Institut für Betriebssysteme und Rechnerverbund

Übungslösungen zur Vorlesung “Verteilte Systeme”, WS 02/03

http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/

Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de> · Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>

4 Interprozesskommunikation

4.1 TCP Client/Server mittels Java

TCPClient.java

```
//
// Institut für Betriebssysteme und Rechnerverbund
// Übungen zur Vorlesung "Verteilte Systeme", WS 02/03
// http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/
//
// Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de>
// Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>
//
// Aufgabe 4.1 - TCP Client/Server mittels Java
//
// Der Client könnte so aussehen...
//
//   Usage: javac TCPClient.java && java TCPClient hostname test.html
//

import java.lang.System;
import java.net.Socket;
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.PrintWriter;

public class TCPClient {

    public static void main (String args[]) {
        try {
            Socket socket = new Socket(args[0], 8000);

            BufferedReader in =
                new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
            PrintWriter out =
                new PrintWriter(socket.getOutputStream());

            String request = "GET " + args[1];
            out.println(request);
            out.flush();
            System.out.println("Sent Request: " + request);

            System.out.print("Received Response: ");
            String line;
            while ((line = in.readLine()) != null) {
                System.out.println(line);
            }

            socket.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

TCPServer.java

```
//
// Institut für Betriebssysteme und Rechnerverbund
// Übungen zur Vorlesung "Verteilte Systeme", WS 02/03
// http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/
//
// Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de>
// Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>
//
// Aufgabe 4.1 - TCP Client/Server mittels Java
//
// Der Server könnte so aussehen...
//
//   Usage: javac TCPServer.java && java TCPServer
//

import java.lang.System;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStreamReader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.util.StringTokenizer;

public class TCPServer {

    public static void main (String args[]) {
        try {
            ServerSocket listenSocket = new ServerSocket(8000);
            while (true) {
                Socket socket = listenSocket.accept();
                Connection conn = new Connection(socket);
            }
        } catch(Exception e) {
            System.out.println(e);
        }
    }
}

class Connection extends Thread {

    BufferedReader in;
    PrintWriter out;
    Socket socket;

    public Connection (Socket clientSocket) {
        try {
            socket = clientSocket;
            in = new BufferedReader(
                 new InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream());
            this.start();
            System.out.println("Spawned Connection.");
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    public void run() {
        try {
            char buf[] = new char[32768];
            int rc;
            String response;
            String request = in.readLine();
            System.out.println("Received Request: " + request);

            StringTokenizer st = new StringTokenizer(request);
            String method = st.nextToken();
            String uri = st.nextToken();
```

```
            try {
                st = new StringTokenizer(uri, "/");
                String basename = null;
                while (st.hasMoreTokens()) { basename = st.nextToken(); }
                FileReader file = new FileReader(basename);
                response = "HTTP/1.0 200 OK";
                out.println(response);
                out.println();
                do {
                    rc = file.read(buf);
                    if (rc > 0) {
                        out.write(buf);
                    }
                } while (rc >= 0);
                file.close();
            } catch (FileNotFoundException e) {
                response = "HTTP/1.0 404 Not Found";
                out.println(response);
                out.println();
            }

            out.flush();
            System.out.println("Sent Response: " + response);

            socket.close();
            System.out.println("Closed Connection.");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

5 Middleware

5.1 CORBA mittels Java

DocService.idl

```
//
// Institut für Betriebssysteme und Rechnerverbund
// Übungen zur Vorlesung "Verteilte Systeme", WS 02/03
// http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/
//
// Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de>
// Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>
//
// Aufgabe 5.1 - CORBA mittels Java
//
// Die Interface Beschreibung könnte so aussehen...
//
//   idlj -fall DocService.idl
//

module DocService
{
    interface Doc
    {
        exception NoSuchDocument{};

        string getURL(in string url) raises (NoSuchDocument);
    };
};
```

DocClient.java

```
//
// Institut für Betriebssysteme und Rechnerverbund
// Übungen zur Vorlesung "Verteilte Systeme", WS 02/03
// http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/
//
// Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de>
// Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>
//
// Aufgabe 5.1 - CORBA mittels Java
//
// Der Client könnte so aussehen...
//
//   Usage: javac DocClient.java
//          java DocClient -ORBInitialHost hansa -ORBInitialPort 9000 test.html
//

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

import DocService.*;


public class DocClient
{
    public static void main(String args[])
    {
        try {

            // Create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // Get the root naming context
            org.omg.CORBA.Object nsobj =
                orb.resolve_initial_references("NameService");
            NamingContext context = NamingContextHelper.narrow(nsobj);

            // Resolve the object reference in naming
            NameComponent component = new NameComponent("DocServer", "");
            NameComponent path[] = {component};
            Doc doc = DocHelper.narrow(context.resolve(path));

            // Call the Hello server object and print results
            String document = doc.getURL(args[args.length-1]);
            System.out.println(document);

        } catch(Exception e) {
            System.err.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
    }
}
```

DocServer.java

```
//
// Institut für Betriebssysteme und Rechnerverbund
// Übungen zur Vorlesung "Verteilte Systeme", WS 02/03
// http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/
//
// Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de>
// Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>
//
// Aufgabe 5.1 - CORBA mittels Java
//
// Der Server könnte so aussehen...
//
//   Usage: javac DocServer.java
//          tnameserv -ORBInitialPort 9000
//          java DocServer -ORBInitialHost hansa -ORBInitialPort 9000
//

import java.io.InputStreamReader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.StringTokenizer;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

import DocService.*;
import DocService.DocPackage.*;


public class DocServer
{
    public static void main(String args[])
    {
        try {

            // Create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // Create the servant and register it with the ORB
            DocServant servant = new DocServant();
            orb.connect(servant);

            // Get the root naming context
            org.omg.CORBA.Object nsobj =
                orb.resolve_initial_references("NameService");
            NamingContext context = NamingContextHelper.narrow(nsobj);

            // Bind the object reference in naming
            NameComponent component = new NameComponent("DocServer", "");
            NameComponent path[] = {component};
            context.rebind(path, servant);

            // Wait for invocations from clients
            java.lang.Object sync = new java.lang.Object();
            synchronized (sync) {
                sync.wait();
            }

        } catch(Exception e) {
            System.err.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
```

```
    }
}


class DocServant extends _DocImplBase
{
    public String getURL(String uri)
        throws NoSuchDocument
    {
        System.out.println("Received Method Incovation: getURL(" + uri + ")");

        char buf[] = new char[32768];
        String response = "";
        try {
            StringTokenizer st = new StringTokenizer(uri, "/");
            String basename = null;
            int rc;
            while (st.hasMoreTokens()) { basename = st.nextToken(); }
            FileReader file = new FileReader(basename);
            do {
                rc = file.read(buf);
                if (rc > 0) {
                    response += new String(buf);
                }
            } while (rc >= 0);
            System.out.println("Returning response");
            return response;
        } catch (IOException e) {
            System.out.println("Throwing Exception: NoSuchDocument");
            throw new NoSuchDocument();
        }
    }
}
```