Zur Leitseite Brecht
Zur Leitseite dieser Veranstaltung

Verteilte Systeme

Übung 2

Abgabe bis spätestens 02.06.14
Beachten Sie bitte die Hinweise zu den Testaten. Sie sind Bestandteil der Aufgabenstellung.

Arbeiten mit Sockets (Anbindung eines Web-Browsers)

Grobstruktur

Die Aufgabenstellung der Übung 1 wird dahingehend verändert, dass für den Dialog mit dem Benutzer ein Web-Browser zu verwenden ist. Das Suchverfahren nach Name und/oder Telefonnummer mit nebenläufig arbeitenden Threads bleibt unverändert.

Der Benutzer arbeitet an einem der Rechner im Übungsraum, z.B. am Host sun65. Dort öffnet er ein Terminalfenster und stellt darüber eine SSH-Verbindung zu einem anderen Host im gleichen Raum her, z.B. zu dem Host sun70. (**Beim Testat wird dieser Host vom Dozenten bestimmt!**) Über dieses Terminalfenster startet er seinen Telefonserver, im Beispiel auf der sun70, der an einem bestimmten Port, z.B. am Port 9876, auf Anforderungen wartet. Er gibt bei jedem Request, den er erhält, zugehörige Informationen im Terminalfenster aus, damit beim Testat sichtbar wird, was er gerade tut. (Das nennt man einen Trace ziehen.)

Läuft der Server, öffnet der Benutzer neben dem Terminalfenster (also im Beispiel auf der sun65) einen Web-Browser. Mit dessen URL-Zeile wendet er sich im Beispiel an http://sun70:9876. Dadurch wird der Server angesprochen und schickt dem Browser per HTTP eine HTML-Seite mit den Eingabefeldern für Name und Telefonnummer. Der Benutzer kommuniziert jetzt per Browser mit dem Server auf (im Beispiel) sun70.

Konkretisierung

1) Der Server schickt dem Browser als erste Seite eine, die folgendermaßen aufgebaut sein könnte (der genaue Aufbau ist Ihnen überlassen):

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html

<html>
<body>
<h2 align=center>Telefonverzeichnis</h2>
<h3>Sie können nach Name oder nach Telefonnummer oder nach beiden (nebenläufig) suchen.</h3>
<form method=get action="http://sun70:9876">
<table>
```

```
<tr> <td valign=top>Name:</td>    <td><input name=A></td>    <td></td> </tr>
<tr> <td valign=top>Nummer:</td> <td><input name=B></td>    <td></td> </tr>
<tr> <td valign=top><input type=submit value=Suchen></td>
      <td><input type=reset></td>
      <td><input type=submit value="Server beenden" ></td> </tr>
</table>
</form>
</body>
</html>
```

2) Der Telefonserver, das ist Ihr Java-Programm, das die Suchaufträge entgegennimmt, setzt sie wie in Übung 1 in Threads um, erzeugt aus den Antworten der Threads eine HTML-Seite und sendet diese dem Browser und gibt für das Testat einen Trace seiner Aktivitäten aus. Achten Sie darauf, dass diese Ausgaben aussagekräftig sind.

3) Schaffen Sie über die HTML-Antwortseite eine Möglichkeit, den Telefonserver zu beenden.

4) Das Ziel der Aufgabe ist die Beschäftigung mit Sockets. Der Browser enthält einen Client-Socket und verbindet ihn beim Abschicken einer Form mit einem Server-Socket an einem bestimmten Port eines Zielrechners. Im obigen Beispiel spricht der Browser das Programm am Port 9876 auf dem Host sun70 der Domain beuth-hochschule.de an.

Der Telefonserver, also Ihr Java-Programm, richtet einen Server-Socket an einem bestimmten Port (im obigen Beispiel 9876) ein und wartet in einer Endlosschleife (es ist kein Einmal-Server!) mit einem accept()-Aufruf auf die Verbindung mit einem Client. Portnummern müssen auf jedem Rechner eindeutig sein. Es sind 16-Bit große Integerzahlen ohne Vorzeichen, also kleiner gleich 65535. Die Ports bis 1.023 sind reserviert.

Der Browser sendet dem Server eine Reihe von Text-Informationen. Genauer gesagt, er sendet HTTP. Von all diesen Informationen ist für die Aufgabe lediglich die erste Zeile von Interesse. Liest der Telefonserver die erste Zeile z.B. mit einem BufferedReader aus dem Socket, erhält er einen String, dessen Aufbau das folgende Beispiel zeigt:

```
GET /?A=Meier&B=4711 HTTP/1.1
```

A und B sind beispielhafte Feldnamen aus der HTML-Form. Der Telefonserver wertet die GET-Zeile aus. Die Feld-Inhalte sind leicht zu isolieren. Kaufmanns-Und und Gleichheitszeichen sind Trenner. Beachten Sie jedoch die URL-Kodierung der Namen (z.B. bei Umlauten).

5) Suchen Sie nicht mit leeren Strings im Telefonverzeichnis! Es ist nicht akzeptabel, diese Fälle mit JavaScript abzufangen, da (ferne) Benutzer dieses Feature möglicherweise abgeschaltet haben.

6) Die Threads füllen einen Stack oder eine andere dynamische Datenstruktur. Der Telefonserver, der den oder die Threads startet, arbeitet die Suchergebnisse in eine HTML-Seite ein und schreibt diese in den Socket. Geben Sie auch **Negativmeldungen** auf diese Art aus.

7) Realisieren Sie auf jeder HTML-Ergebnisseite einen **Zurück-Button**, mit dem die Suchseite wieder geladen werden kann. Auch dafür ist JavaScript unzulässig.

8) Der Browser erwartet HTTP. Jedes Schreiben in den Server-Socket beim Telefonserver erfolgt zeilenweise und hat genau den Aufbau, der im Beispiel (unter 1)) gezeigt worden ist. Beachten Sie die Leerzeile vor <html>. Sie wird von HTTP als Header-Endezeichen gefordert. Beenden Sie die Folge von Schreibbefehlen in den Socket mit einem flush()-Aufruf an den entsprechenden Writer, damit die Puffer ausgeschrieben werden.

9) Beim Testat müssen Server und Browser auf getrennten Maschinen laufen. Starten Sie von dem Rechner aus, auf dem der Browser läuft, Ihren Server per SSH-Verbindung auf einem anderen Host des Übungsraumes.

10) Für das Testat müssen im Telefonverzeichnis wenigstens ein Name und eine Nummer mehrfach vorkommen. Weiterhin muss wenigstens ein Name einen Umlaut enthalten und wenigstens einer eine Wortlücke (wie bei 'von Reibach').