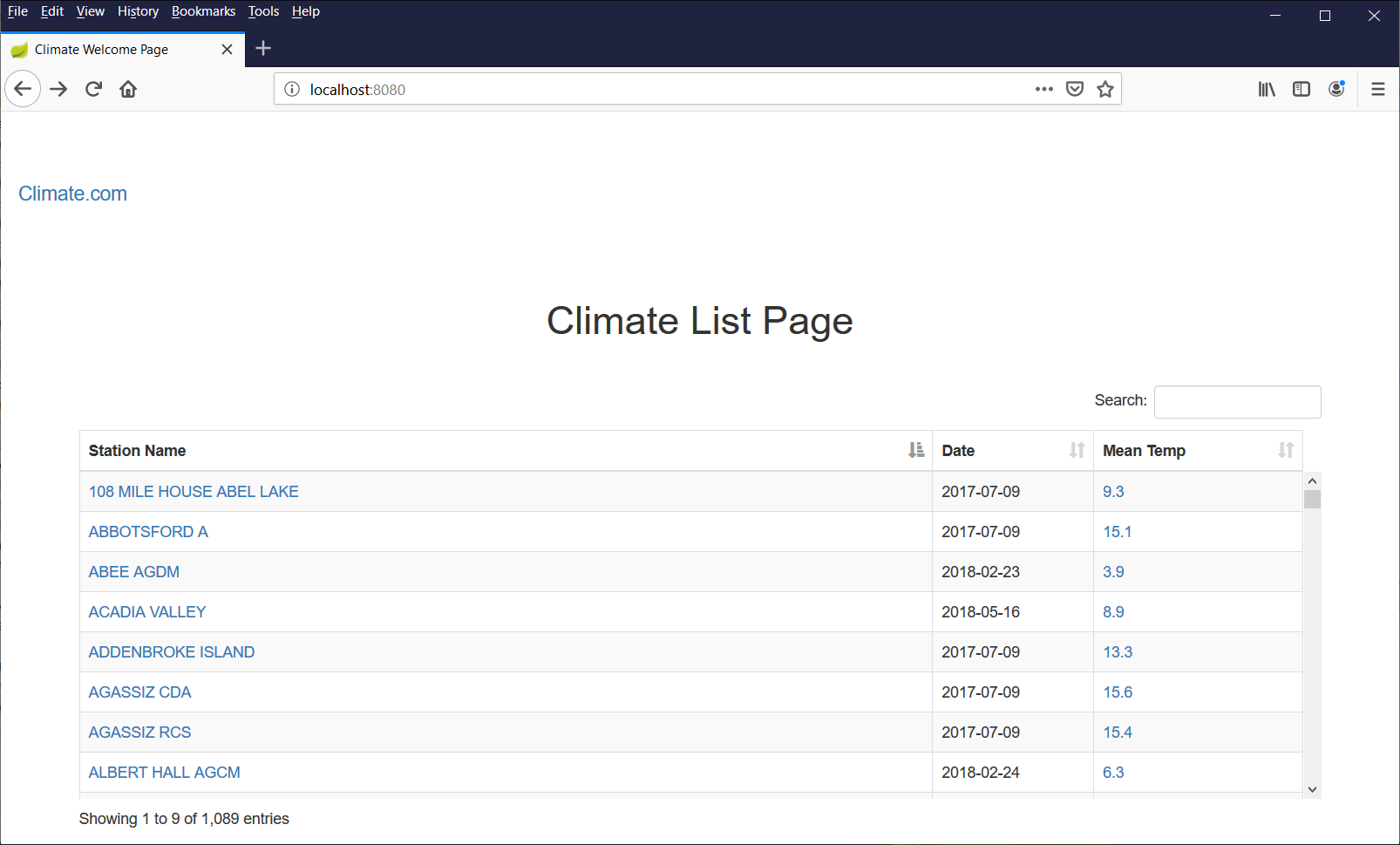
**Climate**

**System configs:**

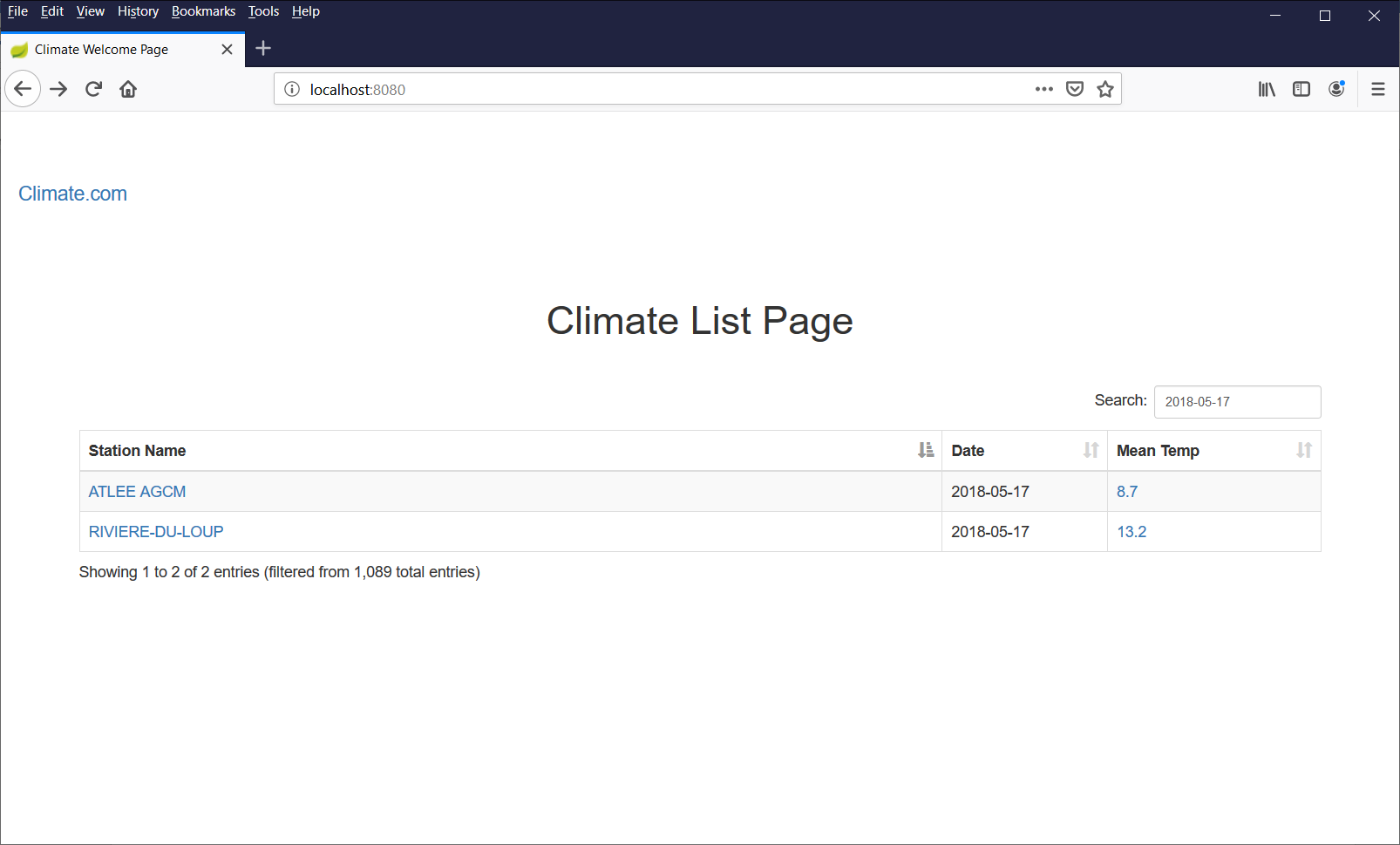
* Use Spring Boot: spring boot 2.1.2
* Use Spring, Spring MVC: spring framework 5.1.4
* Use Thymeleaf: thymeleaf 3.0.11
* Use of a logging technology: log4j2
* Use Junit/Mockito: Junit 4.12
* Use CSS and Javascript: Jquery
* Use Ant/Maven/Gradle to build/test/run your project: maven
* Server: embed Tomcat

**Main landing page:**



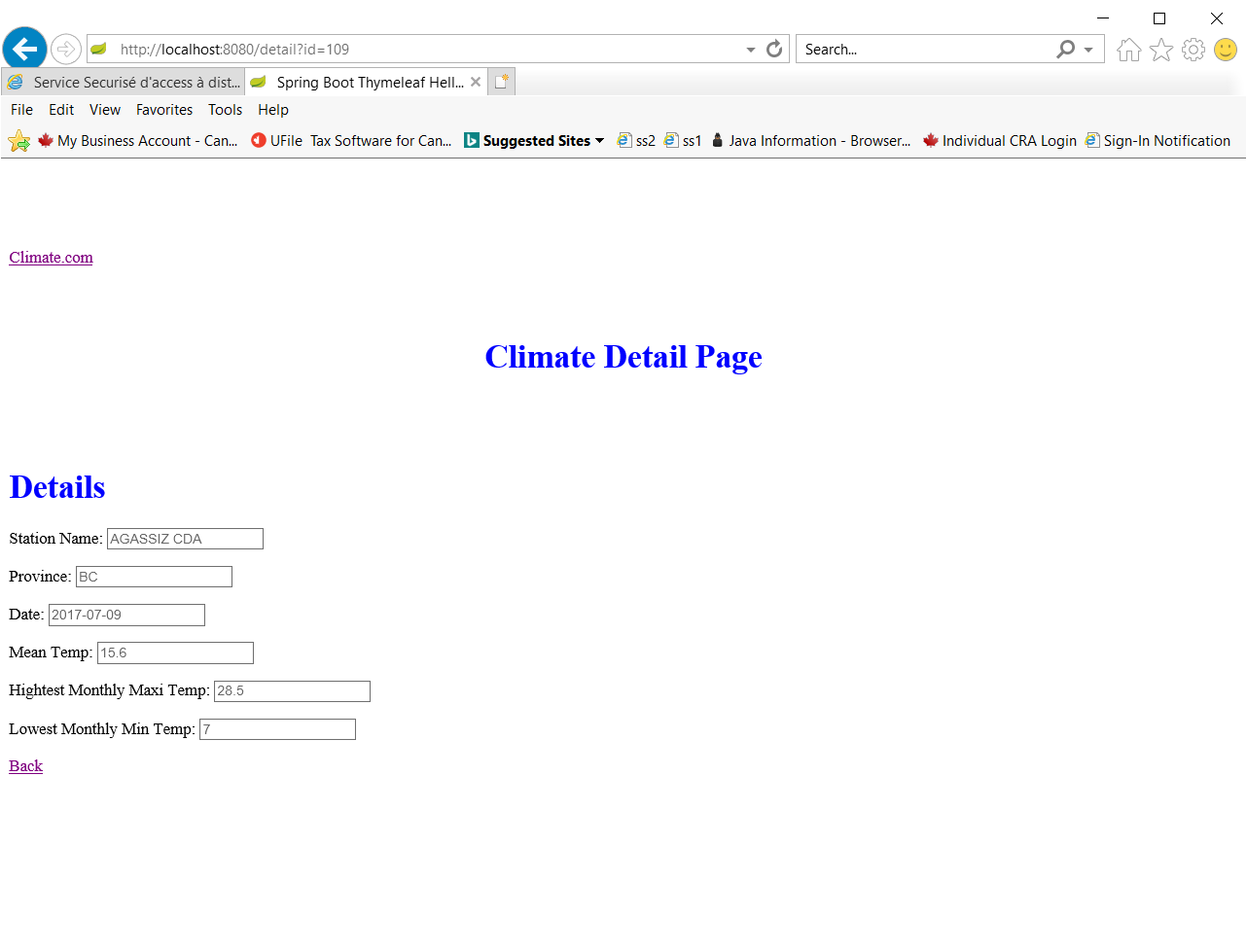
Using JQuery datatable decorate the jsp table, and provide sorting on each columns, scroller, table row striped …etc. The Station Name and Men Temp are hyperlinks to navigate to the detail page

**Filter:**



Using JQuery datatable, provide the filter (search box). User can input the date to narrow down the date range. It can also search on the Station Name and Mean Temp.

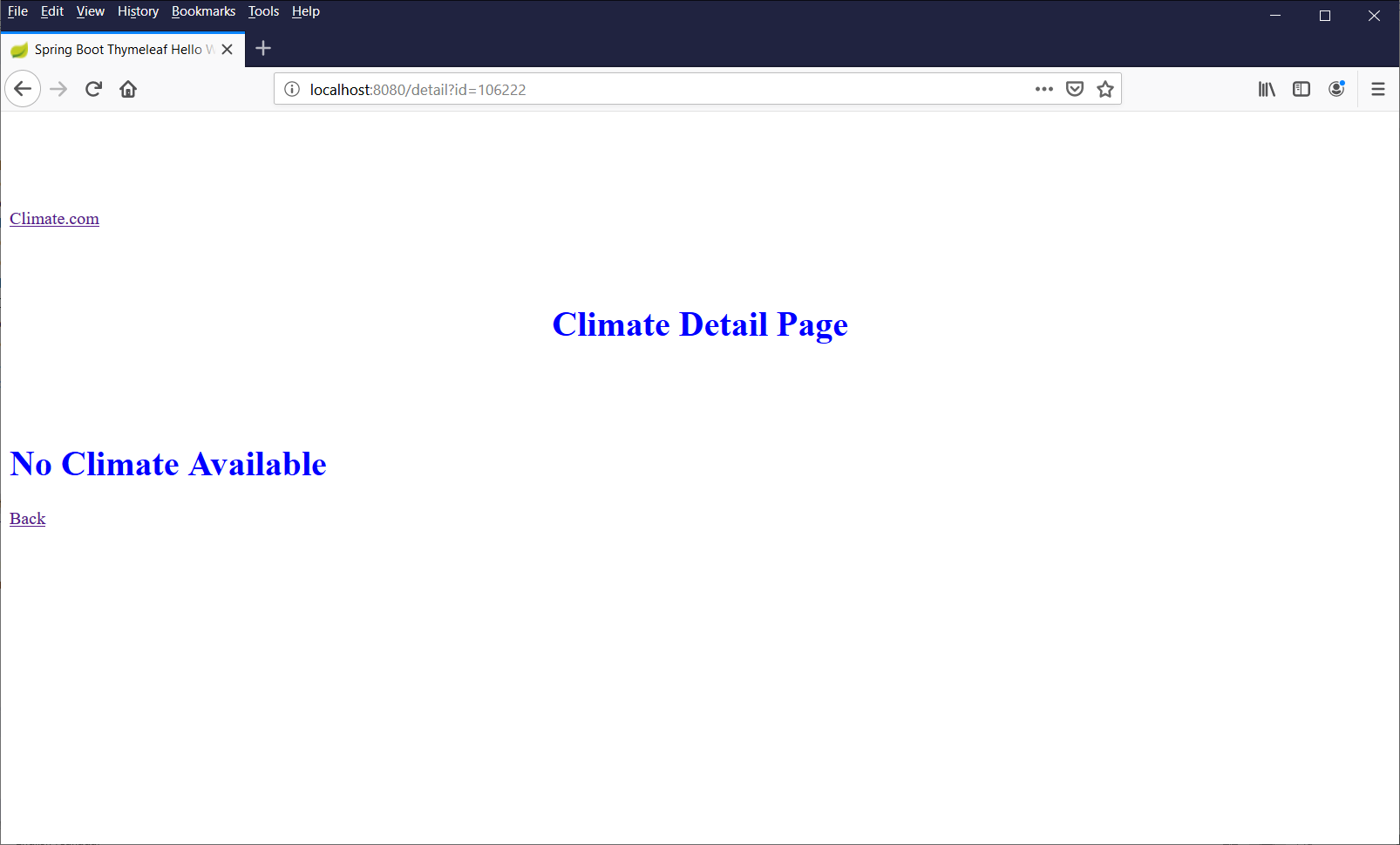
**Details:**



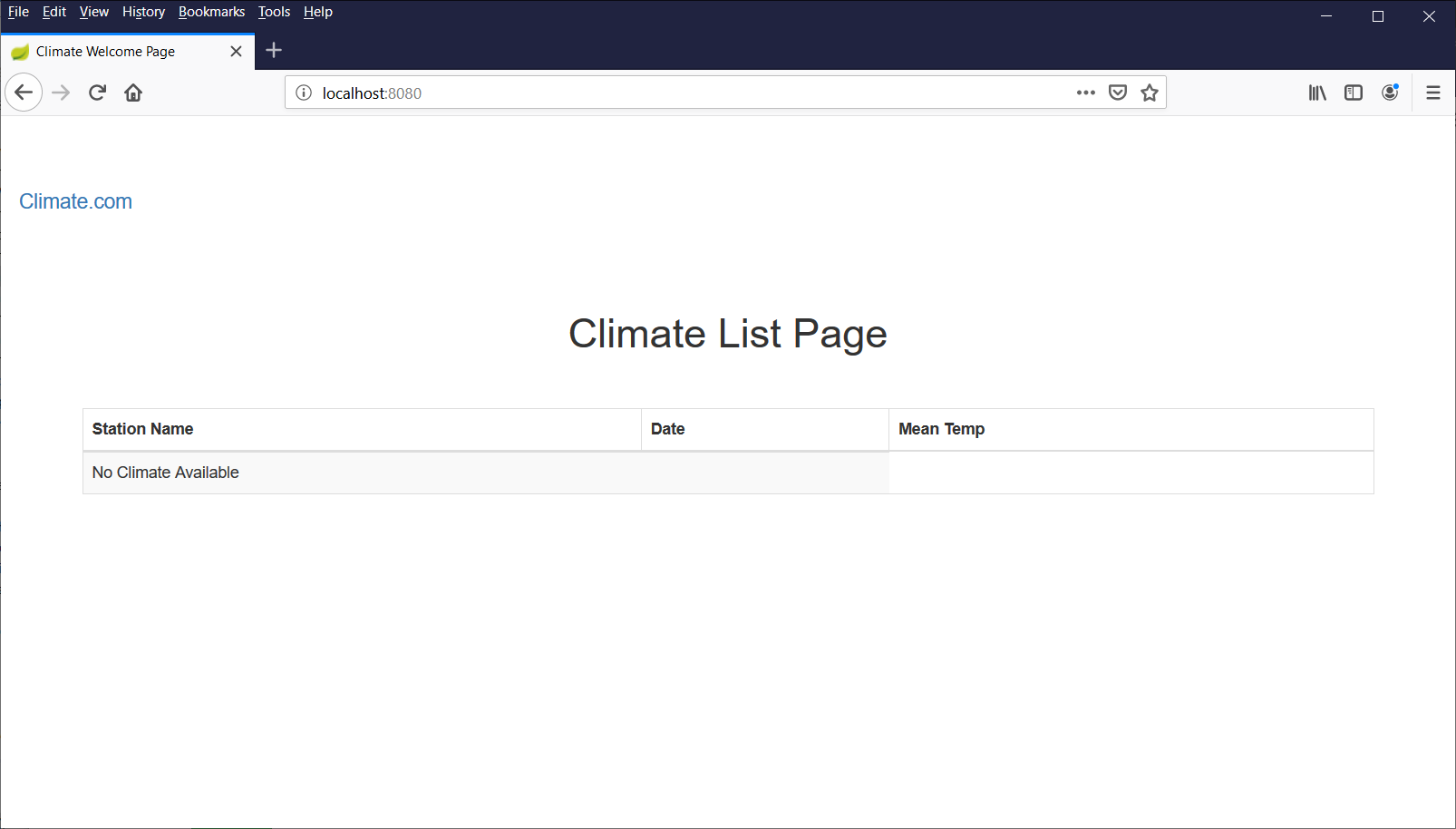
User clicks the “Station Name” or “Mean Temp” hyperlink on the list page to bring the detail page for the specific record.

Click the “back” to navigate back the main page.

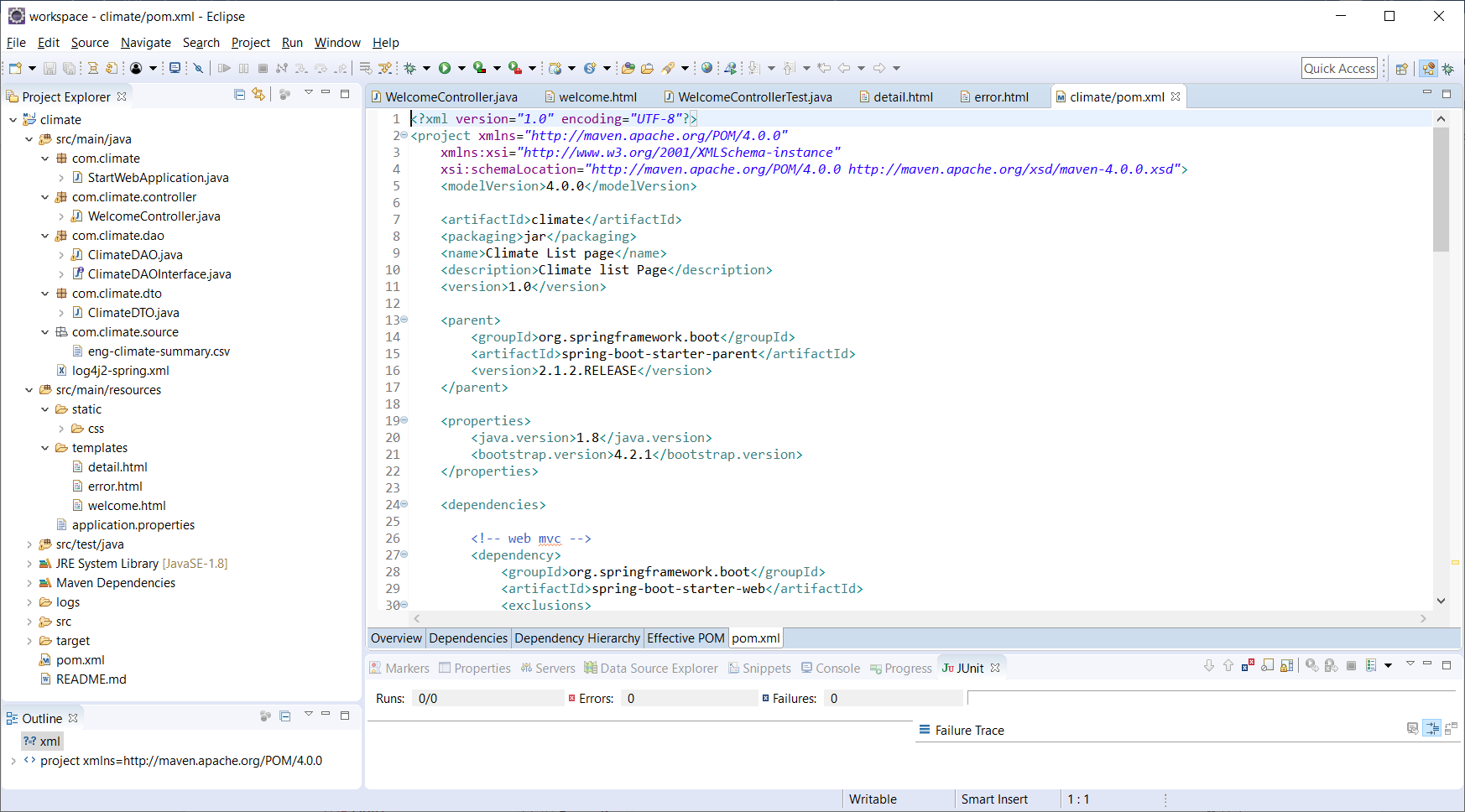
Error page:



No records:



Project Structure:



StartWebApplicaton.java: Spring boot startup class.

WelcomeController.java: Spring MVC controller

ClimateDAO.java: This class respond to read the csv file and convert to ClimateDTO. It did some simple validation to filter the bad records. I just use string parsing. If time permitted, we should use some comprehensive 3rd party CSV parser (like openCSV).

ClimateDTO.java: data model for the climate csv data model.

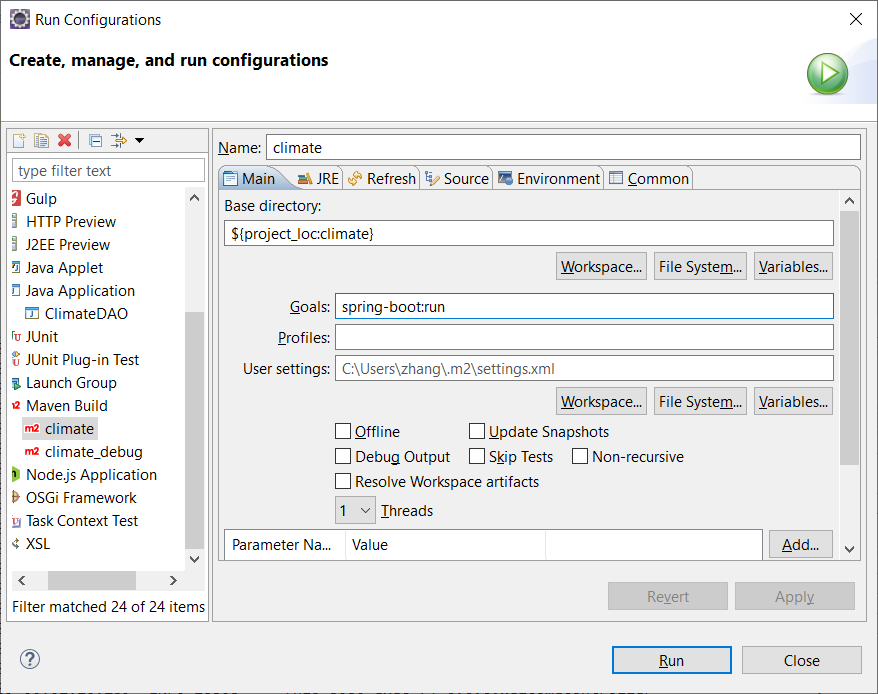
Log4j2-spring.xml: logging configuration.

Welcome.html: main landing page for climate to list the climate records.

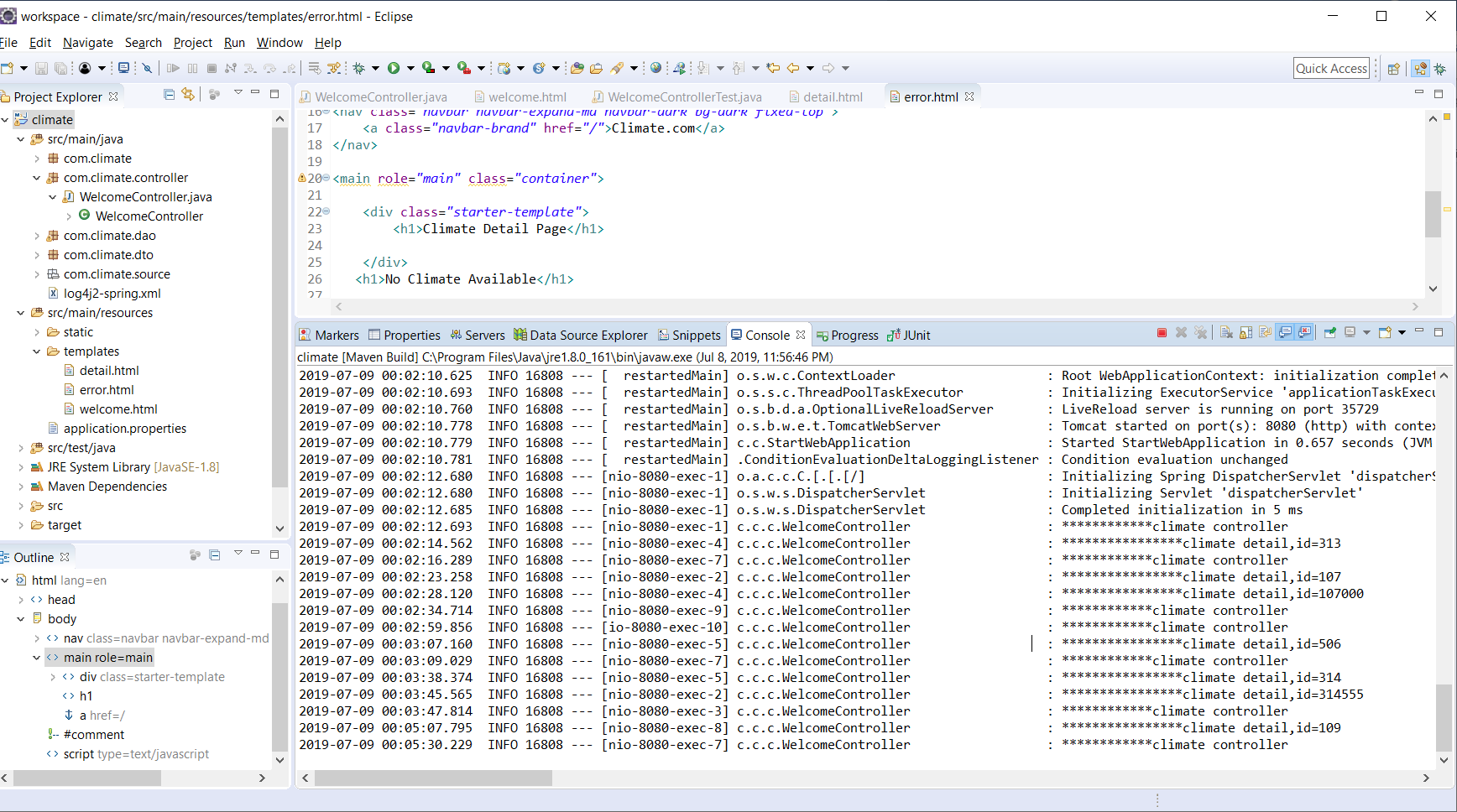
Detail.html: show the detail record in form format

Error.html: error page if validation failed on server side.

Run climate project:



Logging:



Test cases:

