Department of Information Engineering and Computer Science
Fintech
University of Trento

Web Architectures

# Assignment 3

6 November 2022

## Index

1. The case
2. My solution
3. App running
4. Comments

## The case

A little sum-up of the assignment.

1. Implement a simple spreadsheet hosted by a web server. The formula evaluation and calculation must be handled by the server and the result delivered to the client
2. With a simple pooling allow multiple users to interact with the spreadsheet.

## My solution

Once the HTML page is loaded (index.html) the whole grid is fetch from the server. The server hosts a singleting instance of the provided *SSEngine*, this is controllerd by a Controller Class and shared between all the servlets. The servlet that handles the first whole fetch, like all the other, parses the payload, in this case the hashmap of the cells, into a JSON that will be delivered to the client.

To parse the data I used GSON for his simplicity compared to others libraries like Jackson. The GSON instance is created setting `excludeFieldsWithoutExposeAnnotation()` to parse onle the selected properties marked with the `@Expose` annotation.

Listing 1: A simplified version of the response payload at the first fetch.

```
{
  "cells": {
    "D1": {
      "id": "D1",
      "formula": "",
      "value": 0
    },
    "C1": {
      "id": "C1",
      "formula": "",
      "value": 0
    }
    ...
  },
  "grid": {
    "nRows": 4,
    "nColumns": 4
  }
}
```

Looking at the listing we can see that the project allow a dynamic configuration of the grid size. This not only by the server but also by the client. To allow it the grid size is passed at the first fetch.

The client generate the grid from the provided JSON thanks to JQuery that expose some handy APIs. Once a cell is clicked, *JavaScript* updated the formula field and once the enter key is pressed, a listener function is triggered and a *POST* request is sent to the server with the new formula to be evaluated and calculated.

GSON allows also to unparse a JSON string into a Java Class Instance. By doing so I created some base classes for every client request with a payload. Looking for some possible crash eventually a *400* error response is sent.

Listing 2: PostFormula.java

```java
public class PostFormula {
    @Expose
    String cellId;
    @Expose
    String formula;

    public void setCellId(String cellId) {
        this.cellId = cellId;
    }

    public void setFormula(String formula) {
        this.formula = formula;
    }
    public String getCellId() {
        return cellId;
    }

    public String getFormula() {
        return formula;
    }
}
```

Listing 3: GSON verification of a valid payload

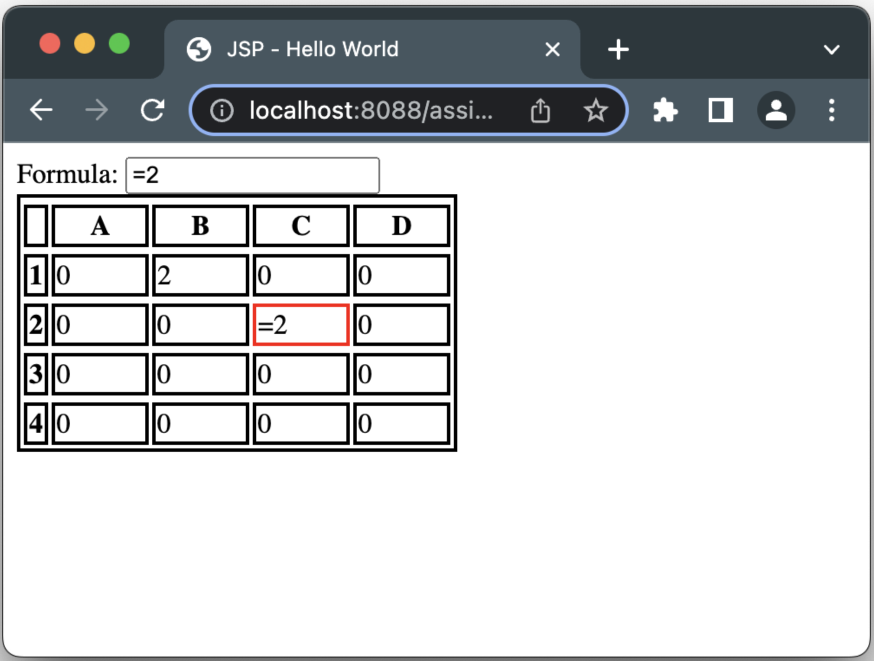
```java
PostFormula postFormula = null;
try {
    postFormula = gson.fromJson(requestBody, PostFormula.class);
} catch (Exception e ){
    e.printStackTrace();
    response.sendError(HttpServletResponse.SC_BAD_REQUEST);
}
if (postFormula == null) return;
```

The modified formula is passed to the SSEngine the set of modified cells is received by the client that will updated the corrisponding HTML tags.

Every time a change is made, the server records the modified cells and the local timestamp into a array. When a client looks for some new updates (every *5* seconds) the server will send all the cells changes occurred since the last request. I choose this solution to main some scalability and to facilitate other future functionalities. Of course for a production solution the servers would need to use the FileSystem or to truncated the change records to avoid RAM overflow.

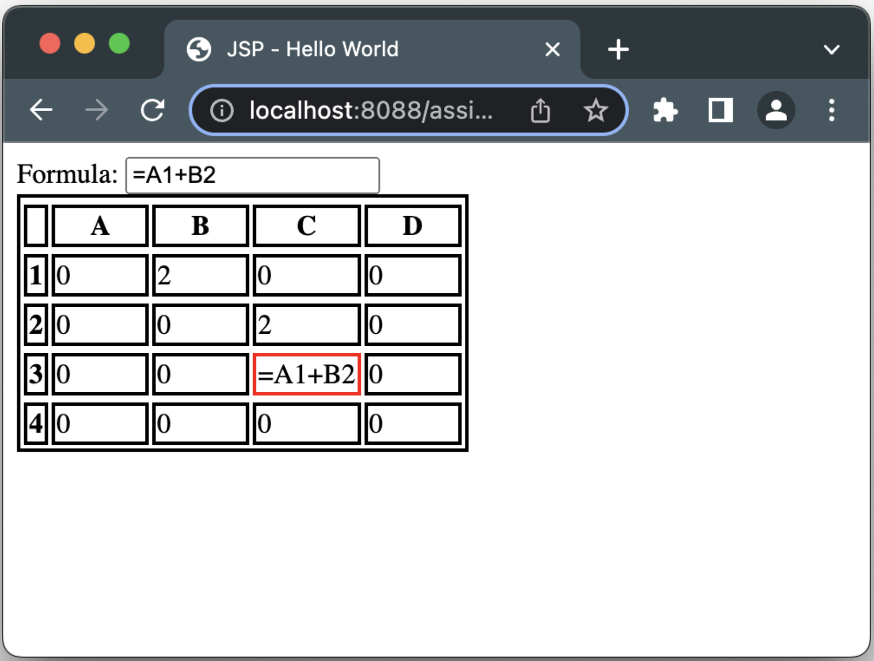
## App running

Here some screenshots of the app running. The UI is the same as shown in the assignment example.



Formula: =2

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 2 | 0 | 0 |
| 2 | 0 | 0 | =2 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |



Formula: =A1+B2

|   | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 2 | 0 | 0 |
| 2 | 0 | 0 | 2 | 0 |
| 3 | 0 | 0 | =A1+B2 | 0 |
| 4 | 0 | 0 | 0 | 0 |

## Comments

This project help to understand the basic concept of HTTP and JSON and to reconfirm the knowledge learned so far. I would love the see in the next assignment a more efficient solution to listen for new updated as seen in class.