

Netzwerkprogrammierung



Netzwerkverbindungen

- Das Entwurfsziel von Java war: Einfache Verbindung zwischen Rechnern und SetBox-Systemen.
- Das Standardpaket `java.net` hilft bei allen Netzwerkverbindungen.
- Mit Datenströmen wird das Lesen/Schreiben so leicht wie der Zugriff auf das Dateisystem.
- Verbindung zu Servern und Abrufen von Daten mit gewöhnlichen Web-Protokollen und Sockets.



Internet-Adressen



IP-Adressen und Rechnernamen

- Jeder Rechner ist im Internet durch eine Adresse identifiziert.
 - ▶ Man nennt diese **IP-Adresse**.



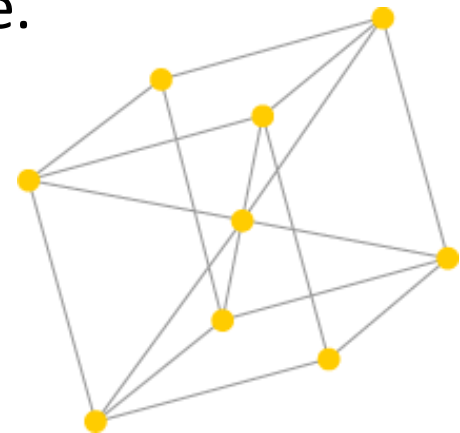
- IP Adressen sind nicht leicht zu merken. Daher stehen stellvertretend für Adressen einfache Namen.
- Ein spezieller Dienst wandelt den Namen in eine IP-Adresse um.
 - ▶ Dieser Dienst ist der Namensdienst und wird von einem DNS-Server übernommen.

InetAddress

- Der Umgang mit IP-Adressen übernimmt eine Java-Klasse `java.net.InetAddress`.
- Die Klasse `InetAddress` hat keine öffentlichen Konstruktoren, nur einige statische Methoden
- `InetAddress getLocalHost()` liefert die IP-Adresse des lokalen Rechners, oft (127.0.0.1)
- `InetAddress getByName(String host)` liefert die IP-Adresse eines gegebenen Hosts.
 - ▶ Der Parameter `host` ist entweder der Maschinenname (»tutego.com«) oder eine String-Repräsentation der IP-Adresse (»209.204.253.212«).
- Kann die IP-Adresse nicht gefunden werden, so lösen die Methoden eine `UnknownHostException` aus.

Abfragen über InetAddress

- Die Objektmethoden stellen Anfragen an das `InetAddress`-Objekt:
 - ▶ `byte[] getAddress()`
Liefert die IP-Adresse als Bytefeld.
 - ▶ `String.getHostAddress()`
Liefert einen String in der Form A.B.C.D
 - ▶ `String getHostName()`
Liefert den Hostnamen für die Adresse.

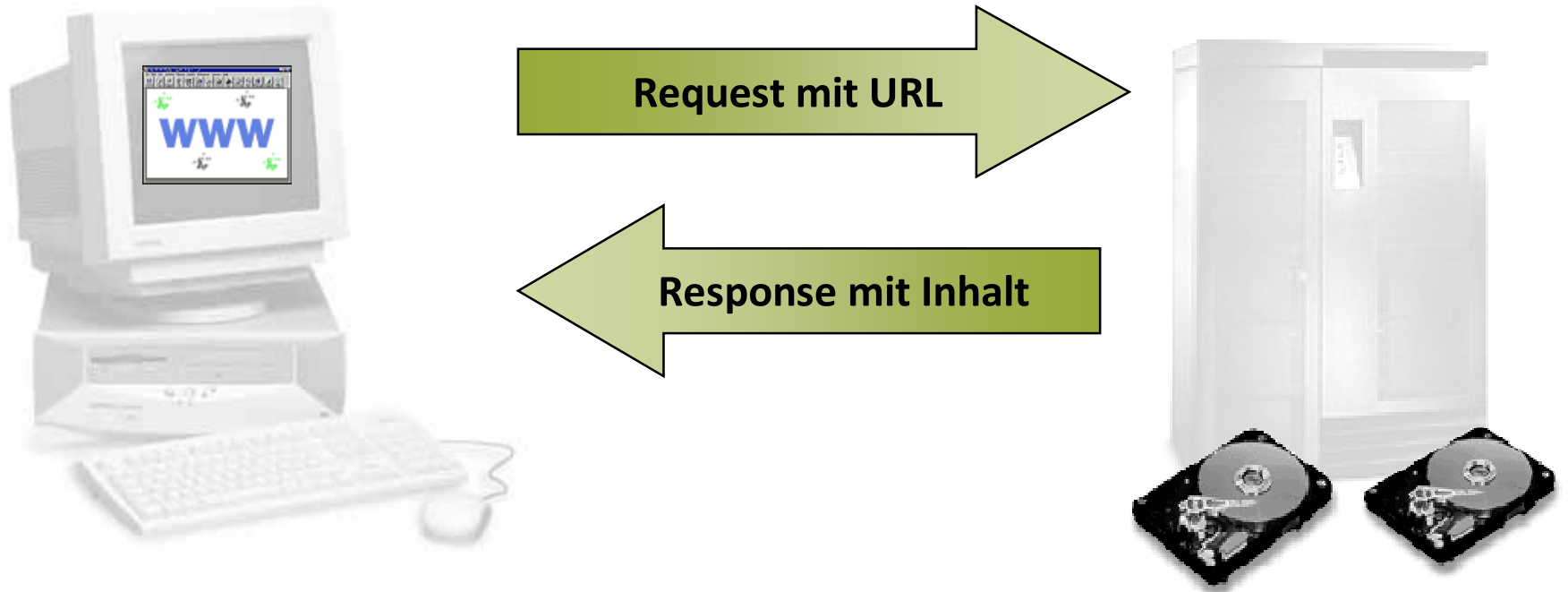


URL-Verbindungen



Anfragen an einen Web-Server

- Anfragen werden über das Protokoll HTTP an den Server geschickt.



URL-Verbindungen

- Eine URL (RFC 1738) ist das Adressenformat für eine Ressource im World Wide Web.
- Die URL (Uniform Resource Locater) enthält
 - ▶ Namen des benutzten Schemas (Protokoll) und
 - ▶ eine Spezialisierung



- Anders gesagt: Eine URL beinhaltet Methode des Zugriffes (HTTP, FTP) und Ort.

URL-Objekte

- URL-Objekte repräsentieren Hostnamen, Port und Dateiname.

```
String s = "http://www.java-tutor.com/index.html";  
try  
{  
    URL url = new URL( s );  
} catch ( MalformedURLException e ) { }
```

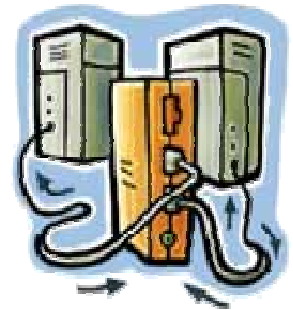
- Die Konstruktoren lösen eine `MalformedURLException` aus,
 - ▶ wenn der Parameter im Konstruktor `null` ist oder
 - ▶ das URL-Objekt ein unbekanntes Protokoll beschreibt.

Verbindungen zum Netz

- Die Klasse `URL` definiert die Methode `openStream()`, die ein `InputStream`-Objekt liefert.
 - ▶ Damit wird eine Netzwerkverbindung mit dem `URL`-Objekt tatsächlich geöffnet.
 - ▶ Die Methode kann eine `IOException` auslösen!
- Um aus dem `InputStream` zu lesen, bietet sich `Scanner` an.
 - ▶ Eine Kombination von `hasNextLine()` und `nextLine()` liefert alle Zeilen.
 - ▶ Auch hilft der `Scanner` bei einer speziellen Konvertierung.

Daten auslesen

```
InputStream is = null;
try {
    is = url.openStream();
    Scanner s = new Scanner( is );
    while ( s.hasNextLine() )
        System.out.println( s.nextLine() );
}
catch ( IOException e ) {
    System.out.println( e );
}
finally { try { is.close(); } catch(Exception e) {} }
```



Einen Internet-Proxy nutzen

- Der Internetverkehr größerer Firmen geht in der Regel durch einen Proxy.
- Ein Proxy-Server lässt sich in Java durch spezielle Property-Variablen zuweisen:

```
System.setProperty( "proxySet", "true" );
```

```
System.setProperty( "proxyHost", "proxy" );
```

```
System.setProperty( "proxyPort", "81" );
```

CGI-Anfragen losschicken



Suchmaschinen mit dem Browser

- Benutzung der Suchmaschine <http://www.google.de>.

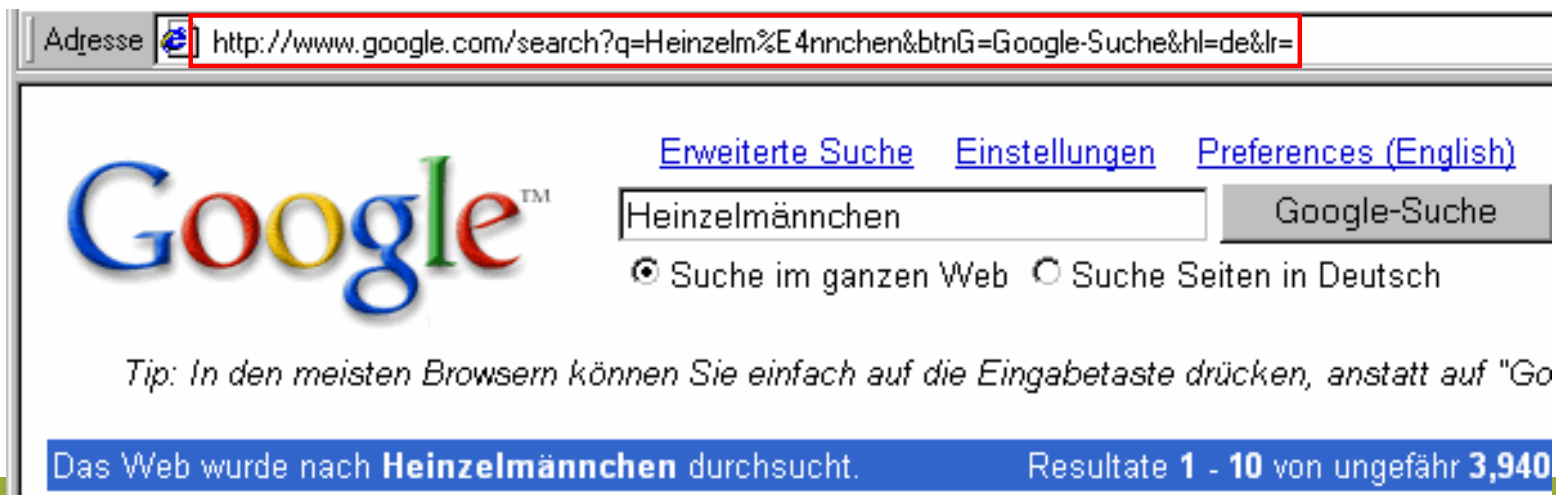


Heizelmännchen [Erweiterte Suche](#) [Einstellungen](#)

Google-Suche Auf gut Glück!

Suche im ganzen Web Suche Seiten in Deutsch

- Der Browser generiert daraus eine spezielle URL, die er zum Server schickt.



Kodieren für CGI-Programme

- Wenn aus HTML-Datei mit Formularen Daten an das CGI-Programm übermittelt werden, dann werden diese Daten kodiert.
- Viele Zeichen sind in der URL nicht erlaubt. Etwa:
 - ▶ Leerzeichen werden durch Plus-Zeichen kodiert.



- Die statische Methode `encode()` der Klasse `java.net.URLEncoder` führt eine Konvertierung durch.

```
String s = URLEncoder.encode("%mi t%Prozenten");
```


Eine Suchabfrage formulieren

- Wenn wir das Abschicken eines HTML-Formulars für eine Suchabfrage nachbilden wollen, dann müssen wir alle Informationen in den Felder sammeln und codieren.
- Anschließend wird der codierte Inhalt hinter die URL des CGI-Programmes gehängt.
 - ▶ Die Parameter werden durch ein Fragezeichen getrennt.

Ran an Google

```
String s = "";

for ( int i = 0; i < args.length; i++ )
    s += args[i] + " ";

s = "q=" + URLEncoder.encode( s.trim() );

try {
    URL url = new URL(
        "http://www.google.com/search?" + s );

    InputStream in = url.openStream();
    ...
}
```

Netzwerk-Clients



Sockets

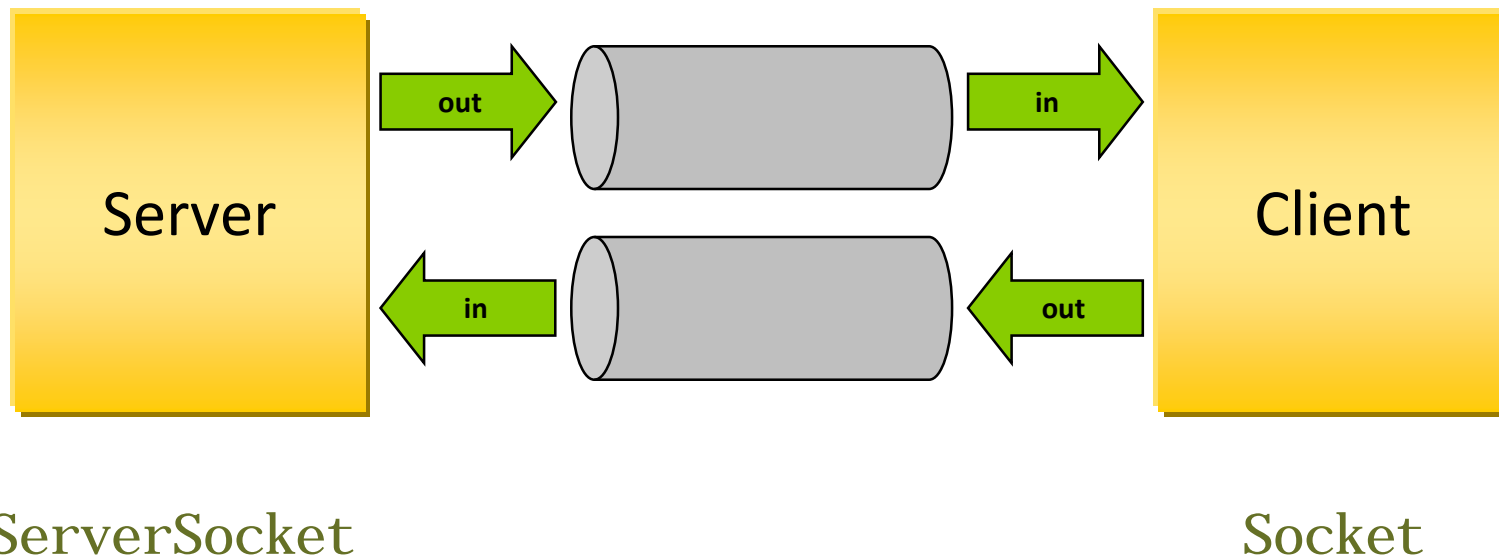
- URL-Verbindungen sind schon High-Level Verbindungen.
- Die Rechner, die im Internet verbunden sind, kommunizieren über Protokolle (bsp. TCP/IP).
- Ein Socket dient zur Abstraktion und ist ein Verbindungspunkt in einem TCP/IP Netzwerk.
- Wer Daten
 - ▶ empfängt (Client) öffnet eine Socket-Verbindung zum Horchen und
 - ▶ derjenige der sendet, öffnet eine Verbindung zum Senden (Server).

Was ist ein Port?

- Für jeden Dienst (Service), den ein Server zur Verfügung stellt, gibt es einen Port.
 - ▶ Alltagsbeispiel: Es reicht nicht aus, die Adresse der Verwaltung in der Stadt zu kennen. Ich brauche auch noch eine Zimmernummer.
- IP-Adressen und Ports werden nicht willkürlich, sondern von der IANA (Internet Assigned Numbers Authority) vergeben.
- Die Portnummer ist eine 16 Bit Zahl, eingeteilt in
 - ▶ System 0-1023 (›Well-Known‹, Contact) und
 - ▶ Benutzer 1024-65535.
- Komplette Liste ist unter <ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers>

Sockets

- Mit einem **Socket** baut man eine feste Verbindung zu einem Rechner auf, die für die Dauer der Übertragung bestehen bleibt.



- Was für den Server Ausgabe-/Eingabestrom ist, ist für den Client Eingabe-/Ausgabestrom.

Unter Spannung: Die Ströme

- Auf der Client-Seite bauen wir zuerst ein `Socket`-Objekt auf. Der Server soll auf Port 80 kontaktiert werden.

```
Socket clientSock;
```

```
clientSock = new Socket( "die.weite.welt", 80 );
```

- Die Klasse `Socket` bietet die Methoden `getInputStream()` und `getOutputStream()`, die einen Zugang zum Datenstrom erlauben.
- Um die Performanz zu heben, können wir einen gepufferten `OutputSteam` nutzen:

```
InputStream is = server.getInputStream();
```

```
BufferedReader in;
```

```
in = new BufferedReader( new InputStreamReader(is) );
```

Server implementieren



Client-Server Kommunikation

- Ein Server durch die Klasse `ServerSocket` repräsentiert.
- Wollen wir Server programmieren, erzeugen wir ein `ServerSocket`-Objekt mit einem Konstruktor, dem wir einen Port Parameter übergeben.

```
try
{
    ServerSocket socket = new ServerSocket( 1234 );
}
catch ( IOException e ) { /* belegt? */ }
```

Auf Verbindungen horchen

- Mit `accept()` der Klasse `ServerSocket` akzeptieren wir eine eingehende Verbindung.

```
Socket client = serverSocket.accept();
```

- Nun können wir mit dem zurückgegebenen Client-Socket genauso verfahren wie mit dem schon programmierten Client.

Die 6 Schritte

- ❶ Ein Server-Socket erzeugen der horcht
- ❷ Mit der `accept()`-Methode auf neue Verbindungen warten
- ❸ Ein- und Ausgabestrom vom zurückgegebenen Socket erzeugen
- ❹ Mit einem definierten Protokoll die Konversation unterhalten
- ❺ Stream vom Client und Socket schließen
- ❻ Bei Schritt 2 weitermachen oder Server-Socket schließen

Server

```
import java.net.*;    import java.io.*;

class Server {
    public static void main( String args[] ) throws IOException {
        ServerSocket server = new ServerSocket( 3141 );

        while ( true ) {
            Socket client = server.accept();

            InputStream in  = client.getInputStream();
            OutputStream out = client.getOutputStream();

            int zahl1 = in.read();    // -128..+127
            int zahl2 = in.read();

            out.write( zahl1 * zahl2 );
            client.close();
        }
    }
}
```

Client

```
class Client
{
    public static void main( String args[] ) throws IOException
    {
        Socket server = new Socket ( "localhost", 3141 );

        InputStream in = server.getInputStream();
        OutputStream out = server.getOutputStream();

        out.write( 4 );
        out.write( 9 );

        int result = in.read();
        System.out.println( result );

        server.close();
    }
}
```