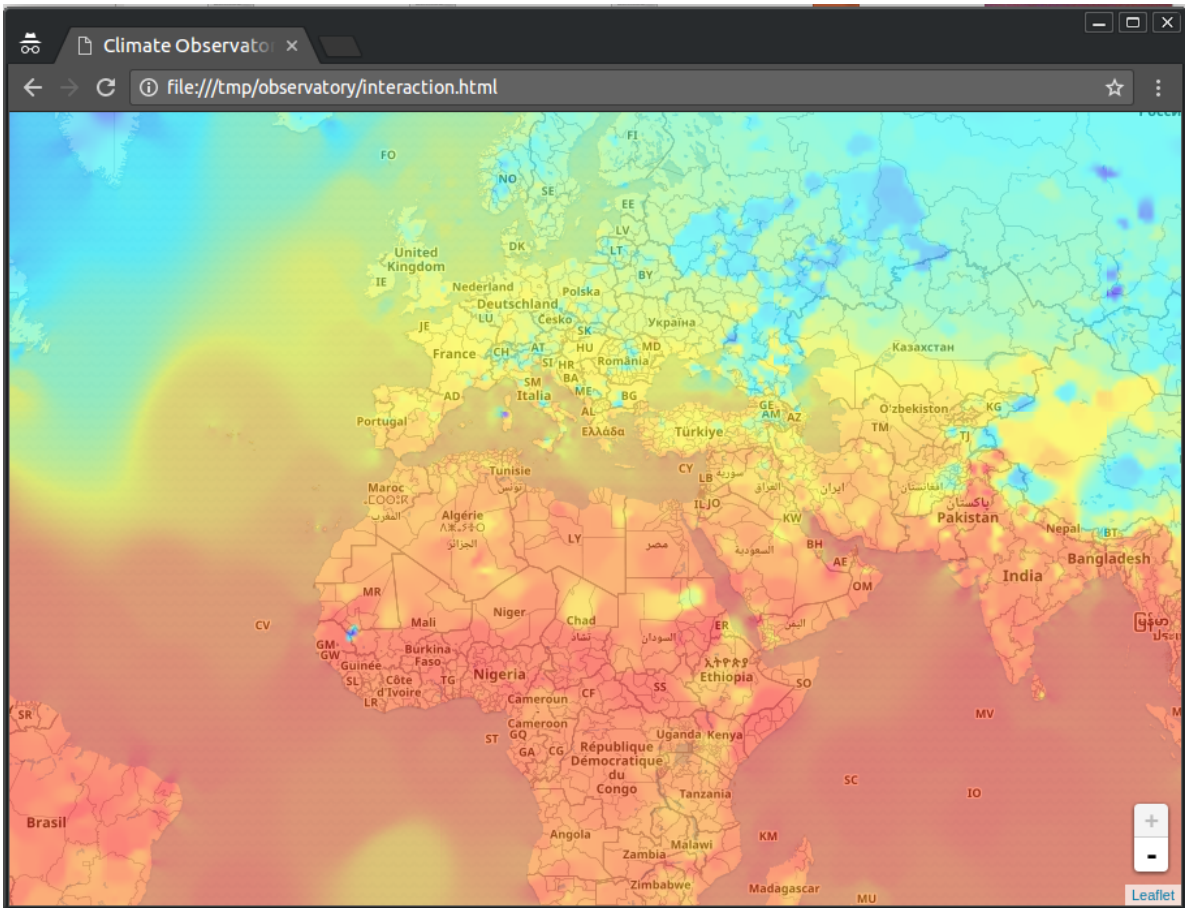
Milestone overview

This milestone consists in producing images compatible with most Web based map visualization tools, so that you can see your data in an interactive Web page. You will have to complete the file **Interaction.scala**. But first, remember to update the grading milestone number:

```
1  val milestone: Int = 3
```

The end result will look like the following (for 2015 with $p = 6$):



In Web based mapping applications, the whole map is broken down into small images of size 256×256 pixels, called **tiles**. Each tile shows a part of the map at a given location and zoom level. Your work consists in producing these tiles using the Web Mercator projection.

You can monitor your progress by submitting your work at any time during the development of this milestone. Your submission token and the list of your graded submissions is available on this page.

Tile generation

To describe the position and size of tiles, we need to introduce the tile coordinate system, which is composed of an x value, a y value and a zoom level. Coordinates in this system are represented by the **Tile** case class, defined in **models.scala**:

```
case class Tile(x: Int, y: Int, zoom: Int)
```

Remember that you are free to add methods to this case class, but that you should not remove or rename its fields.

You have to implement the following two methods:

```
def tileLocation(tile: Tile): Location
```

This method converts a tile's geographic position to its corresponding GPS coordinates, by applying the Web Mercator projection.

```
def tile(
  temperatures: Iterable[(Location, Temperature)],
  colors: Iterable[(Temperature, Color)],
  tile: Tile
): Image
```

This method returns a 256×256 image showing the given temperatures, using the given color scale, at the location corresponding to the given zoom, x and y values. Note that the pixels of the image must be a little bit transparent so that when we will overlay the tile on the map, the map will still be visible. We recommend using an alpha value of 127.

Hint: you will have to compute the corresponding latitude and longitude of each pixel within a tile. A simple way to achieve that is to rely on the fact that each pixel in a tile can be thought of a subtile at a higher zoom level ($256 = 2^8$).

Integration with a Web application

Once you are able to generate tiles, you can embed them in a Web page. To achieve this you first have to generate all the tiles for zoom levels going from 0 to 3. (Actually you don't *have* to generate all the tiles, since this operation consumes a lot of CPU. You can choose to generate tiles for just one zoom level, e.g. 2). To each zoom level corresponds tiles partitioning the space. For instance, for the zoom level "0" there is only one tile, whose **(x, y)** coordinates are **(0, 0)**. For the zoom level "1", there are four tiles, whose coordinates are **(0, 0)** (top-left), **(0, 1)** (bottom-left), **(1, 0)** (top-right) and **(1, 1)** (bottom-right).

The **interaction.html** file contains a minimalist Web application displaying a map and a temperature overlay. In order to integrate your tiles with the application, you must generate them in files located according to the following scheme: **target/temperatures/2015/<zoom>/<x>-<y>.png**. Where **<zoom>** is replaced by the zoom level, and **<x>** and **<y>** are replaced by the tile coordinates. For instance, the tile located at coordinates **(0, 1)**, for the zoom level 1 will have to be located in the following file: **target/temperatures/2015/1/0-1.png**.

Once you have generated the files you want to visualize, just open the **interaction.html** file in a Web browser.

Future integration with a more complete Web application

At the end of the project you will be going to display these temperature data in a more complete Web application, allowing for example to select which year to visualize. You can prepare this integration by generating the tiles for all the years between 1975 and 2015. You should put the generated images in the following location: **target/temperatures/<year>/<zoom>/<x>-<y>.png**. This is going to take a lot of time, but you can make the process faster:

- Identify which parts of the process are independent and perform them in parallel ;
- Reduce the quality of the tiles. For instance, instead of computing 256×256 images, compute 128×128 images (that's going to be 4 times fewer pixels to compute) and then scale them to fit the expected tile size.

Finally, you will have to implement the following method:

```
def generateTiles[Data](
  yearlyData: Iterable[(Year, Data)],
  generateImage: (Year, Tile, Data) => Unit
): Unit
```

This method generates all the tiles for a given dataset **yearlyData**, for zoom levels 0 to 3 (included). The dataset contains pairs of **(Year, Data)** values, usually otherwise, data associated with years. In your case, this data will be the result of **Extraction.locationYearlyAverageRecords**. The second parameter of the **generateTiles** method is a function that takes a year, the coordinates of the tile to generate, and the data associated with the year, and computes the tile and writes it on your filesystem.

Mark as completed