My Weater Application

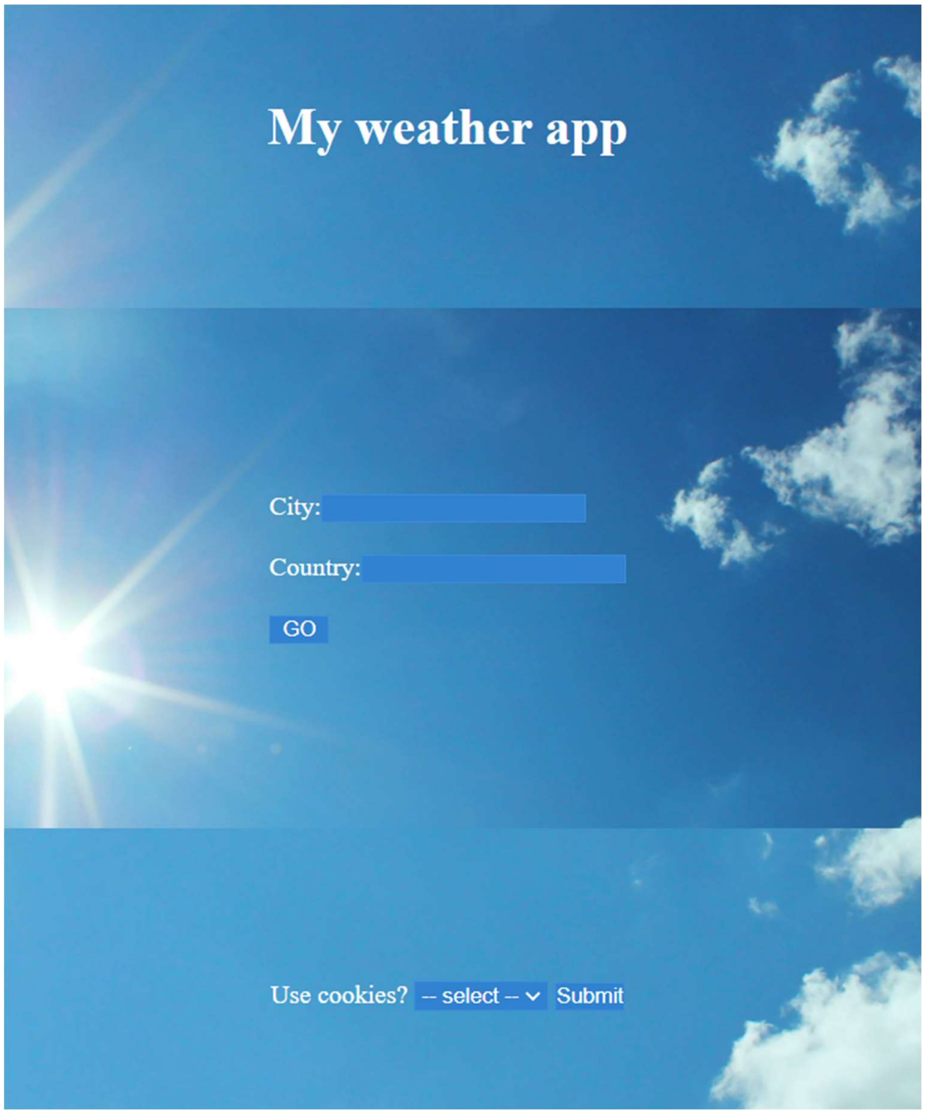


Table of Contents

1 Background and purpose

This is the final project in the course Webdevelopment with Java.
The project aims to develop an app that receives weather data from an API, handles the data, and shows it on a website.

As an API, open weather map is used (https://openweathermap.org/).

The purpose of the project is to show how to use an API, interpret the result and show it on a webpage. Furthermore, the code should follow the MVC structure and interaction between servlets, java beans and other java classes must take place.

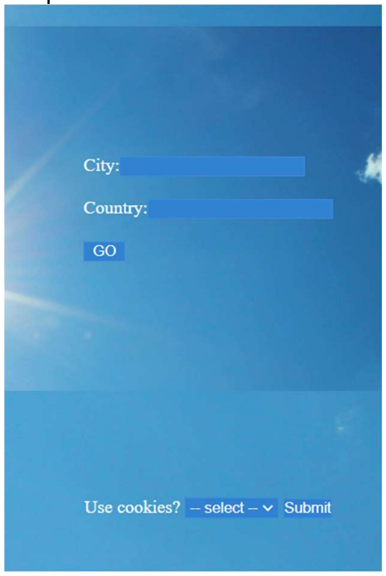
2 Requirements

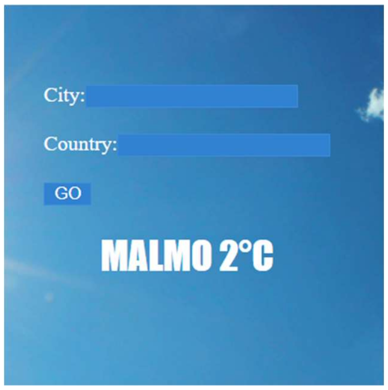
Following requirements must be met to fulfil the task:

- MVC structured code
- Mix JSPs, Servlets, Beans and other Java classes
- CSS and HTML
- Use cookies to show at least 5 former searches
- Variables, methods and classes named logically, and code well commented
- User must be able to decide if cookies are used or not
- The user must be able to search for a city in a certain country
- Each search must show the weather in that city
- Search result must include clouds, date and one more value from the API

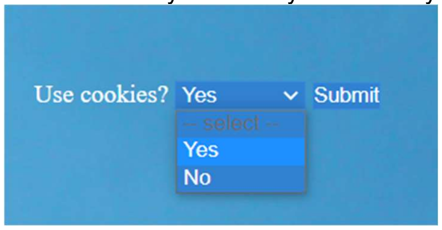
3 How to use the app

From a user point of view the app consists of two pages, the index.jsp page and the cookieWeatherPage.jsp.
If the user selects to not use cookies, in the footer, the user will stay on index.jsp and will only be able to see the search result of the current search. Furthermore, the user will only see the temperature.

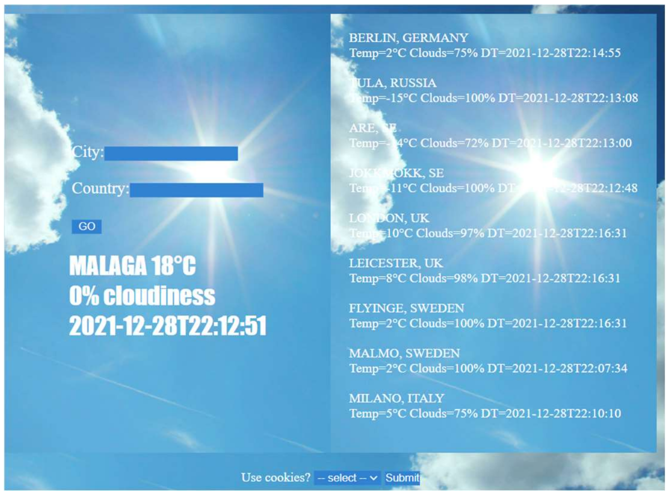




If the user decides to accept the use of cookies, he or she is directed to cookieWeatherPage.jsp and all searches are stored in a cookie.
The cookie only stores city and country.



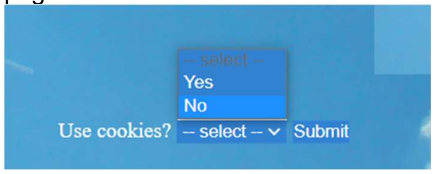
On the cookieWeatherPage.jsp there are two areas, the left area shows the current search result, and the right area shows the last search results in a list.
If the user manually goes to the index.jsp page, he or she will be redirected to cookieWeatherPage.jsp.
The cookiesearch will take place and populate the right hand list.



```
MALAGA 18°C
0% cloudiness
2021-12-28T22:12:51
```

```
BERLIN, GERMANY
Temp=2°C Clouds=75% DT=2021-12-28T22:14:55

TULA, RUSSIA
Temp=-15°C Clouds=100% DT=2021-12-28T22:13:08

ARE, SE
Temp=-14°C Clouds=72% DT=2021-12-28T22:13:00

JOKKMOKK, SE
Temp=-11°C Clouds=100% DT=2021-12-28T22:12:48

LONDON, UK
Temp=10°C Clouds=97% DT=2021-12-28T22:16:31

LEICESTER, UK
Temp=8°C Clouds=98% DT=2021-12-28T22:16:31

FLYINGE, SWEDEN
Temp=2°C Clouds=100% DT=2021-12-28T22:16:31

MALMO, SWEDEN
Temp=2°C Clouds=100% DT=2021-12-28T22:07:34

MILANO, ITALY
Temp=5°C Clouds=75% DT=2021-12-28T22:10:10
```

If the user selects to not use cookies, all cookies are cleared and the user is redirected to index.jsp page.



4 Software design

4.1 MVC

Model classes include

- WeatherBean.java
  - This is the model for the weather app
  - It includes the data that is extracted from the API
    - Cloudiness in percent
    - Date and time
    - Temperature
- WeatherDataParser.java
  - Gets search parameters from OpenWeatherServlet and pass it to the API

View classes include

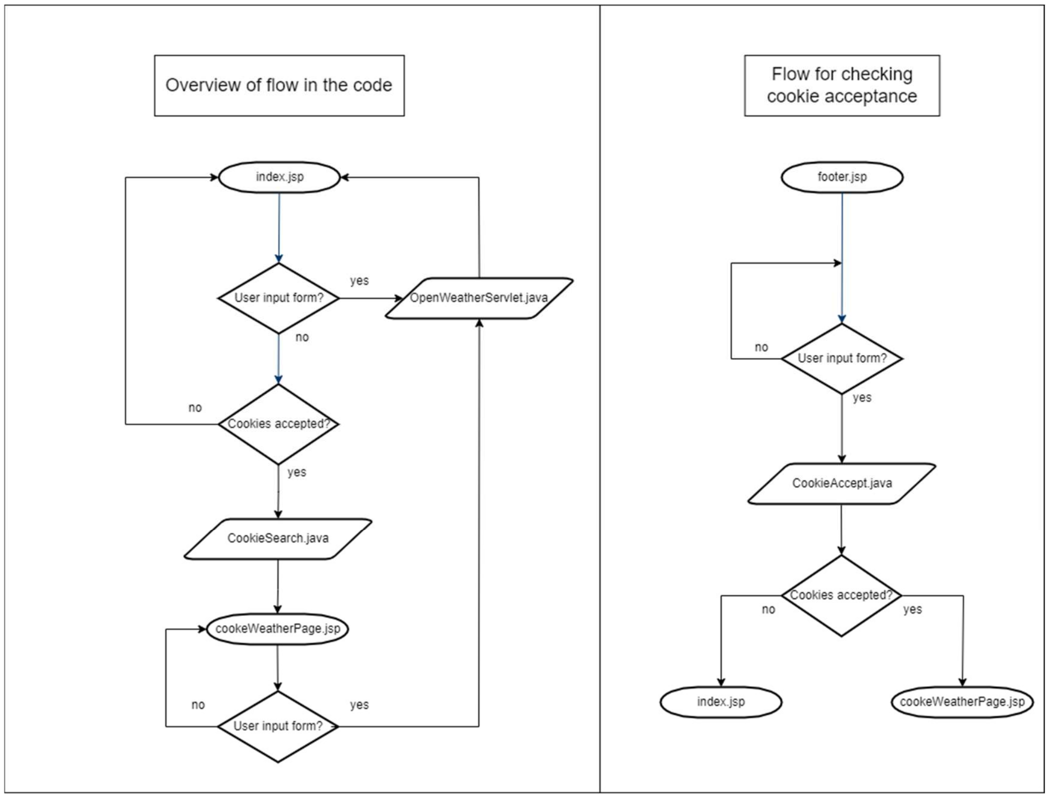
- WEB-INF/weatherForm.html
  - Common search for for index.jsp and cookieWeatherPage.jsp
- footer.jsp
  - Common footer
- header.jsp
  - Common header
- index.jsp
  - First page of the website
  - Decides whether to redirect to cookieWeatherPage.jsp or stay on index.jsp for single searches
- cookieWeaterPage.jsp
  - If cookies are used this page is used

Controller classes

- CookieAccept.java
  - Servlet for checking if user select yes or no to cookies
  - Initiated by form in the footer
- CookieSearch.java
  - Servlet for using the cookies when passing a search request to the API
- OpenWeatherServlet.java
  - Servlet for using an input form when passing a search request to the API

4.2 Flowchart

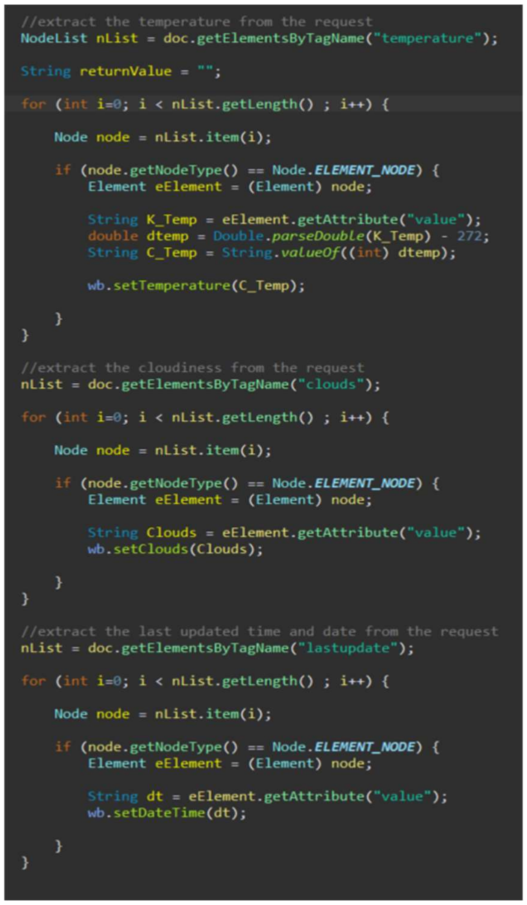
The flowcharts visualize an overview if both the general flow between tasks(left) and how the cookie acceptance is handled(right).
To be noticed is that all searches from the user form will result in going back to the index.jsp who is responsible to direct to either itself or to cookieWeatherSearch.jsp if cookies are accepted.
This is just a design choice and could have been done different, for example handled the choice in the OpenWeatherServlet.java class.



5 Status

1. The search does not handle
   a. inaccurate search parameters
   b. spaces between namnes (for example New York)
   c. ÅÄÖ generates the search but with wrong characters in the result

2. Reading different parameters from the API is now 3 sections of copied code.
   This is of course not state of the art and is a room of improvement.



```
//extract the temperature from the request
NodeList nList = doc.getElementsByTagName("temperature");

String returnValue = "";

for (int i=0; i < nList.getLength() ; i++) {

    Node node = nList.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) node;

        String K_Temp = eElement.getAttribute("value");
        double dtemp = Double.parseDouble(K_Temp) - 272;
        String C_Temp = String.valueOf((int) dtemp);

        wb.setTemperature(C_Temp);

    }
}

//extract the cloudiness from the request
nList = doc.getElementsByTagName("clouds");

for (int i=0; i < nList.getLength() ; i++) {

    Node node = nList.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) node;

        String Clouds = eElement.getAttribute("value");
        wb.setClouds(Clouds);

    }
}

//extract the last updated time and date from the request
nList = doc.getElementsByTagName("lastupdate");

for (int i=0; i < nList.getLength() ; i++) {

    Node node = nList.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) node;

        String dt = eElement.getAttribute("value");
        wb.setDateTime(dt);

    }
}
```

3. Number of cookie search have no limits and it does not look very nice on the website when it grows. This list could have been limited to about 10.

4. It would have looked nice with favourite searches and not only last searches.

5. No responsive design. Does not look good when changing size on screen.