*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++){

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("{\"id\":%d,\"message\":\"%s\",\"readonly\":%b}", billBoardEntry.id,billBoardEntry.text,!billBoardEntry.owner\_ip.equals(caller\_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("BillBoardServer",post.id));

let deleteButtonElement = document.createElement("button");

deleteElement.appendChild(deleteButtonElement);

deleteButtonElement.addEventListener("click",()=>deleteHttpRequest("BillBoardServer",post.id));

deleteButtonElement.innerText = "Delete";

}

}

postersElement.appendChild(tableElement);

}

Nach Veränderung

Für ausschließliche Datenübertragung bei nicht null Werten

In BillBoardEntry:

public BillBoardEntry(int i, String caller\_ip) {

id = i;

//this.text = text;

owner\_ip = caller\_ip;

setTimeStamp();

}

public long getTimeStamp() {

return timestamp;

}

/\* Wird bei Modifikationen und Erzeugung aufgerufen \*/

public void setTimeStamp() {

timestamp = System.currentTimeMillis();

}

public void reset() {

owner\_ip = "<not set>";

timestamp = 0;

text = null;

}

In BillBoardJSONAdapter

Filtzer nach nachrichten mit gesetzter message

stringBuilder.append(

Arrays.stream(this.billboard).filter(new Predicate<BillBoardEntry>() {

@Override

public boolean test(BillBoardEntry billBoardEntry) {

return billBoardEntry.text != null;

}

}).map(new Function<BillBoardEntry, String>() {

@Override

public String apply(BillBoardEntry billBoardEntry) {

return String.format("{\"id\":%d,\"message\":\"%s\",\"readonly\":%b}", billBoardEntry.id, billBoardEntry.text, !billBoardEntry.owner\_ip.equals(copy));

}

}).collect(Collectors.joining(","))

);

BillBoardServlet

Für Longpolling

private final Set<Thread> waiting = new HashSet<>();

boolean wait = Boolean.parseBoolean(request.getParameter("wait"));

if (wait) {

try {

waiting.add(Thread.currentThread());

System.out.printf("%s STARTED SLEEPING\n",Thread.currentThread().getName());

System.out.printf("%d sleeping\n", waiting.size());

Thread.sleep(10000);

waiting.remove(Thread.currentThread());

} catch (InterruptedException e) {

System.out.printf("%s INTERRUPTED\n",Thread.currentThread().getName());

}

}

private void interrupt(){

System.out.println(waiting.size());

waiting.forEach(new Consumer<Thread>() {

@Override

public void accept(Thread t) {

t.interrupt();

}

});

waiting.clear();

}

**Testen des Billboards auf 2 verschiedenen Clients:**

