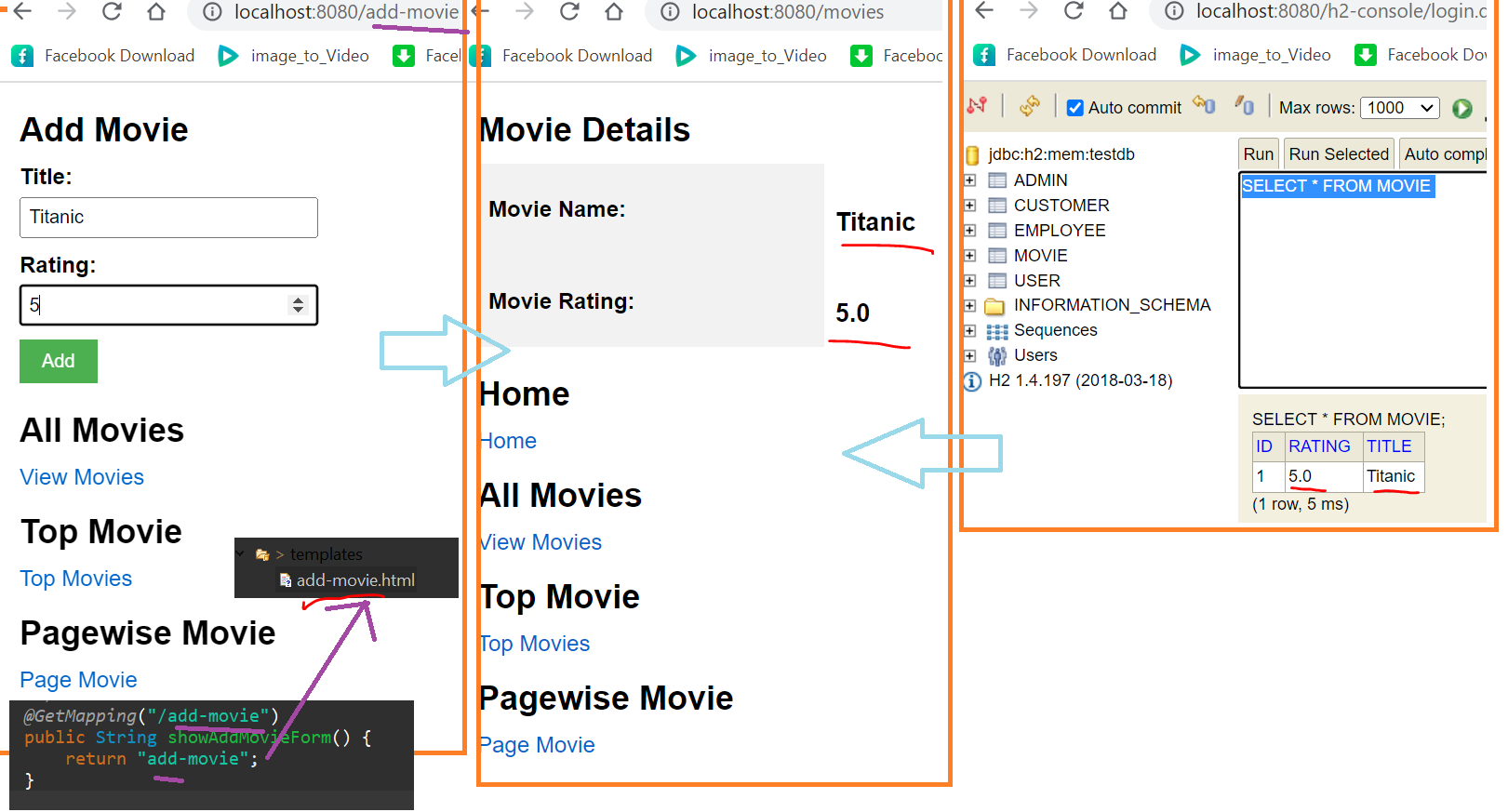
**Date**: 14th Juy,2023



1. **Introduction:**

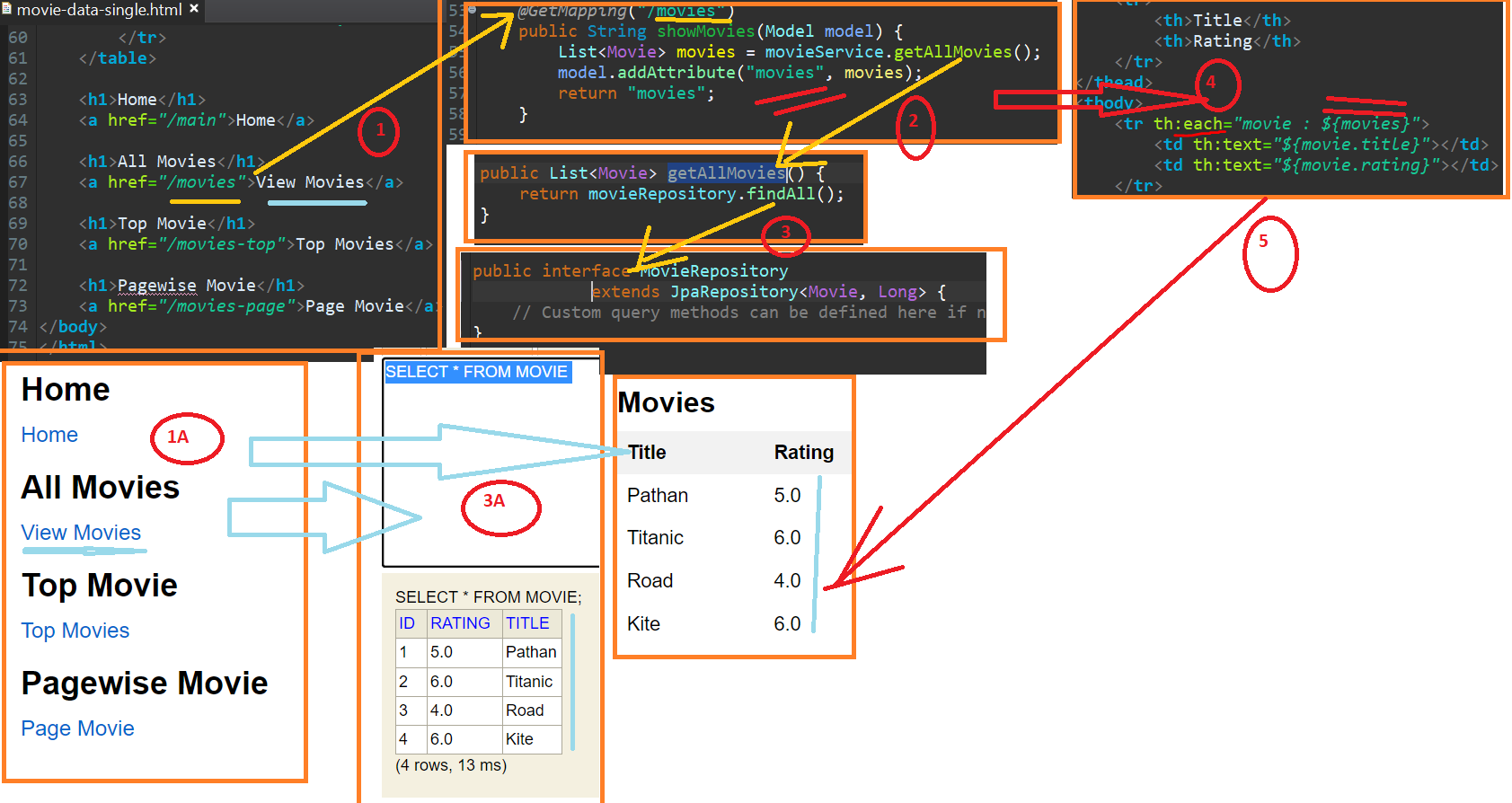
In this technical presentation, we will explore the implementation of Java 8 features, Spring Boot, and Pagination in a **movie** application. Our aim is to demonstrate the seamless integration of these technologies and **showcase the capabilities** they bring to the application.

1. **Add Movie into JPA (in-memory H2 Database)**
   1. ADD add-movie.html ==>POST(/movies)==>movie-data.html (show single entry)
      1. we will perform a POST request ("/movies") to add a new movie entry.
      2. Upon successful addition, the "movie-data.html" template will display the details of the newly added movie.
   2. GET(movies/movies-top)==>movies.html(show the list of movies)
      1. list of all movies in the "movies.html" template.
   3. GET(movies-page)==>movies-page.html(next,previous)
      1. **Pagination**: To navigate through the list of movies, we will implement pagination using the "movies-page.html" template.
      2. This will provide options for moving to the **next** or **previous** pages of movies.
   4. **JPA** – using **save**() function





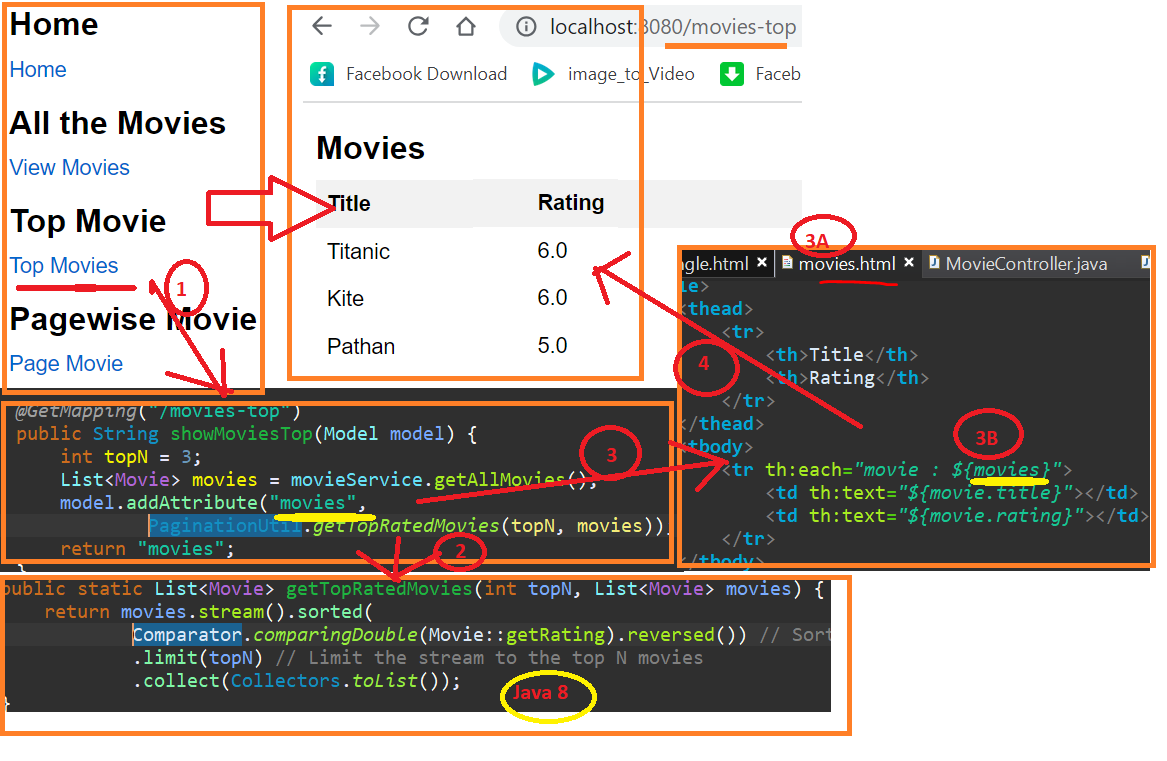
1. **Retrieving/GET Movie Names using Thymeleaf**: **(th:each of Thymeleaf )**
   1. **All Movies in H2 database**- In this section, we will focus on retrieving and displaying all movie names in the application.
   2. **Thymeleaf**- We will utilize the power of Thymeleaf's "**th:each**" directive to iterate over the movie names and present them in an appropriate format.
   3. **JPA**- using **findAll**()

****

1. **Retrieving Top 3 (hardcoded) Movie Names using Java 8 (th:each of Thymeleaf )**

We have used following function of Java8

* Sorted() , reversed(),comparingDouble()
* Limit()
  + we will explore Java 8 features to retrieve and display the top three movie names.
  + We will demonstrate the usage of Java 8 functions such as "**sorted**()", "**reversed**()", "**comparingDouble**()", and "**limit**()" to *accomplish this task effectively.*

****

1. **Pagination implementation using SpringBoot ,Thymeleaf and Java8**

**boxed() convert int[] to List<Integer>**

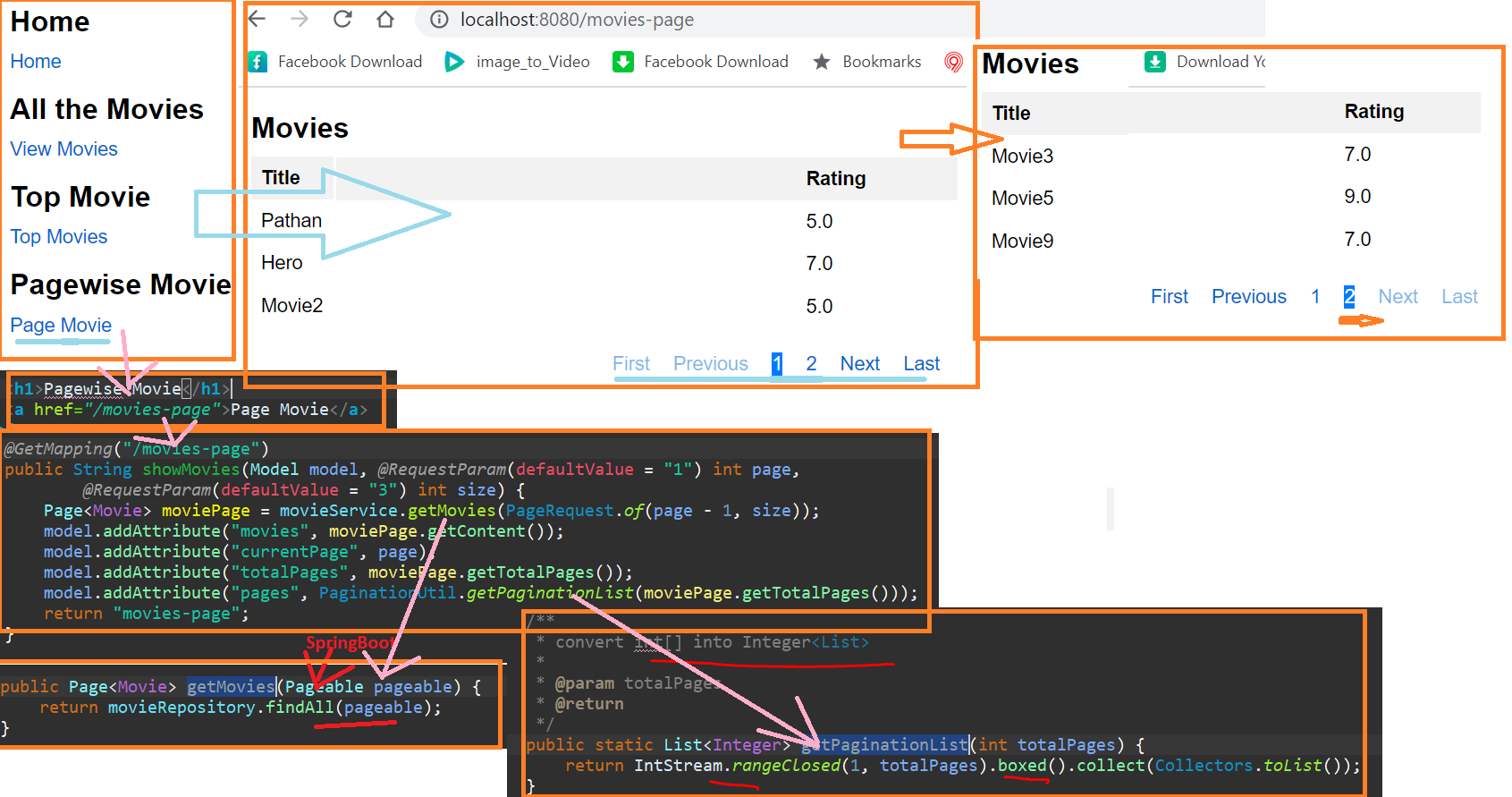
**Intstream.rangeClosed() of Java8**

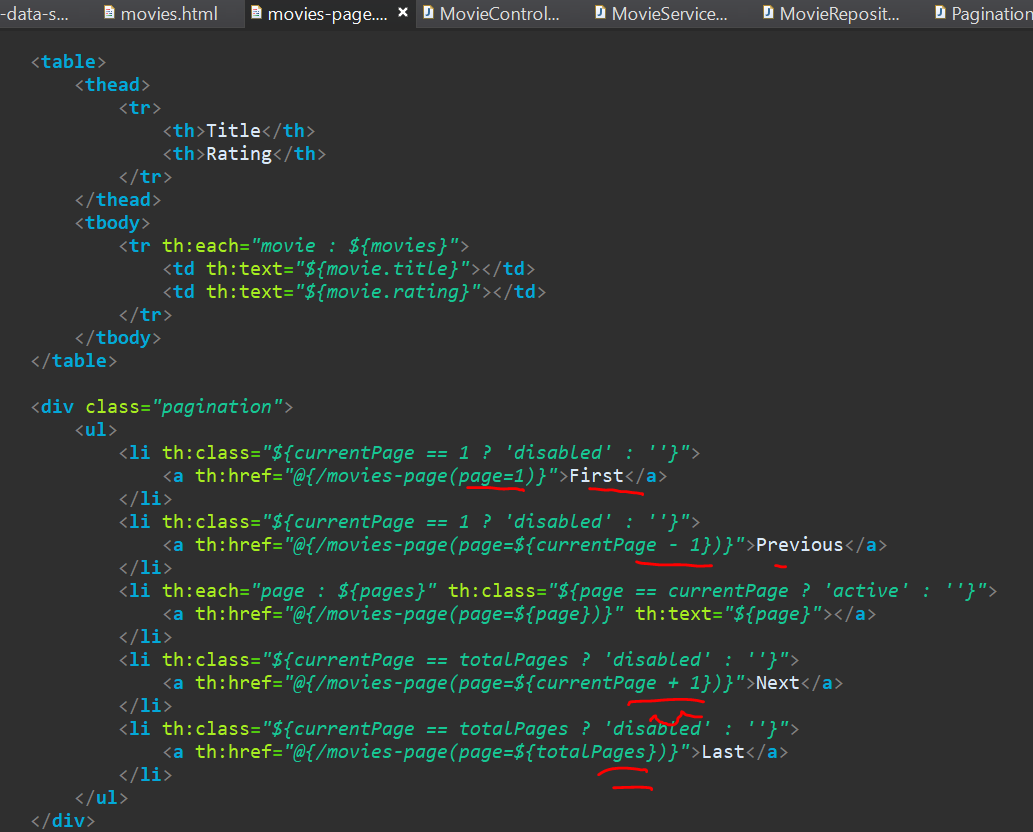
**findAll(Pageable) of JPA**

5.1. **Conversion to List:** We will use the "boxed()" function to convert an int[] to a List<Integer>, allowing us to work with a more **flexible data structure**.

5.2. **Utilizing Java 8's IntStream**: By utilizing "IntStream.rangeClosed()", we will generate a **stream of integers representing the available page numbers**.

5.3. **Leveraging JPA's "findAll(Pageable)**": To retrieve paginated movie data, we will leverage the "findAll(Pageable)" method provided by JPA. This will enable us to fetch and **display a specific range** of movies based on the chosen page.

****



**6. Conclusion**:

* + In conclusion, this technical presentation has explored the integration of Java 8 features, Spring Boot, and Pagination in a movie application.
  + We have highlighted the functionalities of **adding** movies, **retrieving** (all and Top 5) movie names, implementing **pagination**, and utilizing the power of Java 8 throughout the development process.
  + By combining these technologies effectively, we enhance the functionality and user experience of our movie application.