LAB 1: RESTFUL WEB SERVICES

DINESH, Gautam [20040968D] & ISLAM AHNAF AL [20041569D]

VIDEO LINK: 20040968D&20041569D EIE4108 Lab1 recording (youtube.com)

Introduction

RESTful web services simply and decouple various server components so that each part can be modified independently allowing total client server separation. This lab aims to allow students to understand and implement RESTful web services by writing a server side and client-side script and hosting it in a local container.

Objectives:

- Develop, deploy, and consume RESTful web services based on JAX-RS.
- Utilize IT tools to develop RESTful web services.

Methodology:

The server-side code defines 4 different functions: Add Record, Get Record, Update Record and Delete Record. The HTML code defines the body of the webpage and defines the function of each of the buttons. It sends the requests to the container when the buttons are pressed, and container runs the server-side Java program to deal with the requests and send out the returned values to the HTML code to be displayed.

Add function:

The Java code for the add function is as follows:

```java
@POST
@Path("{id},{mark}")
public String addRecord(@PathParam("id") String id, @PathParam("mark") int mark) {
    // Add a record using id as key and mark as value
    if (table.containsKey(id)) {
        return "Record of " + id + " exists";
    } else {
        table.put(id, mark);
        return "Record of " + id + " added";
    }
}
```

The HTML code to implement the add function is as follows:

```javascript
function addRecord() {
    let req = createRequest();
    let id = document.getElementById("ID");
    let mark = document.getElementById("Mark");
    let status = document.getElementById("Status");

    // Complete the code of addRecord() below using the correct HTTP
    // request for adding records
    let url = "mark/" + id.value + "," + mark.value;
    req.onreadystatechange=function() {
        if (req.readyState===4 && req.status===200) {
            status.value = req.responseText;
        }
    }
    req.open("POST",url, true);
    req.send();
}
```

the HTML code takes the values entered in the ID and mark fields and sends them alongside a Post request to the container. The Java program then compares the id value with the values stored in the table, if it is already present in the table then it returns a status message indicating that the record already exists in the table. If the id is not in the table, then it adds the id and mark to the table.

Get Function:

The Java code for the get record function is as follows:

```
@GET
public String getMark(@QueryParam("id") String id) {
    // Retrieve a record using id as key
    if (table.containsKey(id)) {
        return "Mark of " + id + ": " + table.get(id);
    } else {
        return "Record of " + id + " not exists";
    }
}
```

The HTML code for the get function is as follows:

```
function getRecord() {
    let req = createRequest();
    let id = document.getElementById("ID");
    let mark = document.getElementById("Mark");
    let status = document.getElementById("Status");

    // Complete the code of getRecord() below using the correct HTTP
    // request for retrieving records
    let url = "mark?id=" + id.value;
    req.onreadystatechange=function() {
        if (req.readyState===4 && req.status===200) {
            status.value = req.responseText;
        }
    }
    req.open("GET",url, true);
    req.send();
}
```

the HTML code takes the value entered in the ID field and sends the value alongside a Get request to the container. The Java program then compares the id value with those in the table and if the record exists then it returns a status message containing the id and the stored marks alongside it. If the id does not exist in the table, it then returns a Status message indicating so.

Update Function:

The Java code for the update function is as follows:

```
@PUT
@Path("{id},{mark}")
public String updateRecord(@PathParam("id") String id, @PathParam("mark") int mark) {
    // Update a record using id as key and mark as value
    if (table.containsKey(id)) {
        table.put(id, mark);
        return "Record of " + id + " updated";
    } else {
        return "Record of " + id + " not exists";
    }

}
```

The HTML code for the update function is as follows:

```
function updateRecord() {
    let req = createRequest();
    let id = document.getElementById("ID");
    let mark = document.getElementById("Mark");
    let status = document.getElementById("Status");

    // Complete the code of updateRecord() below using the correct HTTP
    // request for updating records
    let url = "mark/" + id.value + "," + mark.value;
    req.onreadystatechange=function() {
        if (req.readyState===4 && req.status===200) {
            status.value = req.responseText;
        }
    }
    req.open("PUT",url, true);
    req.send();
}
```

The HTML code for the functions takes the values entered in the id and mark fields and sends it in a PUT request to the container. The Java code then compares the received id value with already stored values in the table, if the id exists then it updates the mark stored alongside the id value the returns a status message confirming the update, if the id value is not in the table, then it returns a message indicating that the record does not exist.

Delete function:

The Java code for the delete code is as follows:

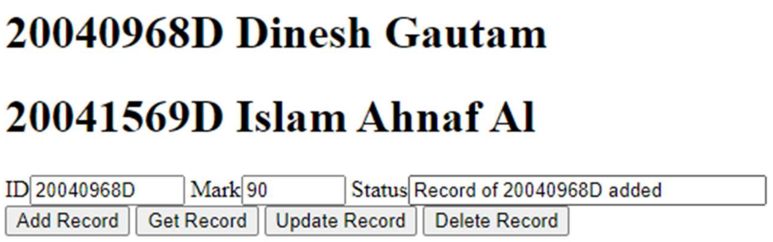
```
@DELETE
public  String deletRecord(@QueryParam("id") String id){
    if (table.containsKey(id)) {
        table.remove(id);
        return "Removed record of " + id + ": " + table.get(id);
    } else {
        return "Record of " + id + " not exists";
    }
}
}
```

The HTML code for the delete function is as follows:

```
function deleteRecord(){
    let req = createRequest();
    let id = document.getElementById("ID");
    let status = document.getElementById("Status");
    let url = "mark?id=" + id.value;
    req.onreadystatechange=function() {
        if (req.readyState===4 && req.status===200) {
            status.value = req.responseText;
        }
    }
    req.open("DELETE",url, true);
    req.send();
}
```

the HTML code takes the id value entered and sends with a delete request to the container. The Java program then compares the received value with those stored in the table and if the record exists then the record is removed from the table and a status message indicating so is returned to be displayed. if the record doesn't exist then a status message indicating it is returned to be displayed.

Results:

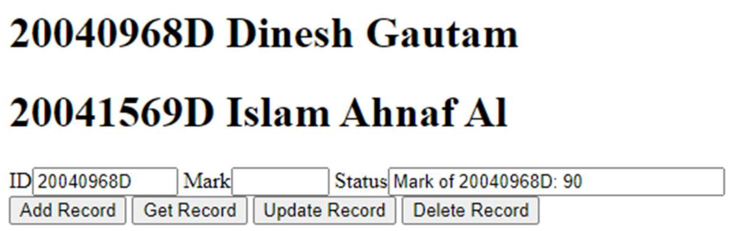


20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

ID 20040968D Mark 90 Status Record of 20040968D added

The above screenshot shows the result after adding a record to the table.

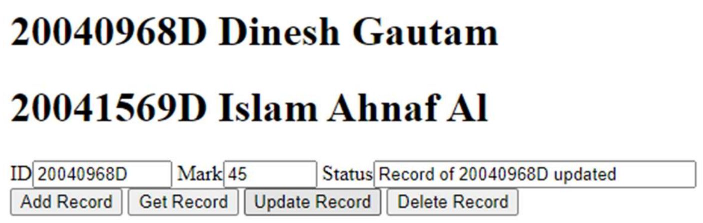


20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

ID 20040968D Mark Status Mark of 20040968D: 90

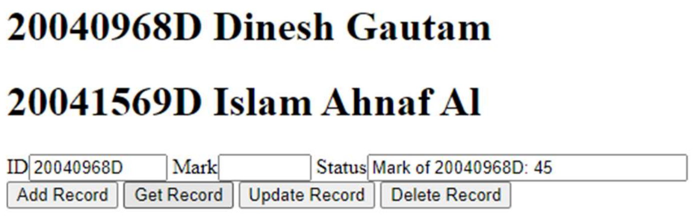
The above screenshot shows the outcome after requesting a record.



20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

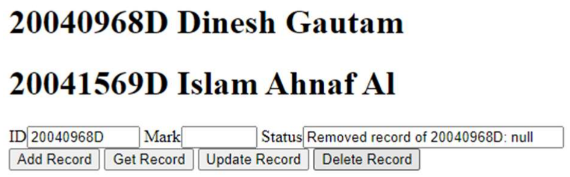
ID 20040968D Mark 45 Status Record of 20040968D updated


20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

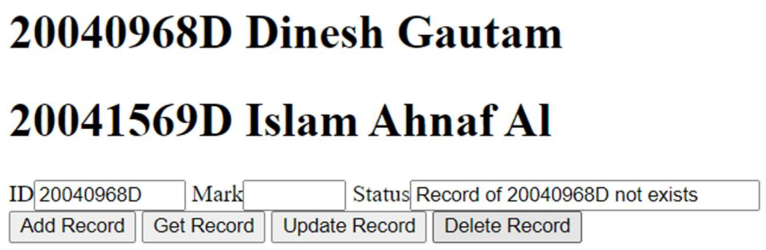
ID 20040968D Mark Status Mark of 20040968D: 45

The above screenshots show the outcome after the update button is pressed.


20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

ID 20040968D Mark Status Removed record of 20040968D: null


20040968D Dinesh Gautam

20041569D Islam Ahnaf Al

ID 20040968D Mark Status Record of 20040968D not exists

The above screenshots show the outcome when the delete button is pressed.

Conclusion:

To conclude, we successfully modified the Database.java and Testmarks.html to achieve the desired outcomes. We successfully implemented the add, get and update records and the Testmarks.html is the code for the client side that that sends requests to the container database functions. Additionally, we also incorporated a special function to delete records stored in the database.