|  |
| --- |
|  |
| Assignment 2 documentation |
| Advanced Java |

## Contents

[**1.Introduction** 2](#_Toc95416888)

[**2.Solution** 2](#_Toc95416889)

[**3.Other information:** 5](#_Toc95416890)

# **1.Introduction**

The reason for making this documentation is to provide more information on the solution of the project problem in assignment 2. The problem to be solved in the assignment is described as : “Your application should read this data into an appropriate data structure, in this case it will be a database and then process it accordingly. This should be done through developing a web application”. This web application has to be built using the JSF framework. The web should implement functionalities for both anonymous users and “regular users” or as they can be called the ones who have account that can login. The anonymous users will be able to access the index view where he/she can view covid data, filter them based on a country or continent or be able to sort the shown data based on a column. As for the regular users they will have the same abilities of anonymous users and some others as well. First of all the regular user has to have an account with which he can login by giving the email and password. Then they have functions about adding, removing or updating countries or covid data. Besides this they can also update their profile besides the email address. Lastly the logged in users will have access to a user API which will allow them to move to the different views that handle the above mentioned functionalities.

# **2.Solution**

The project has a total of 6 views and 4 classes, 3 of which are beans, and 1 which is used for the database connection. From here on I will talk about only the 3 bean classes as the database connection class does nothing else besides establishing a connection. The 3 classes are those of covid data, countries and user. In each of them there are properties, getters and setters for those properties and then the methods that are used in the 6 views to implement different functionalities. Going to the views, lets start with the “index” view which is also the starting view. There is a table(or datatable element) that shows the covid data. Besides that there are field to input email and password to login if you click the login button. If the data provided are not right the view will not change, otherwise the user will be sent to the user view. The above mentioned functionality for the login is handled by a method in the “UserBean” called “LogIn”. There are also input field to enter the iso code, continent or country and for each of them a button which when clicked can filter out data for that given iso code, continent or country. On the data columns that can be used to sort the data the text is displayed as a link which can be clicked in order to sort the data based on that column. All the above functionalities like the 3 filters or all the sorting functionalities are handled each by a separate method in the “CountryBean” class. The logic for the sort methods is the same, there is a Boolean variable that will be used to handle whether to sort data in ascending or descending order and every time that data is sorted the value of the Boolean variable is changed and then the data is sorted in the reverse order from what it was sorted before (always for the same column). Just like the sorting methods, even the filter methods have the same logic, by giving the data(iso code, country or continent), a search in the database is done for that country or continent and then all covid data related to that are searched in the “covid\_data” table so that the iso code of that country or continent which is in the “countries” table and the one referenced in the “covid\_data” table are the same. Then the user view has 3 input fields for the name, surname and password which all are required and have other validations as mentioned to update the profile is handled by a method called “Update” in the “UserBean” class. After executing the update by clicking the button the user is redirected to the index view. When a user is logged in there are 5 links at the top that redirects the user to one of the other 5 views based on which link is clicked. Then there are the 2 views “removeCountry” and “removeData” which as the name suggests as well are used to remove a country or a data. For the country the user only has to supply the iso code, which is a required field, and this is handled by a method called “Remove” in the “CountryBean” class. As for the removal of the data, the user has to supply both the iso code and the date for the data to be removed. Both fields are required and this is handled by a method called “Remove” in the “CovidDataBean” class. Both of them check if for the data provided there is a country or covid data and if there is that data is removed and the user is sent to the index view, otherwise the user just stays in the current view (either removeCountry or removeData view). Lastly there are 2 more views, the “countries” and “covidData” views, where there are the tables for either the countries or covid data depending on the view and on top of that table there are input fields for the data for the countries and covid data depending on the views and 2 buttons, 1 for update and 1 for save. As the name suggests the save button saves(or adds) new data and the update button updates the current data based on the iso code for the country and gives the values of the other input fields as the updated value, and based on both the iso code and date for the covid data and gives the values of the input fields there as the updated value for that country or covid data. These are both held by a method called “Save” and “Update” in both “CountryBean” and “CovidDataBean” respectively. As for the classes, besides the methods that we already mentioned above there are other methods that are used by these method to fulfill the functionality. As for the properties, the “UserBean” class has 4, 1 for the name, 1 for the surname, 1 for the email and 1 for the password, all of type String. The other 2 classes have the same properties as in the first assignment and the same types, besides the population which I made BigDecimal instead of double because if the number was too big it would be represented in a scientific way, the same could be done for other values if needed but with the few data I tested it, the population was the only one that showed in scientific notation, but for a bigger range of data I believe that the “totalCases” or other “total” properties may also be shown in a scientific notation. After making the code cleaner I had to create a new class which has a method that creates the connection with mysql and returns that connection and then I call that method of that class at the beginning of the other 3 classes that use the database. This way the user will not have to initiate a connection and provide the data for it in every method but just provide the data in that class. Then I added a new functionality to add all data from a csv file which is done by tapping the button “Add Data” when logged in and it is handled by a method which I put in the “CountryBean” class called “addData”. This method adds all countries and covid data from the csv file I have provided through the path. If we wanted to ask the user for the path we could have done it just like the filter functionalities are done. Lastly I created a class for the API called “WebServiceClass” which returns a list of covid data. It first checks for the continent, which if it is not null it returns a list a data belonging to that continent, if it is not but the list of all countries is not null then it returns a list of data about all those countries, otherwise I have made it to return the list of all data.

# **3.Other information:**

In the code there are many repetitions for a certain code like the part of getting a connection, of setting and getting all the data from or to an sql statement or the methods which are used to filter or sort the data, but as the focus was to implement those functionalities mentioned in the assignment and not to make the code “cleaner” I decided not to change any of them. If that part had to be done as well a very easy way would be to implement a certain method once and then call it where needed, for example we could make a method to have the connection and instead of initiating a connection in every different method we could just call that method in those methods, or for the case of the connection even establish a class to do this. Like this could be done for all other repeating codes if needed.

Besides that for the user API when the user is logged in, I was not sure about what had to be done but from the small explanation that you gave us at our make-up class on 29/01/202 to make it as “some sort of questions or ways for the user to go and do those functionalities mentioned”, I understood it as a way to travel to the different vies as they handle those functionalities that is why when the user is logged in there are 5 links at the top of the page to travel to one of the other views. During our make-up class on 29/01/202 we were told that we did not need the role of the regular user that at first it was there for adding new users if the user was an admin but that you decided to remove that, so I understood that we did not need the role of the user so I decided to remove it from both the database and as a property of the user I the “UserBean” class.

Also when the user does an update, remove or save of the data that they are redirected to the index view, the data are as they were left, but if one of the buttons like the “Show all” “Show last date” button is clicked the data will be shown updated. When a covid data for a country that does not exist in the database is tried to be added there will be an error thrown. In the “CountryBean” for the filtering of all data and last date data they can both be done by using the database, or both by calling the methods that handle those functionalities in the “CovidDataBean” class, but I decided to call the methods from the “CovidDataBean” class in order to keep the code cleaner. As for the API I could not test it, just wrote the code how I thought it would logically work because of the problem we saw together at the last class we had, and that is why I didn’t even create a wsdl file about in in the project. So just added the class there to show the logic part of how it would work. As for the part of adding data from csv file, it needs a lot of time to add the data at the beginning but after adding the data the user can sort the data, can view all countries, can filter the data, can show last date and even remove, update or add countries, but when the function to show all covid data is called that is where problems start because the application gives the warning that there are too many connections opened and does this in an unending loop. So I decided to remove the data once more from the database and input a few data myself so I could do some more testing, and when there are not that much data all functionalities(even those who called the function to show all data) worked fine. As for the database when I use my application I always use “xampp” to start “mySQL”. Lastly because I could not add the option for a user that is not logged in to not access the other views through the search bar, I decided to add a “panel grid” which will be available only to logged in users that will let them to interchange views by clicking on those links that are there.

**Worked by:** Enis Sejdini