Java Standard: Netzwerkprogrammierung

Java ist eine der ersten Programmiersprachen welche von Grund auf mit Netzwerkunterstützung entwickelt wurde. Es waren auch die Applets welche Java zum Durchbruch in der Programmierwelt verhalfen.

Eine der wichtigsten Neuerungen in Java war, dass man einen sehr geringen Aufwand als Programmierer leisten muss, um eine Software netzwerkfähig zu machen. Die "Magie" in Java (oder konkret die Socketklasse) erledigt den Hauptteil der Arbeit. Jeder der schon einmal einen kleinen Server / Client in C geschrieben hat, kann davon berichten wie aufwendig dies sein kann. Im Gegensatz dazu bietet das java.net.* package alle Grundlagen, die es braucht um in 5 Minuten ein kleines Netzwerkbasierendes Programm zu schreiben.

Grundlagen der Netzwerkprogrammierung

In diesem Kapitel werden die Grundzüge netzwerkbasierter Software, Protokolle, Sockets, Streams etc. erläutert

Sockets für Clients

In diesem Kapitel wird beschrieben, wie ein Programm Verbindung mit einem Server aufnimmt und Daten mit diesem austauschen kann.

In den meisten Fällen geschieht die Datenübertragung zwischen Server und Host auf der Basis von TCP. Die java.net.Socket Klasse stellt uns eine solche TCP - Verbindung zur Verfügung. Es sind jedoch nicht alle Ports eines Betriebssystems für die Datenübertragung auf Basis von TCP ausgelegt. Um zu wissen auf welche Ports wir eine TCP Verbindung öffnen könnten, können wir uns mit Java einen eigenen kleinen Portscanner "basteln".

```
1  import java.net.*;
2  import java.io.*;
3
4  public class PortScanner {
5
6  public static void main(String[] args) {
7
8    String host = "localhost";
9
10   if (args.length > 0) {
11    host = args[0];
12   }
13   for (int i = 1; i < 1024; i++) {
14     try {
15       Socket s = new Socket(host, i);
16       System.out.println("There is a server on port " + i + " of "
17        + host);
18     }
19     catch (UnknownHostException e) {
20       System.err.println(e);
21       break;
22     }
23     catch (IOException e) {
24       // must not be a server on this port
25     }
```

```
26   }
27  }
28 }
```

Das Programm testet alle Ports im Bereich 1 - 1024. Ist auf einem dieser Ports ein TCP - fähiger Server am "lauschen", so wird eine entsprechende Meldung ausgegeben. Wichtig an diesem Programm ist Zeile 15. Hier wird einer der Konstruktoren der Klasse Socket gerufen.

Hier die schnellere Variante:

```
1 package PortScanner;
2
3 /*
4  * Der Portscanner ist ein gutes Beispiel für den sinnvollen Einsatz
von Threads.
5  * Ansonsten müsste man bei jedem Port zu dem die Connection nicht
hinhaut auf einen
6  * Timeout warten bevor man weitermacht. Dieser Timeout ist zwar
nicht sehr lang,
7  * aber wenn man 1000 Ports testet lange genug.
8  */
9
10 public class PortscannerMain {
11
12      public static void main(String[] args) {
13            String host = "localhost";
14            for (int i = 0; i < 1023; i++) {
15                  Connthread curr = new Connthread(host, i);
16                  Thread th = new Thread(curr);
17                  th.start();
18                  try {
19                        th.sleep(1);
20                  }
21                  catch (InterruptedException ex) { }
22            }
23      }
24 }
```

```
1 package PortScanner;
2
3 import java.io.IOException;
4 import java.net.*;
5
6 public class Connthread implements Runnable {
7      int i;
8      String host;
9      public Connthread(String host, int i) {
10            this.i = i;
11            this.host = host;
```

```
12      }
13
14      public void run() {
15            try {
16                  Socket target = new Socket(host, i);
17                  System.err.println("Connected to " + host + " on
Port " + i);
18                  target.close();
19            }
20            catch (UnknownHostException ex) {
21                  System.out.println("Unkown Host " + host);
22            }
23            catch (IOException ex) {System.out.println(i);};
24      }
25
26 }
```

Sockets für Server

In diesem Kapitel wird beschrieben wie man eine Server schreibt, welcher auf eingehende Verbindungen "hört" und auf die Anfragen antwortet.

Quellen und Bearbeiter des Artikels

Java Standard: Netzwerkprogrammierung *Quelle*: http://de.wikibooks.org/w/index.php?oldid=417111 *Bearbeiter*: Bastie, Stefan Majewsky, 4 anonyme Bearbeitungen

Lizenz