HttpListenerContext decode/encode Umlaute

Liest man die Url aus dem HttpListenerContext, die Umlaute wie äöü enthält, so sieht das ungefähr so aus:

aus süß wird s%FC%df.

Abhilfe schaft da die Klasse HttpUtility:
[crayon-673f343d6214a372704828/]
möchte man zurück encodieren macht man einfach :
[crayon-673f343d62152673492657/]

TCP / IP – Teil 3: HTTP – Protokoll

Das HTTP (Hypertext Transfer Protocol) baut auf das TCP auf, bildet in seiner Übertragung jedoch weitreichende eigene Konzepte, als das die Übertragung über TCP/IP.

Ebenfalls ist das HTTP ist heute das Standardprotokoll um Webseiten im Webbrowser darzustellen, welches sich laut W3C im 1.1 Standard befindet. HTTP/2 befindet sich aber bereits in Entwicklung.

Eine Ausführung sieht im kurzem immer so aus:

1. Server wartet über einen Port i.d.R. Port 80 auf eine Anfrage

2. Client sendet über POST oder GET eine Anfrage (Request) an

den Server

3. Falls dieser Erreichbar ist sendet der Server eine Antwort (Response) zurück

URL

Eine URL kann z.B. so aussehen:
[crayon-673f343d62602521948777/]
http – ist das verwendete Protokoll

www.devandy.de – hostname oder Domain

/meinOrdner/2015-04-01/ – Verzeichnis wo sich die Datei auf dem Server befindet

datei01.php – die Datei an die ein Request gesendet wird

?vorname=peter&nachname=lustig – Querystring, der immer mit einem ? beginnt. vorname= und nachname= bilden dabei die variablen und peter und lustig sind die darin enthaltenen werte.

Übertragungsmethode GET

Die Übertragung findet über die URI, über den Querystring statt. Die Anweisungen aus der method sind in der URL Sichtbar und können für spätere selbe Zwecke gespeichert werden

Übertragungsmethode POST

Die Übertragung über POST hingegen kann man nicht bookmarken, da die Übertragung im Content geschieht. Dabei werden die Informationen im Header weiter gegeben. Ein Dateiupload ist z.B. nur mit Post möglich, ein Textfeld mit einem Roman macht auch nur über Post Sinn, weil dieser nicht begrenzt ist.

1. Schritt: Server wartet auf Anfrage

In der Regel sind die Serverprogramme wie der IIS Server oder

Apache standardisiert eingerichtet, so dass immer über Port 80 gelauscht wird. Man kann dies aber Serverseitig auch auf einen anderen Port umlegen.

2. Schritt: Client Request

Der Client wie der Webbrowser sendet an den Host über das http Protokoll eine Anfrage in Form einer URL.

Dabei enthält diese Anfrage 2 Teile. Zum einen die Anforderungszeile (URL) und zum anderen einen Header, der so aussehen kann:
[crayon-673f343d62607068309065/]
Die erste Zeile beinhaltet mit GET die Methode über den Request, index.html die zu aufrufende Datei und dann die Protokollart und Versionsnummer.

Darunter folgen weitere Informationen wie Ursprungsland, Bilder erlauben, Browsertyp und Versionsnr, Referer (welche Seite man vorher besucht hat), usw.

3. Schritt Server Response

Ist der Server erreichbar, sendet er eine 3 teilige Antwort wieder. Diese besteht aus

1. Statuszeile
[crayon-673f343d62608536892773/]
bestehend aus dem Protokoll, Versionsnummer, Statusnummer und Status.

2. Header

Der Header beinhaltet das Date, Datum und Uhrzeit wann die Antwort geschickt wurde, Server Name und Versionsnummer des Webservers, Content-Length, Länge des Nachrichten-Body in Byte usw.

3. Body

Darin befindet sich der eigentliche HTML Code, der dann im Browser angezeigt wird.

Beispiel Projekt in C#, ein Http Server:

HTTP Server

Quellen:

SelfPHP Get/Post

Andreas Olesch

TCP / IP – Teil 2: Verbindungsaufbau

Verbindungsaufbau

Server:

1. Programm muss dem BS mitteilen, dass es nun Verbindungen über Port X annehmen möchte.

2. Der Firewall Port X öffnen, damit darüber kommuniziert werden kann

Server wartet nun auf eine Verbindung über Port X von außen

Client:

Programm möchte eine Verbindung über Port X zu Host herstellen.

1. Dem Betriebssystem mitteilen, dass es eine Verbindung herstellen möchte

2. Das Betriebssystem weist dem Client nun auch einen freien Port ab 1024 zu

3. Versuchen eine Verbindung zu Host herzustellen

Server:

1. erkennt, dass zu ihm eine Verbindung hergestellt werden will und nimmt diese an. Bekommt auch die Mitteilung auf welchem PC und an welcher Portnummer der Client läuft.

2. Damit der Server in der Lage ist mehrere Verbindungen von mehreren Clients anzunehmen und zu verwalten, erhält jede Verbindung eine Verbindungsnummer auch Handle genannt.

3. für jede Verbindung wird ein Prozess zur Verfügung gestellt, welcher nur diese Verbindung verarbeitet.

Der Server wartet also nur auf eine Anfrage vom Client und gibt dann die Antwort zurück.

Nach Abarbeitung kann sowohl der Server als auch der Client die Verbindung beenden.

Wenn eine Zeitlang keine Reaktion vom Client kommt, kann der Server sagen „timeout!“ und die Verbindung schließen.

TCP / IP – Teil 1: Begriffserklärung

kurze Begriffserklärung:

Server

ist kein Rechner an sich, sondern es ist nur ein Programm, oft ein Dienst welches auf einem Rechner läuft und seine Dienste zur Verfügung stellt.

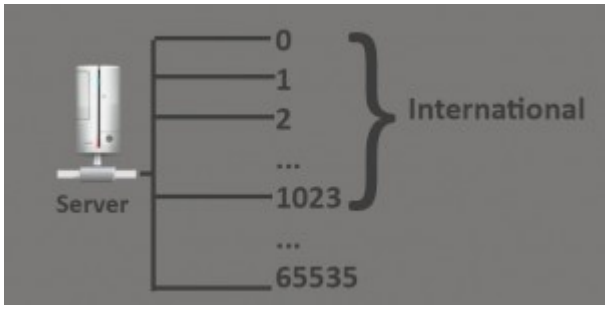
Client

ist genauso wie der Server nur ein Programm, welches jedoch die Dienste empfängt.

Host

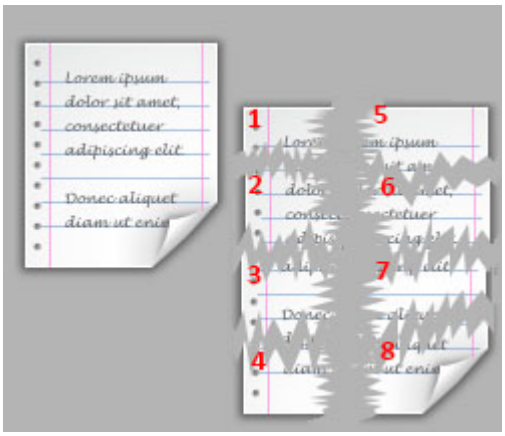
Der Rechner an sich wird dann als ein Host bezeichnet, wenn Datenübertragungen stattfinden.

Port



Vergleichbar mit einer Strasse mit 65536 Häusern. Jedes dieser Ports kann eine Aufgabe übernehmen. Berühmte standardisierte Ports sind z.B. Port 80 für das WWW / HTTP, Port 21 für FTP. Dabei sind die Ports von 0 – 1023 standardisiert für Internationale Zwecke bestimmt wie z.B. die eben genannten.

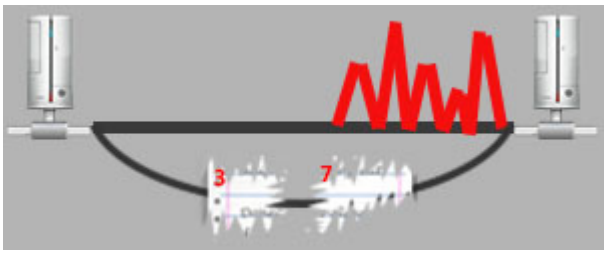
TCP – Transmission Control Protokoll



Die Aufgabe von TCP ist es, das Datenpaket, welches zugrunde liegt zu nehmen und in viele einzelne Teile zu splitten. Jedes dieser Teile wird nummeriert.

Nach der Übertragung auf dem Zielsocket, sind die einzelnen Datenpakete durcheinander. Es ist nun ebenfalls die Aufgabe von TCP diese nummerierten Pakete in richtiger Reihenfolge aufzustellen. Nachdem die einzelnen Elemente zusammengefügt wurden, erhält das Zielsocket eben dieselbe Datei wie die Quelle diese geschickt hat.

IP – Internet Protokoll



Die Aufgabe des Internet Protokolls ist es nun die kürzeste Verbindung zu finden. Ist diese aber Fehlerhaft, so sucht die IP nach einer Alternativen Route und übermittelt darüber. Dies sind die 2 Aufgaben der IP.

Sockets

Auf Deutsch auch Sockel sind die beiden Endpunkte vom Server

und Client. Diese kann man sich wie eine Steckdose vorstellen. Dabei kann entweder darüber das TCP oder das UDP Protokoll genutzt werden.