10 HTTP- und TCP/IP-Protokolle

HTTP ist der Standard zur Übertragung von Nachrichten über ein TCP/IP-Netzwerk. XML ist ein Standard zur Codierung von Daten in einen ASCII-Text mit dem Ziel, diese Daten über ein Netzwerk zu übertragen. SOAP wiederum sind in XML codierte Aufrufe von Methoden eines Objekts beziehungsweise von Funktionen auf einem entfernten Rechner über das Netzwerk.

10.1 TCP/IP und UDP/IP Network Protocol

Das TCP/IP-Protokoll (korrekt ausgesprochen: TCP-over-IP-Protokoll) und das verwandte UDP/IP-Protokoll sind die einzigen überlebenden Standards im Client-Server-Computing, die noch von nennenswerter Bedeutung sind.

Applikation	Protokoll	Session	Transport	Netzwerk	Data Link	Physikalisch
E-Mail	SMTP/ POP	Port 25	TCP	IP	SLIP	ISDN
Webapplication	HTTP	Port 80			PPP	ADSL
File-Transfer	FTP	Port 21				
SAP RFC	RFC	Port 1080				
Host-Sessions	TELNET	Port 23			802.2	Wireless
Directory Service	DNS	Port 53	UDP			FDDI
File-Service	NFS	RPC-Mapping			Ethernet II	Coax

Tabelle 10.1 Open-Systems-Interconnection-(OSI)-Seven-Layer-Modell

TCP/IP ist der Standard

TCP/IP ist somit der De-facto-Standard für Computernetzwerke. Zwar gibt es sehr viele andere Level-4-Protokolle, zum Beispiel Microsofts *NETBIOS*, IBMs *NETBEUI* oder Novells *IPX*, jedoch haben diese fast keine Bedeutung mehr und existieren allenfalls aus Gründen der Kompatibilität mit bestehenden Anwendungen, wenn diese noch kein TCP/IP unterstützen. In der Regel wird dieses Protokoll nur noch emuliert, indem die Daten des gewünschten Protokolls auf einem TCP/IP-Protokoll »Huckepack« genommen werden.

IP, Internet Protocol

IP bedeutet ganz einfach *Internet Protocol*. Es wurde ursprünglich als ausfallsichere Transportschicht für die Implementierungen des Internet geschaffen. IP ist ein Level-3-Protokoll und bestimmt, wie die Bytes des Datenstroms zu Paketen zusammengefasst werden und auf welchem Weg ein Paket zum Empfänger übertragen wird. Ohne ins Detail zu gehen, kann man sagen, dass IP für die Zustellung der Datenpakete verantwortlich ist.

Eindeutige Nummer für jeden Knoten

Das typische Kennzeichen des IP-Netzwerks sind seine Adressen, die jedem Knoten innerhalb eines Netzwerks zugeordnet sind. Diese Adressen sind 4-Tupel von Zahlen im Bereich 0–255, zum Beispiel 192.168.1.1. Eine solche Adresse ist immer eindeutig innerhalb eines Netzwerks. Die kleinste IP-Adresse ist demnach 0.0.0.1 und die größte 255.255.255.254. Es ist technisch nicht möglich, alle Bits auf 0 oder alle auf 1 zu setzen.

TCP, Transmission Control Protocol

Das TCP-Protokoll liegt eine Stufe höher auf der vierten Ebene des OSI-Modells (Transport Layer). Es legt die logischen Einheiten fest, in denen Daten vom Sender zum Empfänger übertragen werden. Dabei spielt eine wesentliche Rolle, dass IP zum Senden keine feste Verbindungsstrecke zwischen Sender und Empfänger aufbaut, sondern die einzelnen Datenpakete grundsätzlich über verschiedene Strecken anliefern kann. TCP ist dann dafür verantwortlich, die Pakete auf Senderseite zu zerlegen und beim Empfänger wieder in der richtigen Reihenfolge zusammenzubauen.

UDP, User Datagram Protocol

Das Senden von Datenpaketen in beliebiger Reihenfolge garantiert aus vielerlei Gründen eine hohe Zuverlässigkeit vor allem bei langen Verbindungsstrecken mit vielen Hops. Wenn die Datenmengen allerdings klein sind und die Verbindung zwischen benachbarten Rechnern erfolgen soll, ist der zusätzliche Aufwand oft störend oder im Falle von sicheren Übertragungen auch unerwünscht. Deshalb existiert parallel noch UDP (*User Datagram Protocol*). Hierbei handelt es sich um ein abgespecktes TCP-Protokoll, das Daten in einem Stück überträgt.

IP-Adresse identifiziert Computer

Jeder Computer in einem Netzwerk hat seine eindeutige IP-Adresse, zum Beispiel 127.0.0.1 oder 169.128.1.1. Genau genommen muss man sagen, mindestens eine Adresse, denn ein Rechner kann auch mehrere IP-Adressen gleichzeitig haben, zum Beispiel kann er zur selben Zeit über eine Netzwerkkarte und über ein Modem mit dem Internet verbunden sein. Mit dieser Adresse identifizieren wir die Hardware, über die die Kommunikation abläuft.

Wenn die Daten über das Netzwerk angeliefert werden, müssen sie aber auch innerhalb des Rechners verarbeitet werden. Dies geschieht durch die Zuweisung von Ports zu den Applikationen. Das IP-Protokoll kennt pro IP-Adresse 256×256 = 65.235 IP-Ports. Jedem Port kann eine eigene Anwendung zugewiesen werden, die den Datenstrom entgegennimmt und verarbeitet.

Listener

Ein solches Programm, das an einem Port einer IP-Strecke auf Daten wartet, nennt man *Listener*, denn diese Programme »horchen« an den Ports und hören die Nachrichten ab. In Tabelle 1.2 sind eine Reihe bekannter Listener-Dienste aufgeführt.

Protocol	Port
FTP	21
TELNET	23
HTTP	80
POP	110
HTTPS	443
SAP R/3	1023
LPD or SAPLPD	515

Tabelle 10.2 Typische Port-Zuweisungen in einer IP-Umgebung

10.2 Client-Server-Protokolle

Ein TCP/IP-Netzwerk ist ein vielseitiges Transportprotokoll, über das sehr verschiedene Arten von Webdiensten ausgetauscht werden. An den beiden Enden einer IP-Kommunikationsstrecke – den Ports – können sich die unterschiedlichsten Programme befinden, die auf ankommende Nachrichten und Anfragen warten.

10.2.1 Clientseite

Beispiele für Programme auf der Clientseite, also dem Ende einer TCP/IP-Strecke, das eine Anfrage initiiert, sind:

- Webbrowser, z. B. Internet Explorer, Netscape oder Opera
- Microsoft-Office-Komponenten, etwa MS Excel, MS Word oder MS Project
- SAP R/3 via RFC und HTTP oder direkt über den Web Application Server
- jede andere Software, die HTTP-Requests absetzen kann

10.2.2 Serverseite

Auf der Serverseite sieht es noch etwas komplizierter aus. Dort werden Ressourcen verwaltet, die dem anfragenden Client zur Verfügung gestellt werden können. Eine solche Ressource kann im Grunde ein beliebiges Programm sein, das eine API zur Verfügung stellt, die vom Webserver aufgerufen werden kann.

Limitierungen bei Webservern

Die Einschränkungen liegen dabei vor allem in den Möglichkeiten des Webservers. Zum Beispiel kann ein Microsoft Internet Information Server eine Java-Applikation zunächst nur eingeschränkt ausführen. Andererseits tun sich die Open-Source-Webserver wie Tomcat noch schwer, eine Microsoft.NET-Anwendung auszuführen oder in einfacher Weise mit einem Microsoft-Office-Produkt zu kommunizieren.

10.2.3 URN, URI und URL – Resource Locator

Adressen im Web werden durch eine URN angegeben

Um eine Kommunikation zwischen einem Requester und einem Server zu gewährleisten, müssen Nachrichten zuverlässig von einem Ende der Kommunikationsstrecke zum anderen Ende übermittelt werden. Dazu wird jede Nachricht in einen formalen Umschlag gepackt und dieser mit der Bestimmungsadresse versehen. Diese Adresse nennt man *Resource Locator*.

URL – Uniform Resource Locator

URL bezeichnen Dateien

Eine URL ist die häufigste Form einer Ressourcenangabe in Webdiensten. Damit gibt der Client genau den Namen und den Typ der gewünschten Ressource an. Diese Ressourcen können selbst wieder sehr vielfältig sein, in der Regel unterscheidet man aber durch die Angabe, in welchem Protokoll die Ressource die Information übermittelt.

URI-Präfix	Beispiel	Ressource
HTTP	*http://logosworld.com/index.htm*	HyperText Transfer Protocol, Verweis auf eine Website
FTP	*ftp://ftp.microsoft.com/*	File Transfer Protocol, Verweis auf einen FTP-Server
File	*file://C:/autoexec.bat*	Verweis auf eine Datei in der Syntax des Betriebssystems des Servers
Urn	*urn:sap-com:document:sap:business:rfc*	Referenz auf den SAP Business Connector

Tabelle 10.3 Beispiele für Präfixe einer URI

URI – Uniform Resource Identifier

URI = URL mit Parametern

Eine URI ist eine um zusätzliche Informationen an den HTTP-Server erweiterte URL. Dabei handelt es sich in der Regel um Parameter, die von der gerufenen Ressource zur weiteren Verarbeitung verwendet werden. Das weiter unten beschrieben CGI-Interface basiert vollständig auf den parametrisierten URIs. Eine typische URI kann wie folgt aussehen:

```
http://www.logosworld.de/asp/testdrive.htm?user=micky
&pass=mausi
```

Diese URL besteht aus mehreren Teilen:

- **Ressource Type** `http://`
  Dieser teilt dem Server mit, welches Protokoll für die Kommunikation verwendet werden soll.
- **Resource Address** `www.logosworld.de/asp/testdrive.htm`
  Dies ist der eindeutige Name der Ressource. Normalerweise handelt es sich dabei einfach um den Netzwerkdateinamen des angeforderten Dokuments. Allerdings kann es sich dabei um einen beliebigen String handeln, der vom server verstanden wird. Gewöhnlich bestimmt der Server aus dem Suffix (z. B. .htm, .txt, .asp, .php .cgi etc) die Art der angeforderten Ressource.
- **Resource parameters** `?user=micky&pass=mausi`
  In diesem Fall sind das Username und Passwort.

URN – Uniform Resource Name

URN verweist auf eine ständig verfügbare oder virtuelle Ressource

Eine URN verweist auf eine ständige Verbindung innerhalb eines Netzes. Wir sprechen von einer URN, wenn die angegebene Ressource durch einen logischen Namen spezifiziert wird, während im Unterschied dazu eine URL immer angibt, wo die Ressource innerhalb des Web zu finden ist. Eine solche typische URN ist z. B. der Verweis auf den SAP Business Connector:

```
urn:sap-com:document:sap:business:rfc
```

Das Arbeiten mit URNs setzt voraus, dass die Kommunikation vom Client über ein intelligentes Gateway abgewickelt wird, das erkennen kann, an welche IP-Adresse die Anfrage physikalisch geleitet werden kann.

URNs können sich selbst am Gateway registrieren

Ein typisches Verfahren, um dies zu gewährleisten, ist der passive Gateway-Modus. Dabei meldet sich die Ressource an dem interessierten Gateway an und registriert sich dort. In diesem Fall wird die Verbindung

also nicht durch das Gateway initiiert, sondern durch die Ressource selbst aufgebaut. Dabei wird ein permanenter Kanal geöffnet, über den das Gateway alle Anfragen an die Ressource leitet.

SAP Gateway-Modus

Ein solches URN-Verfahren wendet auch SAP R/3 an, wenn es eine RFC-Verbindung im Gateway-Modus aufbaut. Näheres finden Sie hierzu an anderer Stelle in diesem Buch (siehe Abschnitt 3.6) und unter den Transaktionen SMGW und SM59.

10.2.4 CGI – Common Gateway Interface

CGI ist übliche Form der Parameterangabe

Eine verbreitete Form der URI-Angabe wird durch das *Common Gateway Interface* (CGI) spezifiziert. Die Idee hinter CGI ist der Wunsch, mit einer HTTP-Zieladresse ein Programm aufzurufen, das dann eine HTTP-Response dynamisch erstellt. Der Aufruf des Programms erfolgt dabei einfach durch Angabe des Programmnamens. Falls das Programm Parameter erwartet, werden diese als String durch ein Fragezeichen getrennt an die URL angehängt. Das folgende Beispiel ruft das SAP Interface Repository auf und gibt an, dass die gewünschte Anzeigesprache `DE`, also deutsch sein soll:

```
http://ifr.sap.com/catalog/query.asp?language=DE
```

Die Parameter werden immer in der folgenden Form angegeben:

```
Parametername=parameterwert
```

Falls mehrere Parameter spezifiziert werden sollen, werden sie durch ein Ampersand (&) voneinander getrennt. Ein Beispiel sieht so aus:

```
http://ifr.sap.com/catalog/query.asp?namespace=
urn:sap-com:ifr:LO:46C&type=bapi
&name=SalesOrder.GetList&xml=schema
```

10.3 HTTP-Kommunikation mit einem Webserver

Direkt über dem TCP/IP_Protokoll hat sich das HTTP-Protokoll als der bedeutendste Standard etabliert. Er verdrängt bisher konkurrierende Protokolle wie GOPHER entweder ganz oder aber auf Nischenplätze wie z. B. SMTP für Mailanwendungen. Auf HTTP setzen dann höhere Protokolle auf, die in erster Linie dazu dienen, Daten eine Semantik zu geben, zum Beispiel durch HTML oder XML. Abbildung 10.1 zeigt, wie die Protokolle sich übereinander »zwiebeln«.

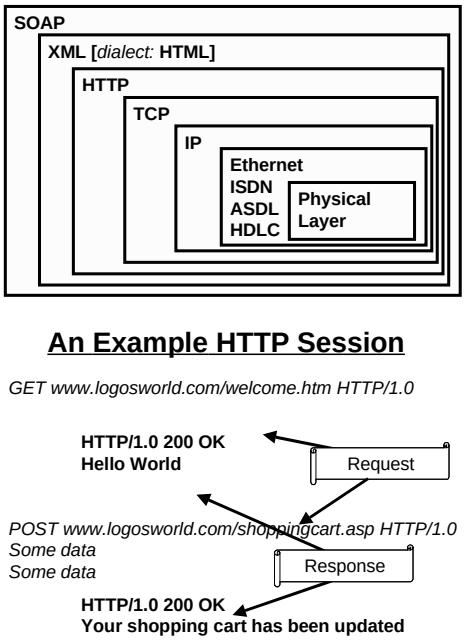

Abbildung 10.1 Hierarchie der Protokoll-Layer

Ein HTTP-Service und ein Webbrowser sind typische Client-Server-Anwendungen, wobei im Normalfall (aber nicht immer) der Browser die Rolle des Clients und der HTTP-Service die des Servers übernimmt. Beide, Client und Server, bauen eine bidirektionale Verbindung miteinander auf und benutzen dazu ein der Aufgabe angemessenes Protokoll, das über TCP/IP ausgeführt wird.

ASCII-Klartext

HTTP ist ein reines ASCII-Protokoll, wobei jedes übertragene Byte als alphanumerisches ASCII-Zeichen interpretiert wird. Dadurch wird das Protokoll von Menschen in Klartext lesbar, ist aber gleichzeitig auf einfache Weise von jedem beliebigen Computer, von Palm oder PC bis Mainframe, zu verarbeiten.

Frage-&-Antwort-Protokoll

HTTP ist gewöhnlich ein Dialog zwischen einem Browser-Client und einem Webserver. Dabei sendet der Client Anfragen an den Server, der darauf antwortet. Eine Anfrage im Sinne eines Fire-and-Forget ist in HTTP nicht vorgesehen, das heißt, es muss immer eine Antwort auf die Anfrage kommen, um zumindest den Erhalt der Nachricht zu quittieren. Diese Quittierung, englisch *acknowledgement*, ist erforderlich, da die Reisezeiten einer Datenübertragung im Internet nicht vorhersehbar sind.

HTTP-Requests: GET und POST

HTTP unterscheidet zwischen einer Reihe von Request-Typen, wobei die wichtigsten `HTTP GET` und `HTTP POST` sind.

Listing 10.1 Einfaches Beispiel für HTTP GET

```
GET /TimeService/TimeService.asmx/getUTCTime? HTTP/1.1
Host: www.nanonull.com

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://www.Nanonull.com/TimeService/">
   string
</string>
```

Listing 10.2 Einfaches Beispiel für HTTP POST

```
POST /TimeService/TimeService.asmx/getUTCTime HTTP/1.1
Host: www.nanonull.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://www.Nanonull.com/TimeService/">
   string
</string>
```

Die folgende Tabelle erläutert die Bedeutung grundlegender HTTP-Befehle.

HTTP-Kommando	Beschreibung
`HTTP GET`	Sendet die Adresse einer gewünschten Ressource (URI) als String zum Server.
`HTTP POST`	Sendet zusätzlich zur Adresse einer gewünschten Ressource (URI) auch noch weitere Daten im Body der Anfrage mit. Somit besteht ein `HTTP POST` aus zwei Teilen: aus einem Data-Header und dem Data-Body.

Tabelle 10.4 HTTP-Kommandos

HTTP-Kommando	Beschreibung
`HTTP PUT`	Ersetzt die als URL angegebene Ressource (Datei) durch die im Rumpf des HTTP-Requests mitgesandten Daten. Dies entspricht einem Upload von Daten auf den Webserver und funktioniert natürlich nur, wenn der Client auch die notwendigen Schreibrechte auf dem Server besitzt.
`HTTP HEAD`	Gibt den Header der als URN angegebenen Ressource zurück. Dies ist nützlich, um die Existenz einer Ressource vorab zu testen oder die Größe des zu erwartenden Datenstroms abzufragen.

Tabelle 10.4 HTTP-Kommandos (Forts.)

HTTP-Forms werden im Body versendet

Typische Daten im Body einer HTTP-Anfrage sind die Daten einer HTML-Form, das heißt die Inhalte der Eingabefelder innerhalb eines HTML-Dokuments, die unter anderem zwischen den Tags `<FORM>…<INPUT>…</INPUT>…</FORM>` zu finden sind.

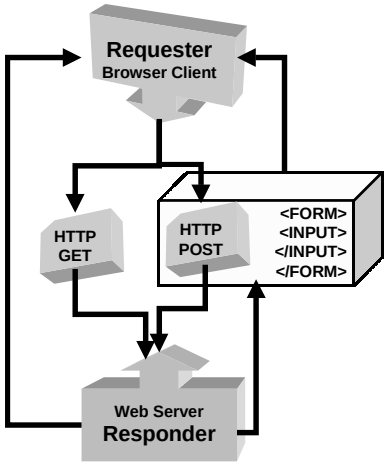

Abbildung 10.2 Kommunikation zwischen einem HTTP-Browser und einem Webserver

10.3.1 Beispiel einer HTTP-Session

Kommunikation wird immer vom Client aufgebaut

Die Kommunikation einer HTTP-Session wird immer vom Client initiiert, in unserem Fall dem Browser. Dieser Browser sendet eine Anfrage an den Webserver, der die Anfrage interpretiert und eine Antwort zurücksendet. Eine solche Anfrage ist ein simpler ASCII-String. Sie können diese Kom-

munikation jederzeit mit *TELNET* oder jedem anderen einfachen Terminal-Emulator testen. TELNET kann von der DOS-Kommando-Zeile gestartet werden:

```
telnet localhost 80
```

Potnummern können beliebig festgelegt werden

Dieser Aufruf geht davon aus, dass Ihr Webserver auf dem lokalen PC unter dem Port 80 installiert ist. Port 80 ist der Default für die meisten HTTP-Server, insbesondere für den mit Windows 2000 installierten Internet Information Server. In einer Produktivumgebung ändert sich die Port-Nummer meistens, üblicherweise nimmt man die Nummer 8080, aber auch das ist willkürlich.

Sobald Sie erfolgreich eine Verbindung zum Server aufgebaut haben, wird der Screen von TELNET gelöscht und der HTTP-Server wartet auf weitere Eingaben. Sie können nun eine gültige HTTP-Anfrage eingeben, entweder mit oder ohne absoluter Pfadangabe:

```
GET default.htm HTTP/1.0
{     eine Zeile freilassen}
GET /asp/helloworld.htm HTTP/1.0
{     eine Zeile freilassen}
```

HTTP-Requests werden durch Leerzeile angezeigt

Das Ende eines Requests wird durch eine Leerzeile angezeigt. Der Server wartet so lange, bis er eine Leerzeile im Datenstrom, also zwei aufeinander folgende <CR><LF>-Sequenzen (Carriage Return/Linefeed), erhält, und beginnt dann, die Anfrage zu bearbeiten. Kommt die Leerzeile nicht, wartet der Server unter Umständen ewig.

Antwort beginnt mit HTTP-Statuscode

Wenn die Antwort zurückkommt, zeigt TELNET den erhaltenen Datenstrom an. In dem gezeigten Beispiel handelt es sich dabei um die Verwaltungsdaten der HTTP-Antwort und den Inhalt der gewünschten Datei. Dabei ist die erste Zeile der Antwort *immer* der Status-String von HTTP, der uns mitteilt, auf welche Weise der Server die Anfrage beantwortet hat, insbesondere ob die Anfrage erfolgreich war oder nicht. Die folgende Zeile meldet eine erfolgreiche Verarbeitung mit Status 200:

```
HTTP/1.0 200 OK
```

So meldet HTTP einen Fehler mit Status von 400 und größer

```
HTTP/1.0 404 NOT FOUND
```

10.3.2 Beispiel für einen HTTP POST-Request

HTTP POST sendet Daten im Body

Ein `HTTP POST` erlaubt es, im Rumpf der Anfrage beliebige Daten mitzusenden. Die häufigste Verwendung ist es, die Eingaben in einem HTML-Formular an den Server zu übertragen. Praktisch sieht dies so aus, dass der CGI-String, der bei einem `HTTP GET` einfach an die URL angehängt wird, im Body der Anfrage übertragen wird.

Listing 10.3 Beispiel für ein HTML-Formular mit Feldern »ui«, »pw« und dem Pushbutton »OK«

```
<html>
  <head>
    <title>Login</title>
    <link rel="stylesheet" href="style.css"
    type="text/css">
  </head>

  <body>
    <IMG border=0 src="logo.gif" width="215"
    height="35">
    <hr>
    <h3>Login</h3>
    <form method="POST" action="login.asp"
    class="shopform">
    <font size="4" color=red>
    </font>
    <p>Please enter your User Data below to proceed:
       (Userid=logos Password=world)</p>
      <table border="0" cellpadding="0" cellspacing="0"
      width="300">
        <tr>
          <td width="175">User Name</td>
          <td width="125"><input name="ui" size="20">
          </td>
        </tr>
        <tr>
          <td width="175">Password</td>
          <td width="125"><input name="pw" size="20"
          type="password"></td>
        </tr>
      </table>
```

```
      <hr>
      <p><input type="submit" value="OK" name="OK"
      tabindex="1" class="button">
      <input type="reset" value="Cancel" name="Cancel"
      tabindex="2" class="button"></p>
    </form>
    <hr>
  </body>
</html>
```

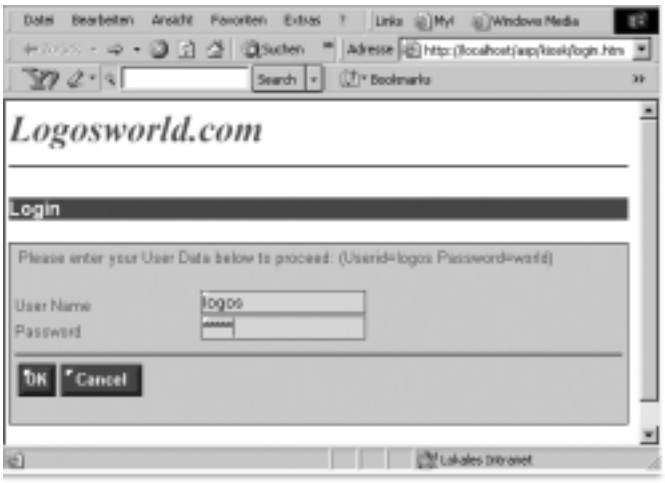

Abbildung 10.3 HTML-Formular in der Browserdarstellung

Wenn das HTML-Formular abgesendet wird, erzeugt der Browser daraus einen HTTP-Post-Request, in dessen Bauch die Formulardaten übertragen werden. Das nachstehende Listing zeigt ein Beispiel für eine HTTP-Anfrage mit typischen Header-Informationen und der Übergabe der Formularvariablen `ui`, `pw` und `OK`.

Listing 10.4 Beispiel für einen HTTP POST-Request

```
POST http://localhost/asp/kiosk/login.asp HTTP/1.1
Accept: */*
Referer: http://localhost/asp/kiosk/login.htm
Accept-Language: de
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.0; .NET CLR 1.0.3705)
```

```
Host: localhost
Content-Length: 23
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: ASPSESSIONIDQGGGQJEC=HGGLFPKDPIAMLFEGLDCFJGKE
Extension: Security/Remote-Passphrase

ui=logos&pw=world&OK=OK
```

10.3.3 Anfragen direkt ins Internet

Zugriff über Proxy-Server

Wenn Sie noch keinen Webserver installiert haben, können Sie versuchen, sich über TELNET mit dem Internet zu verbinden. Falls Ihr PC über einen Proxy-Server auf das Internet zugreift, dann müssen Sie sich anstatt mit dem Webserver mit dem Proxyserver verbinden, der die Kommunikation mit dem Webserver selbst herstellt und die Anfragen und Antworten zwischen den Parteien vermittelt. Folgendes wäre die Anfrage über das Internet:

```
telnet www.yahoo.com 80
GET index.htm HTTP/1.0
{eine Zeile frei lassen}
```

Und so sähe die Anfrage über den Proxy-Server aus:

```
telnet myproxy 80
GET http://www.yahoo.com/index.htm HTTP/1.0
{eine Zeile frei lassen}
```