Verteilte Systeme im Sommersemester 2021

Steffen Herweg, Matr. Nr. 873475 Luca Fabio Kock, Matr. Nr. 879534

Osnabrück, 14.06.2021

Aufgabenblatt 8

BillboardServlet.java:

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String caller ip = request.getRemoteAddr();
        System.out.println("BillBoardServer - POST (" + caller ip + ")");
        // TODO implementation of doPost()!
        String message = request.getParameter("message");
        if (message != null) {
            bb.createEntry(message, caller ip);
        }
        response.getWriter().close();
    }
    protected void doDelete(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String caller ip = request.getRemoteAddr();
        System.out.println("BillBoardServer - DELETE (" + caller ip + ")");
        try {
            int id = Integer.parseInt(request.getParameter("id"));
            if (bb.getEntry(id).owner ip.equals(caller ip)) {
                bb.deleteEntry(id);
            }else{
                response.sendError(403, "Owner falsch!");
            }
```

```
} catch (NumberFormatException e) {
            response.sendError(400, "Malformed Parameter id!");
        }
        response.getWriter().close();
    }
    protected void doPut(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String caller ip = request.getRemoteAddr();
        System.out.println("BillBoardServer - PUT (" + caller ip + ")");
        try {
            int id = Integer.parseInt(request.getParameter("id"));
            String message = request.getParameter("message");
            if (message != null) {
                if (bb.getEntry(id).owner ip.equals(caller ip)) {
                bb.updateEntry(id, message, caller ip);
            }else{
                response.sendError(403, "Owner falsch!");
            } else {
                response.sendError(400, "Parameter message missing!");
            }
        } catch (NumberFormatException e) {
            response.sendError(400, "Malformed Parameter id!");
        }
        response.getWriter().close();
    }
```

BillboardJSONAdapter.java:

```
@Override
    public String readEntries(String caller ip) {
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("{\n");
        stringBuilder.append("\t\"billboard\":[\n");
        for(int i = 0; i < this.billboard.length; i++) {</pre>
            stringBuilder.append("\t\t" + readEntry(i,caller ip));
            if(i != this.billboard.length - 1) {
                stringBuilder.append(",");
            stringBuilder.append("\n");
        }
        stringBuilder.append("\t]\n");
        stringBuilder.append("}");
        return stringBuilder.toString();
    }
    @Override
    public String readEntry(int idx, String caller ip) {
        BillBoardEntry billBoardEntry = this.billboard[idx];
        return
String.format("\{\'': \&d, \''' = adonly\'': \&b\}'', \'''
billBoardEntry.id,billBoardEntry.text,!billBoardEntry.owner ip.equals(calle
r_ip));
    }
Billdboard.js:
function getxyzHttpRequest(url) {
    fetch(url).then(response=>response.json().then(buildTable));
}
function postHttpRequest(url) {
    const inputElement = document.getElementById("contents")
```

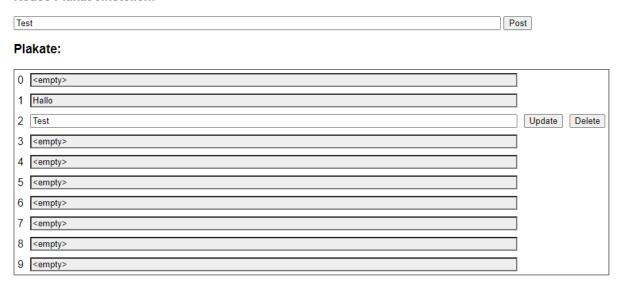
```
fetch(url + "?message=" +
encodeURIComponent(inputElement.value), {method: "POST"});
}
function putHttpRequest(url, id) {
    let inputElement = $(`input field ${id}`);
    fetch(url + `?message=${inputElement.value}&id=${id}`,{method: "PUT"});
}
function deleteHttpRequest(url, id) {
    fetch(url + `?id=${id}`, {method: "DELETE"});
}
function buildTable(object) {
   console.log(object)
    let postersElement = $("posters");
   postersElement.innerHTML = "";
    let tableElement = document.createElement("table");
    tableElement.rules = "none";
    tableElement.cellSpacing = 4;
    tableElement.cellPadding = 5;
    tableElement.border = "1";
    for(let post of object.billboard) {
        let postElement = document.createElement("tr");
        tableElement.appendChild(postElement);
        let idElement = document.createElement("td");
        postElement.appendChild(idElement);
        idElement.innerText = post.id;
        let messageElement = document.createElement("td");
        postElement.appendChild(messageElement);
        let messageInputElement = document.createElement("input");
        messageElement.appendChild(messageInputElement);
```

```
messageInputElement.type = "text"
        messageInputElement.size = 100;
        messageInputElement.minlength = 100;
        messageInputElement.maxlength = 100;
        messageInputElement.id = `input element ${post.id}`;
        messageInputElement.value = post.message;
        messageInputElement.readOnly = post.readonly;
        let updateElement = document.createElement("td");
        postElement.appendChild(updateElement);
        let deleteElement = document.createElement("td");
        postElement.appendChild(deleteElement);
        if(!post.readonly){
            let updateButtonElement = document.createElement("button");
            updateElement.appendChild(updateButtonElement);
            updateButtonElement.innerText = "Update";
updateButtonElement.addEventListener("click",()=>postHttpRequest("BillBoard
Server", post.id));
            let deleteButtonElement = document.createElement("button");
            deleteElement.appendChild(deleteButtonElement);
deleteButtonElement.addEventListener("click",()=>deleteHttpRequest("BillBoa
rdServer",post.id));
            deleteButtonElement.innerText = "Delete";
        }
    }
    postersElement.appendChild(tableElement);
}
```

Testen des Billboards auf 2 verschiedenen Clients:

Plakatwand

Neues Plakat einstellen:



Plakatwand

Neues Plakat einstellen:

Hallo

Plakate:

