*Verteilte Systeme im Sommersemester 2021*

Steffen Herweg, Matr. Nr. 873475

Luca Fabio Kock, Matr. Nr. 879534 Osnabrück, 07.06.2021

# Aufgabenblatt 7

**getMessage.jsp:**

<%@page import="java.util.Date"%>

<%@ page import="chatsystem.servlet.\*" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="refresh" content="5; URL=/chatsystem/Chat?action=getMessage">

<title>JSP Page</title>

</head>

<body>

<textarea readonly cols="50" rows="10"><%

ClientProxyImpl clientProxy = (ClientProxyImpl)session.getAttribute(Chat.CLIENT\_PROXY\_ATTR);

for(ClientProxyImpl.Message message: clientProxy.messages){

out.print(message.toString());

}

%></textarea>

<p>Last Refresh: <% out.print(new Date()); %></p>

</body>

</html>

**Chat:**

<head>

<title>Chat System</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

iframe{

border:none;

width:100%;

margin: 0;

padding: 0;

height:400px;

}

body{

margin: auto;

width:500px;

}

html{

width:100%;

}

</style>

</head>

<body>

<iframe src="?action=getMessage"></iframe>

<iframe src="?action=sendMessage&message="></iframe>

<form>

<input type="hidden" name="action" value="unsubscribe">

<input type="Submit" Value="Close">

</form>

</body>

<script>

const unsubscribeForm = document.getElementById("unsubscribeForm");

window.addEventListener("onbeforeunload",(event)=> {

unsubscribeForm.submit();

event.preventDefault;

e.returnValue = "";

return "null";

});

</script>

**Chat.java:**

@WebServlet(name = "Chat", urlPatterns = {"/Chat"})

public class Chat extends HttpServlet {

public static final String ACTION\_PARAM = "action";

public static final String NAME\_PARAM = "name";

public static final String MESSAGE\_PARAM = "message";

public static final String ACTION\_INIT = "init";

public static final String ACTION\_SENDMESSAGE = "sendMessage";

public static final String ACTION\_UNSUBSCRIBE = "unsubscribe";

public static final String ACTION\_GETMESSAGE = "getMessage";

public static final String CHAT\_PROXY\_ATTR = "chatProxy";

public static final String CHAT\_SERVER\_ATTR = "chatServer";

public static final String NAME\_ATTR = "name";

public static final String CLIENT\_PROXY\_ATTR = "clientProxy";

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String action = request.getParameter(ACTION\_PARAM);

if (action == null) {

request.getRequestDispatcher("/init.html").forward(request, response);

}

switch (action) {

case ACTION\_INIT:

doInit(request, response);

break;

case ACTION\_SENDMESSAGE:

doSendMessage(request, response);

break;

case ACTION\_UNSUBSCRIBE:

doUnsubscribe(request, response);

break;

case ACTION\_GETMESSAGE:

doGetMessage(request, response);

break;

default:

response.sendError(400, "Invalid Action");

}

}

private void doInit(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {

try {

String name = request.getParameter(NAME\_PARAM);

if (name == null) {

response.sendError(400, "No Name Provided");

return;

}

Registry registry = LocateRegistry.getRegistry(Main.PORT);

ChatServer chatServer = (ChatServer) registry.lookup(Main.NAME);

ClientProxyImpl clientProxy = new ClientProxyImpl();

ChatProxy chatProxy = chatServer.subscribeUser(name, clientProxy);

HttpSession session = request.getSession(true);

session.setAttribute(CHAT\_SERVER\_ATTR, chatServer);

session.setAttribute(CHAT\_PROXY\_ATTR, chatProxy);

session.setAttribute(NAME\_ATTR, name);

session.setAttribute(CLIENT\_PROXY\_ATTR, clientProxy);

request.getRequestDispatcher("/chat.html").forward(request, response);

} catch (RemoteException | NotBoundException ex) {

response.sendError(503, "Could not connect to RPC ");

}

}

public void doSendMessage(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {

//request.getRequestDispatcher("/sendMessage.html").forward(request,response);

HttpSession session = request.getSession();

ChatProxy chatProxy = (ChatProxy) session.getAttribute(CHAT\_PROXY\_ATTR);

if (chatProxy == null) {

response.sendError(500, "ChatProxy for this Session is null");

return;

}

String message = request.getParameter(MESSAGE\_PARAM);

if (message == null) {

response.sendError(400, "No Message Provided");

return;

}

if (message.length() > 0) {

chatProxy.sendMessage(message);

}

request.getRequestDispatcher("/sendMessage.html").forward(request, response);

}

public void doUnsubscribe(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {

HttpSession session = request.getSession();

ChatServer chatServer = (ChatServer) session.getAttribute(CHAT\_SERVER\_ATTR);

if (chatServer == null) {

response.sendError(500, "ChatServer for this Session is null");

return;

}

String name = (String) session.getAttribute(NAME\_ATTR);

if (name == null) {

response.sendError(500, "Name for this Session is null");

return;

}

chatServer.unsubscribeUser(name);

request.getRequestDispatcher("/init.html").forward(request, response);

}

public void doGetMessage(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {

request.getRequestDispatcher("/getMessage.jsp").forward(request, response);

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

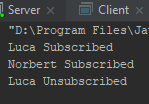
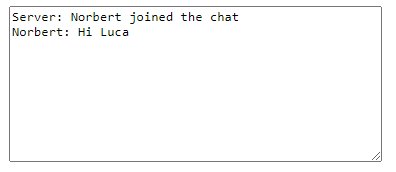
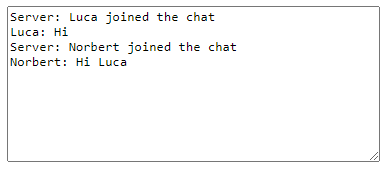
public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**Test eines funktionierenden Chats**:



Die Chats wurden in verschiedenen Browsern (verschiedene Sessions) gestartet. Pro Browser ist nur eine Session gegeben.

Gesamte Seite:

