PÁZMÁNY

Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Számítógépes hálózatok
#08 – FTP, SMTP, HTTP

2024. november 8.

Naszlady Márton Bese

naszlady@itk.ppke.hu

Alkalmazások

Alkalmazások, amik igazából programok, illetve IoT eszközök

Google Chrome, Internet Explorer, PUBG, Fortnite, Spotify, MS Teams, Windows Update, Zoom, Outlook, Tinder, ...

Amazon Echo, Apple Homepod, WiFi-s hűtőszekrény, WiFi-s lámpa, IoT kutyatál, IP kamerák, NAS, ...

Alkalmazások, amik igazából protokollok, eljárások, szabványok

DHCP, DNS, HTTP, FTP, IMAP, LDAP, SMTP, SSH, TLS/SSL, NTP, ...

Alkalmazások

Alkalmazások, amik igazából programok, illetve IoT eszközök

Google Chrome, Internet Explorer, PUBG, Fortnite, CS:GO, , MS Teams, Windows Update, Zoom, Outlook, Tinder

Amazon Echo, App , Wi-Fi-s lámpa, IoT kutyatál, IP kamerák, NAS, ..

Ezekkel más tárgyakon fogtok foglalkozni (pl. Java, Web programozás, IoT, stb...)

Alkalmazások, amik igazából protokollok, eljárások, szabványok

DHCP, DNS, HTTP, FTP, IMAP, LDAP, SMTP, SSH, TLS/SSL, NTP, ...

Alkalmazások

Fájlátvitel

- módszer arra, hogy vékony klienst bootoljunk (letöltsük az oprendszert)
- módszer arra, hogy állományokat mozgassunk gépek közt

Levelezés

- az email mint elektronikus üzenet mibenléte
- email küldése és fogadása
- az email adattartalma, MIME

Web

- erőforrások szövevényes rendszere
- hiperszöveg és hiperhivatkozás

#08/1 – FTP

File Transfer Protocol

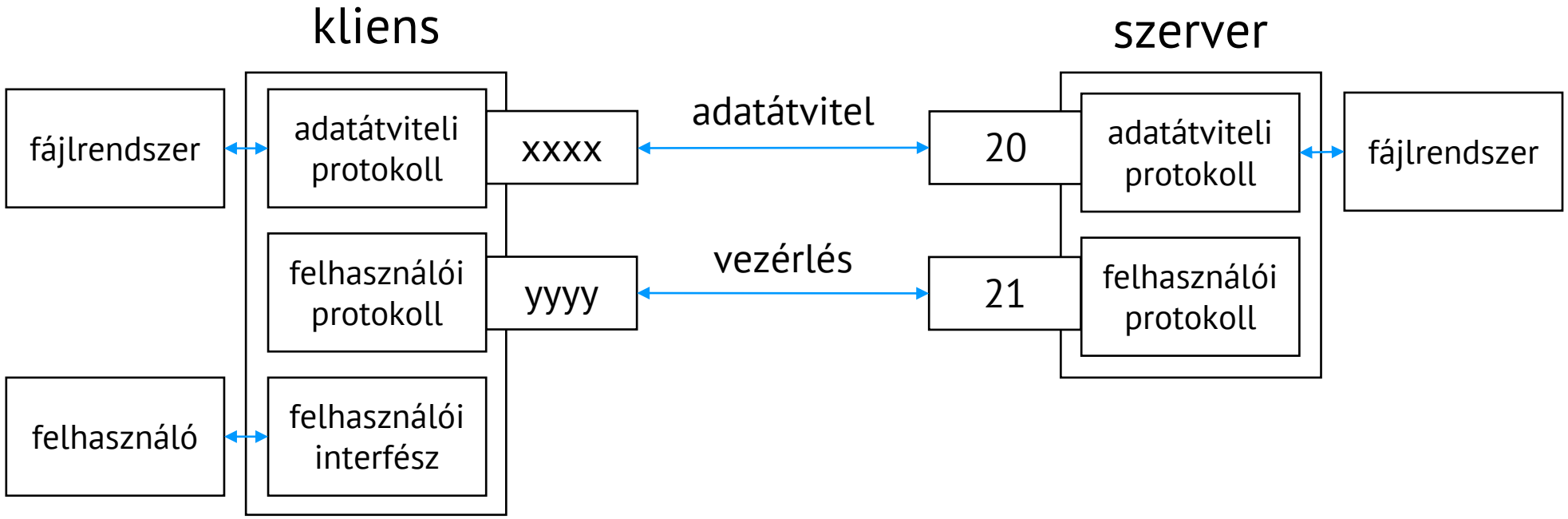
Motiváció: A TFTP nem kényelmes, mert

- nem stabil az átvitel (pl. bűvészinas),
- nincsenek extra szolgáltatások (azonosítás, könyvtárak böngészése, …),
- nincsenek fájlműveletek (törlés, feltöltés, jogosultságok módosítása, …).

Megoldás: Új protokollt írunk; két párhuzamos TCP kapcsolatot építünk:

- vezérlésért felelős kapcsolat (itt mennek a parancsok, visszajelzések)
- adatátvitelért felelős kapcsolat (itt megy maga az adat, pl. a fájl bitjei)

FTP kapcsolatok



A **control** (vezérlő) kapcsolatot a kliens építi fel egy random portról a szerver 21-es portjára.

A **data** (adat) kapcsolatot vagy a szerver (active FTP) vagy a kliens (passive FTP) építi ki a kliens random és a szerver 20-as portja közt.

FTP jellemzők

Az FTP során különböző architektúrák, fájlrendszerek, fájlformátumok, karakter-kódolások kezelésére van szükség. (pl. Linux Ext4 lemezről Windows NFTS-re).

Tulajdonképpen bármilyen fájl átvihető a megfelelő formátum megválasztásával (ASCII, EBCDIC, image, local, ...).

Az átvitel módja lehet:

- stream (byte folyam) – ez az alapértelmezés
- block mode – az átvitel blokkokban zajlik; blokk = fejrész + adat
- compressed mode – az egymás után következő egyforma byteokat tömöríti
  (nem hasznos, jobb ötlet tömöríteni a fájlt és utána átvinni)

FTP control kapcsolat

A kliens vezérli a szervert.

A vezérlés egysoros parancsokkal történik, a sorok végén sorvége karakter (\r\n)

A parancs egy betűkóddal kezdődik, majd esetleg paraméterek követik. A betűkód egészen értelmes rövidítése annak, amit a parancs csinál.

A szerver a parancsra egy három számjegyből és szövegből álló válasszal felel. A számok szólnak a gépnek (válaszkód), a szöveg az embernek (érthető leírás).

A kommunikáció tehát „emberileg is fogyasztható” formában történik.

FTP control kapcsolat

FTP command példák – ilyeneket küld a kliens

`USER felhasználónév`	belépési kísérlet a megadott felhasználóval
`PASS jelszó`	az előbb megadott felhasználó jelszava
`TYPE típus`	fájltípust határoz meg az átvitelekre
`CWD mappa`	mappát változtat
`RETR file`	letölt egy fájlt a kliensre
`STOR file`	felad egy fájlt a kliensről
`ABOR`	a folyamatban levő átvitelt elveti
`PORT n1,n2,n3,n4,n5,n6`	adatkapcsolat kezdeményezése (aktív)
`PASV`	adatkapcsolat kezdeményezése (passzív)
`QUIT`	bontja a kapcsolatot

FTP control kapcsolat

FTP response codes – ilyeneket küld a szerver

A válasz első számjegye dönti el, hogy mennyire örüljünk:

- `1xx` – részleges, előzetes siker
- `2xx` – teljes siker
- `3xx` – valamilyen köztes állapot
- `4xx` – átmeneti kudarc
- `5xx` – teljes kudarc

Például:

```
150 File status okay; about to open data connection.
200 Command okay.
331 User name okay, need password.
450 Requested file action not taken.
500 Syntax error, command unrecognized.
```

FTP data kapcsolat

Az adatátvitelhez a vezérléstől független TCP kapcsolat épül fel, rendszerint minden fájl átviteléhez külön-külön.

A mappalistázás és hasonló parancsok eredményeit is így kapjuk meg; a mappalistázás során kvázi egy fájlt töltünk le, amiben benne van a mappaszerkezet.

Az adatkapcsolat felépítése történhet aktív és passzív módon is:

- **aktív** esetben a szerver indítja a TCP handshake-et
- **passzív** esetben a kliens indítja a TCP handshake-et

Aktív FTP kapcsolat

A kliens a **PORT** parancssal kezdeményezi az adatkapcsolat felépítését:

```
PORT 193,225,150,80,14,178
```

A parancs után 6 byte:

- első 4 byte a kliens IP címe: **193.225.150.80**
- utolsó 2 byte együttesen a kívánt portszám: 14 × 256 + 178 = **3762**

A szerver kezdi a TCP handshake-et, a kiinduló port a 20-as port lesz.

Passzív FTP kapcsolat

A kliens a **PASV** parancssal kezdeményezi az adatkapcsolat felépítését:

```
PASV
```

A parancsban nem küldünk semmit, a szerver fog válaszolni, hogy hol akarja fogadni az FTP adatkapcsolatot:

```
227 Entering Passive Mode (195,228,252,133,236,56)
```

Az üzenet nagyon hasonlít a PORT parancsra.

Ezután a kliens indítja a handshake-et a szerver megadott portja felé.

Biztonsági kérdések

Az FTP-vel lehetőségünk van felhasználónevet és jelszót kérni a hozzáférés előtt. Ezzel valamennyire biztosítjuk azt, hogy illetéktelenek ne férjenek hozzá a fájlokhoz.

Létezik anonymous FTP is, ekkor az USER parancsban az „anonymous” felhasználónevet küldi a kliens, a jelszó pedig „bármi” lehet (a szerver általában egy email címet kér).

Az FTP esetében a felhasználónév és jelszó kódolatlanul utazik az egész világon át!

#08/1 – Összefoglalás

Fogalmak

FTP motiváció

Megvalósítás (két TCP kapcsolat)

Vezérlési parancsok felépítése, válaszok jellemzője

Aktív és passzív FTP

#08/2 – SMTP

Simple Mail Transfer Protocol

Motiváció:

El szeretnénk juttatni egy email üzenetet a feladó gépéről a címzett gépére. Az email üzenetet tartalmazó fájlt (adatot) szeretnénk átvinni a levelezést lebonyolító szerverek között.

Azt is szeretnénk, ha a címzett gépén a címzett fiókjába kerülne a levél.

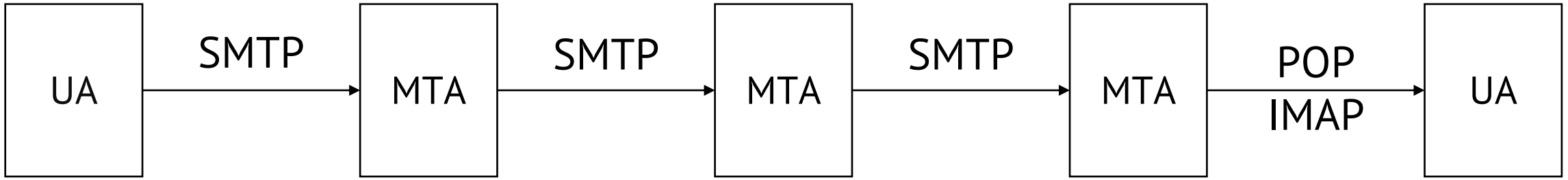
Megoldás:

Készítünk egy protokollt, ami nem (csak) gépnevek, hanem a gépnevek és fiókok alapján közlekedteti az üzeneteket.

Levelezés összetevők

Az üzenetet a baloldali UA (User Agent – felhasználói levelezőkliens) küldi. Az üzenetet az MTA (Mail Transfer Agent – üzenettovábbítók) továbbítják az utolsó MTA-ig. Ez a továbbítás az SMTP protokoll használatával zajlik.

Az utolsó MTA-tól már általában POP vagy IMAP protokoll használatával töltődik le a levél a jobboldali UA számára.



SMTP vezérlés és adatátvitel

Az SMTP az FTP vezérlőüzeneteknél is látott formájú egysoros, sorvége jellel (\r\n) záródó parancsokkal működik.

A parancsokra az FTP-hez hasonlóan itt is háromjegyű válaszkódokkal és az utánuk fűzött szöveggel válaszol a szerver.

A vezérlés és az adat közös csatornán közlekedik.

Az SMTP TCP protokollt használ a kliens tetszőleges és a szerver 25-ös portja közt.

SMTP parancsok

`HELO valami.domain.nev`

A kliens köszönti a gépet. Az itt szereplő domain a kliens domainneve. Ha ez hazugság (inverz DNS lekérdezés szerint), akkor el lehet utasítani a kapcsolatot. Ha a szerver rendben találja, akkor várja a további parancsokat.

`MAIL FROM: <mailbox@valahol>`

A küldendő levél feladója. A paraméter a feladó címe <> jelek közé zárva. A paraméter lehet üres is, pl. visszapattanó üzenetnél nincs feladó.

Ha van feladó, akkor a valahol résznek érvényes fully-qualified domain névnek kell lennie. A mailbox rész nem feltétlenül egy létező személy, lehet no-reply is vagy valami huncut dolog (pl. levlista címe, root, webmaster stb).

SMTP parancsok

`RCPT TO: <mailbox@valahol>`

A küldendő levél címzettje. A paraméter a címzett címe <> jelek közé zárva.

Egy levélnek lehet több címzettje is, akkor mindegyikhez egy új RCPT TO parancsot kell kiadni, ezzel lehet felsorolni mindenkit, akinek meg kell kapni az üzenetet.

Az erre a parancsra kapott „siker” értékű válasz csak annyit jelent, hogy felírta a borítékra a címzett címét, arról semmi infónk sincs, hogy a címzett amúgy létezik-e, vagy a szerver tud-e neki levelet küldeni. Ez majd később derül ki.

SMTP parancsok

DATA

Ezzel a paranccsal jelzi a kliens, hogy a következő sorokban maga a levél adattartalma fog következni (ez az, ami a borítékban van).

A levél adattartalmát 7 bites ASCII karakterekként kell max. 998 + \r\n hosszú sorokba tördelve megadni.

Az adat (levél) végét az jelzi, hogy küldünk egy olyan sort, ami kizárólag csak egy pontot tartalmaz.

Ha a DATA mező bevitele után 2xx kódot kapunk, akkor a levél bekerült a rendszerbe, innentől a szerver feladata, hogy továbbítsa azt.

SMTP parancsok

`HELP`

Emberek számára fogyasztható beszélgetés a szerverrel. Néha jól jön :D

`RSET`

Kidobja a borítékot; az eddig megadott összes feladó, címzett és tartalom törlésre kerül.

`QUIT`

A beszélgetés végét jelzi, lezárja az SMTP szerverrel való kapcsolatot.

Biztonsági kérdések

Mi alapján szűrje az SMTP szerver a beérkező parancsokat?

Régen nem volt szokás szűrni a klienseket => SPAM

Manapság többnyire két engedélyezett eset van:

- olyan IP címekről fogadunk parancsokat, ahol ez az SMTP szerver a kiindulópont (vagyis ez a legközelebbi postahivatal).
  Ezek az IP-k általában közös subneten vannak a szerverrel.
- bárhonnan fogadunk parancsokat, ha ez az SMTP szerver a végpont (vagyis ez a lakásnál lévő postaláda).
  Ezek azok a címzettek, akiknek itt van postafiókjuk (vagy akik számára mi vagyunk az MX szerver).

#08/2 – Összefoglalás

Fogalmak

Levelezés motiváció

Email üzenet továbbítása

SMTP control és response üzenetek

#08/3 – MIME

E-mail formátum

Motiváció:
Szeretnénk, ha a levélnek lennének „metaadatai” és valós tartalma, akár egyszerre többféle formátumban is. Pl. szeretnénk, ha a levélnek lenne CC, BCC mezője, lenne tárgya, lenne olyan adat, ahol az van, hogy hova kell válaszolni, esetleg legyen prioritás is; plusz szeretnénk szöveget és csatolmányt küldeni.

Hogy kell mindezt beleönteni az SMTP törzsbe úgy, hogy azt a fogadó oldalon értelmesen tudjuk kivenni?

Megoldás:
Az RFC822 (majd RFC2822 (majd RFC5322)) a levélformátum szintaxisát írja le. A levél egy fejlécből és egy törzsből fog állni, ezeket még további részekre bontjuk.

E-mail formátum

A levél fejrésze tartalmazza azokat a metainformációkat, amik segítik a levél értelmezését. Ilyen pl. a cc vagy bcc mező, feladás dátuma, tárgy, stb.

A levél törzse tartalmazza a tényleges szöveget / fájlt vagy a kettőt (sőt, többet is) együtt vegyesen.

A fejlécet a törzstől egy üres sor választja el; a fejléc az első üres sorig tart.

E-mail formátum

A fejlécben mezők vannak. A mezők szerkezete a következő:

- a sor elején szerepel a mező neve
- kettőspont
- szóköz (ill. whitespace)
- a mező értéke

Példa:

```
From: sales@ceg.hu
Message-ID: <Pine.LNX.4.58.0412082029001.10200@login03.caesar.elte.hu>
Date: Mon, 23 Nov 2020 15:53:12 +0100 (CET)
```

E-mail formátum

Gyakori és „hasznos-tudni” fejlécek:

`Date:` A feladás idejét jelzi. Fontos, hogy része az időzóna is. Példa:
`Date: Mon, 23 Nov 2020 15:53:12 +0100 (CET)`

`To:` Az üzenet címzettje vagy címzettjei. Ha több, akkor vesszővel elválasztva. Megadható `"Teljes Név" <email@cim>` formában is. Például:
`To: "Kiss Pista" <kisspista@ceg.hu>`

`Cc:` Carbon copy, azaz indigós másolat. Feltüntethető, hogy más is megkapta az üzenetet.
Fromátuma megegyezik a To fejléc formátumával.

`Bcc:` Blind carbon copy, azaz titkos indigós másolat. Annak, aki így kapta a levelet, benne lesz a fejlécében, de a többieknek nyilván nem kerül bele (mert akkor lebuknánk).

`From:` Azt a küldőt jelenti, akinek a nevében megy a levél.
Pl. `From: sales@ceg.hu`

`Sender:` Azt a küldőt jelenti, aki ténylegesen el is küldi a levelet.
Pl. `Sender: kisspista@ceg.hu`

E-mail formátum

Gyakori és „hasznos-tudni” fejlécek:

`Reply-to:` Kérem, hogy erre a címre válaszoljon.

`Message-ID:` Az üzenet egyedi azonosítója. Ez alapján akár vissza is lehet keresni a levelet a naplófájlokban. Általában `valami.hash@valami.gep` alakú, ahol a @ utáni rész azt a gépet azonosítja, ahol a levél keletkezett.

`Subject:` A levél tárgya. Elvileg lehet üres/hiányzó is, de az egy kicsit neveletnen :(

`Received:` A közvetítőt MTA-k által betett mező, ezzel nyomon lehet követni, hogy merre járt a leveleünk, hogyan kézbesült az számunka. Ebből a mezőből instant le lehet bukni, hogy sose járt azon az SMTP szerveren a levél, ahonnan elvileg küldték.

Ezzel a mezővel a körbe-körbe keringő levelek is szűrhetők.

A fentieken túl további (szinte bármilyen) fejlécek megadhatók a levélben, nagyon gyakran csinálják is a levelezőprogramok, MTA-k, hogy mindenféle jóságot pakolnak ide. Például spam szűrést segítő cuccok, levlista információk, debug üzenetek, random szemét stb.

MIME

Motiváció:
Nem csak 7-bites ASCII sorokat akarunk küldeni, hanem képet, videót, vírust stb.

Megoldás:
Az üzenet nem csak fejléc-törzs részekből állhat, hanem fába szervezett módon az egyes törzs-részek is állhatnak további fejléc-törzs részekből.

Az egyes részek más és más kódolással csomagolhatók ki, így az egyik részből sima szöveg, a másikból formázott szöveg, a harmadikból kép stb lehet.

MIME

Multipurpose Internet Mail Extensions (MIME)

A levél fejrészét két további mezővel fogjuk bővíteni, ezek a MIME-version és Content-Type mezők.

`MIME-vesrison:` megadja, hogy melyik MIME implementációval készült a levél

`Content-Type:` megadja, hogy a tartalom milyen kódolással van, ill. miként kell értelmezni azt, milyen fájl/adat lesz belőle.

Például:

`text/plain; charset=UTF-8`	UTF8 kódolású szöveg
`application/pdf`	PDF fájl
`image/png`	PNG kép

MIME

A Content-Type értéke lehet `multipart/*` is, ami azt jelenti, hogy az üzenet több további részből áll, amiket külön-külön még ki kell csomagolni.

A részeket a fejlécben megadott „elválasztó” jelöli. Például:

```
Content-type: multipart/mixed boundary "ezittaz"
```

Ekkor az adatban a részeket ilyen sorok választják el:

```
--ezittaz
```

Az utolsó darabot pedig ez a sor zárja le:

```
--ezittaz--
```

Content Transfer Encoding

A `Content-transfer-encoding` mezőben tudjuk megadni, hogy a levél törzsében milyen formában van benne az adat. A lehetséges értékek:

`7bit` – közönséges 7-bites ASCII karakterek sorokra tördelve

`8bit` – nem csak 7-bites karakterek, de még mindig sorokra tördelve

`quoted-printable` – a 7-bites ASCII karaktereket nem bántjuk, de amik nem azok, azokat 3 karakteres szekvencia kódolja. Például a 0xC3 értékű byte-ot =C3 fogja jelölni. Sajnos így az egyenlőségjelet is kódolni kell: =3D

`base64` – az üzenetet a base64 kódolással kódoljuk (így bármilyen binárisból ASCII karaktereket tudunk kapni, pl. egy kép vagy exe is átkódolható).

#08/3 – Összefoglalás

Fogalmak

Email motiváció

Néhány mező jellemzője

MIME típus

Üzenetek részeinek elválasztása

Content Transfer Encoding

#08/4 – Web alapok

Web alapok

Szeretnénk, hogy egy erőforrás könnyen azonosítható és elérhető legyen.

Ehhez elnevezzük az erőforrást valahogy.

Web alapok

Szeretnénk, hogy erőforrások
más erőforrásokra mutassanak.

Ehhez hiperszöveget hozunk létre.

Ezen igényekre szerver-kliens elvű hálózat.

Tim Berners-Lee, CERN
1990-es évek

World Wide Web

Web alapok

Hiperszöveg

- szöveges adat
- formázási utasításrendszer
- referencia más erőforrásra

Médiaelem

- lehet bármilyen (akár hiper-) szöveg
- kép, hang, videó,
- bármilyen komplexebb állomány (pl. zip, pdf, exe)

Web alapok

Hiperhivatkozás = a web lényege

Hiperszövegek tartalmazzák.
Egy névvel leírt (azonosítható) erőforrásra mutat.

Web alapok

Böngészőprogram = a web másik lényege

Hiperszövegek megjelenítésére (is) alkalmas.
Ismeri a szükséges application layer szintű protokollokat.

HyperText Markup Language (HTML)

Hiperszöveg leírására használt jelölőnyelv (és egyben formátum).

Három fontos feladata:

- felsorolja az oldalt felépítő komponenseket (címsor, bekezdés, táblázat, …)
- megadja az elemek hierarchikus rendszerét (fa struktúra)
- további erőforrásokra hivatkozik

HyperText Markup Language (HTML)

A HTML kódot a böngésző értelmezi. Felépíti a szerkezetet, és letölti a HTML kódban hivatkozott további médiaelemeket (pl. képeket).

```
<body>
  <h1>Ez egy fontos címsor</h1>
  <p>Ez a dolog egy bekezdésben megjelenő szöveg. Magát a
szöveget a HTML kódban adom meg. Ezt fogja majd a böngésző a
formázási szabályoknak megfelelően tördelni és
megjeleníteni.</p>
  <p>Ha azt szeretném, hogy legyen az oldalon egy kép is,
akkor ezt úgy írom le, hogy berakok egy „kép” típusú dobozt,
és megmondom, hogy ebben mit kell megejeníteni, hol található
az a kép a World Wide Weben.</p>
  <img src="https://ppke.hu/uploads/image/EUA_logo.png">
  <ul>
     <li>Ez pedig egy lista</li>
     <li>Ez a lista második eleme</li>
     <li>Tovább is van, mondjam még?</li>
  </ul>
</body>
```

Ez egy fontos címsor

Ez a dolog egy bekezdésben megjelenő szöveg. Magát a szöveget a HTML kódban adom meg. Ezt fogja majd a böngésző a formázási szabályoknak megfelelően tördelni és megjeleníteni.

Ha azt szeretném, hogy legyen az oldalon egy kép is, akkor ezt úgy írom le, hogy berakok egy „kép” típusú dobozt, és megmondom, hogy ebben mit kell megejeníteni, hol található az a kép a World Wide Weben.

- Ez pedig egy lista
- Ez a lista második eleme
- Tovább is van, mondjam még?

Hiperszöveg és multimédiás tartalom

A HTML kódban felsorolt hivatkozásoknak sokféle típusa lehet:

- médiatartalom beszúrás az oldal egy pontjára

```
<img src="https://ppke.hu/uploads/image/EUA_logo.png">
```

- kattintható hivatkozás (hiperlink) egy másik oldalra (erőforrásra)

```
<a href="http://example.com/">
```

- kattintható hivatkozás, ami megnyitja a levelezőprogramot, és feldobja a levélírási ablakot

```
<a href="mailto:kispista@ceg.hu?subject=Megrendelés">
```

- hivatkozás, ami az adott fejezethez görget az oldalon

```
<a href="https://wiki.itk.ppke.hu/twiki/bin/view/TO/Vesélyhelyzet#Oktatás">
```

#08/4 – Összefoglalás

Fogalmak

WWW motiváció

Böngészőprogram

Hiperszöveg

#08/5 – Erőforrások azonosítása

URI, URN, URL

Motiváció:

Szeretnénk megoldani hogy valahogy állományokat, erőforrásokat érjünk el a world wide weben.

Megoldás:

Készítsünk egy olyan egyezményes erőforrás-azonosítót, amivel meg tudjuk adni, hogy pontosan mit szeretnénk elérni (és esetleg azt is, hogy kik vagyunk, és hol keressük azt a dolgot).

Unified Resource Identifier

Az URI egy egységes erőforrás-azonosító, amelyet egy webes erőforrás azonosítására használunk.

Az URI felépítése a következő:

```
URI = scheme:[//authority]path[?query][#fragment]
```

shceme	megadja az erőforrás sémáját (pl. http, mailto, file, ftp, chrome stb.)
authority	opcionális, megadhat felhasználói adatot, hosztnevet, portszámot
path	az erőforrás elérési útja, szegmensenként '/' jellel elválasztva
query	opcionális, adat küldhető vele az erőforrás számára (lekérdezés)
fragment	az erőforrás egy részére hivatkozik (pl. egy címre egy HTML oldalon)

Unified Resource Identifier

Az URI-ben vannak különleges szerepű karakterek. Ezek az alábbiak:

- generic delimiters (általános határolók)

  `: / ? # [ ] @`

- subset delimiters (egyes részeken belüli határolóelemek)

  `! $ & ' ( ) * + , ; =`

Szabadon használható (nincs korlátozva):

`a-z  A-Z  0-9  - . _ ~`

Minden más karatert át kell kódolni „szabályos” formába

URL encode

Az URI-ben nem megengedett karaktereket átkódoljuk:

%XX

Az átkódolt formában a % jel után két hexadecimális digit, a karakter kódja áll.

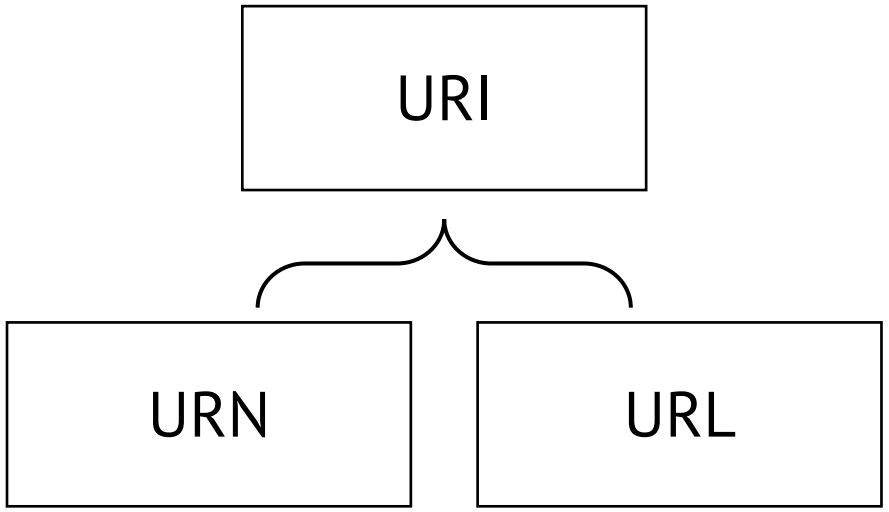
Például:

karakter	UTF-8 felírás	enkódolt
@	0x40	%40
%	0x25	%25
(space)	0x20	%20
ű	0xC5 0xB1	%C5%B1

Nem feltétlenül szükséges, de a nem veszélyes karaktereket is megadhatjuk átkódolva.
Pl. %30%30%37 => 007

URN és URL

Az URI-knek két csoportja van, ezek az URN és az URL.



Az URN a hivatkozott absztrakt vagy konkrét dolog egyértelmű azonosítója.

Az URL a hivatkozott erőforrás helyét adja meg.

URN és URL

URN – Unified Resource Name

Egy absztrakt vagy konkrét erőforrás azonosítására szolgál.

Példák:

`urn:isbn:9781400079988`	könyv
`urn:ietf:rfc:2648`	RFC 2648
`urn:lex:eu:council:directive:2010-03-09;2010-19-UE`	EU jogszabály

Az URN-ből egyértelműen kiderül, hogy mire hivatkozunk, de az nem, hogy azt hol találjuk.

URN és URL

URL – Unified Resource Locator

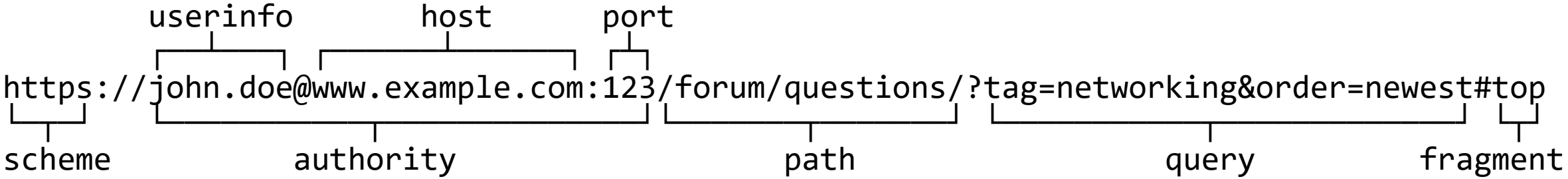
Egy erőforrás konkrét helyét azonosítja.

Példák:

`ftp://user:password@domain:port/path/to/file`	fájl egy FTP szerveren
`https://www.youtube.com/watch?v=dQw4w9WgXcQ`	videós weboldal
`https://moodle.ppke.hu/mod/resource/view.php?id=2662`	ez a diasor PDF-ben

Az URL-ből egyértelműen kiderül, hogy hol találjuk azt a dolgot, amire hivatkozunk.

Az URL részletesen



A scheme, path, query és fragment részeket már láttuk, nézzük az authority részt.

userinfo: a felhasználó azonosítását segítő infók, pl. név, esetleg jelszó

host: a kiszolgáló gép domain neve

port: a kiszolgáló gépen elérni kívánt port száma

Az URL részletesen

Az URL-ben a host névben és a séma névben nem számít a kisbetű-nagybetű.

```
http://example.com/File == HTTP://EXAMPLE.COM/File != HTTP://EXAMPLE.COM/FILE
```

Az URL-ben akár kódolva, akár kódolatlanul is szerepelhetnek a karakterek.

```
http://example.com/File == http://example.com/%46%69%6C%65
```

Az URL-ben nem kötelező megadni a portszámot, ha az nincs megadva, akkor azt tesszük fel, hogy a protokollhoz tartozó well-known port a cél.

```
http://example.com:80/File == http://example.com:/File == http://example.com/File
```

Az URL részletesen

Az URL lehet relatív, azaz a környezettől függő is.

Ebben az esetben az abszolút URL-t a relaítv URL felhasználásával, a relatív URL-t tartalmazó erőforrás URL-jéből képezzük:

Pl. a `http://itk.ppke.hu/` oldalon megadott `http:karunkrol/szabalyzatok` relatív URL az alábbi abszolút helyre mutat: `http://itk.ppke.hu/karunkrol/szabalyzatok`

És pl. a `http://itk.ppke.hu/karunkrol/szabalyzatok` oldalon lévő `http:../oktatas` URL az alábbi abszolút helyre mutat: `http://itk.ppke.hu/oktatas`

#08/5 – Összefoglalás

Fogalmak URI, URN, URL

#08/6 – HTTP

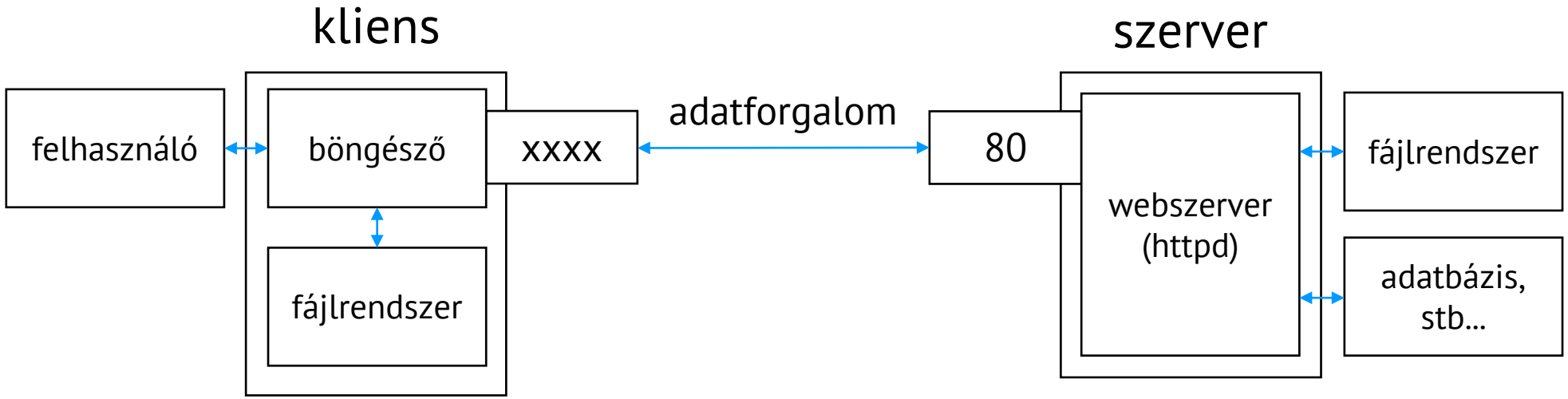
HyperText Transfer Protocol

Motiváció:
Szeretnénk hiperszöveget (multimédia-tartalmat) átvinni gépek között, mégpedig kicsit kényelmesebben, mint ahogy az az FTP esetén volt (két kapcsolat).

Olyat is szeretnénk, hogy néha mi küldünk adatot a szervernek (pl. űrlapkitöltés, bejelentkezési adat, keresés), és ez az átvitel is legyen könnyű

Megoldás:
Implementáljuk a HTTP-t, ami a hiperszövegek és az abban hivatkozott erőforrások oda-vissza átvitelére lesz alkalmas.

HTTP kapcsolat



A HTTP adatátvitel a szerver meghatározott (általában 80-as) portja és a kliens random portja közt kiépített kapcsolaton történik. Ezen a kapcsolaton HTTP kérések és az azokra adott HTTP válaszok közlekednek.

HTTP szabvány

A HTTP jelenlegi legelterjedtebb verziója a HTTP/1.1
(A HTTP/2 verzió 2015-ben jelent meg, jelenleg bevezetés alatt van.)

A HTTP egy TCP feletti protokoll, jellemzően a 80-as (ritkán 8000-es vagy 8080-as) porton működik.

A HTTP kliens-szerver architektúrán alapul, egy kérésre egy választ küld.

A HTTP állapotmentes, azaz nem tart fenn kapcsolatot. Ez alapesetben nem teszi lehetővé, hogy két kérést összekapcsoljunk, de van mód ennek megkerülésére.

HTTP/1.1

A HTTP/1.1 verziójában a kérdések és válaszok szerkezete megegyezik.

A kéréseket olvasható karakterekből álló sorok alkotják. Az üzenet felépítése:

- első sor: megmondja, hogy milyen kérés vagy milyen válasz
- fejrész: RFC822 szintaxis (mint a levelezésnél), minden sorban név-érték párok vannak \r\n sorvége jellel elválasztva
- üres sor: ez választja el a fejrészt a tartalomtól
- adatrész: az adatrész opcionális, nem mindig van küldendő adat (pl. egy kérés esetén nem mindig releváns.)

HTTP/1.1 kérés

A HTTP kérés első sora három részből áll, ezek a metódus, a kérdés-URI és a HTTP verzió. Ezeket a részeket inkább angolul nevezzük meg:

```
Method Request-URI HTTP-version
```

A method rész mindig csupa nagybetűs, a kérdés jellegét adja meg.

A request-URI rész lehet abszolút vagy relatív megadású.

Példa:

```
GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1
 |   |                                          |
method                request-URI              HTTP-version
```

HTTP/1.1 kérés

```
GET /index.html HTTP/1.1
Host: example.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/87.0.4280.66
Accept: text/html,application/xhtml+xml,application/xml
Accept-Encoding: gzip, deflate
Accept-Language: hu-HU,hu;q=0.9,en-US;q=0.8,en;q=0.7
```

HTTP/1.1 válasz

A HTTP válasz első sora is három részből áll, ezek a HTTP verzió, státuszkód és státusz-indoklás. Ezeket a részeket is inkább angolul nevezzük meg:

```
HTTP-version Status-code Reason
```

A Status-code rész az FTP-hez és SMTP-hez nagyon hasonló módon itt is azt fejezi ki, hogy mennyire sikerült végrehajtani az adott kérést.

- 1xx – részleges, előzetes siker
- 2xx – teljes siker
- 3xx – átirányítás
- 4xx – kliens kudarc (rosszat kérdeztünk)
- 5xx – szerver kudarc (belehaltunk a válasz megalkotásába)

HTTP/1.1 válasz

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Age: 224293
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 01 Dec 2019 15:42:44 GMT
Etag: "3147526947+ident+gzip"
Expires: Tue, 08 Dec 2019 15:42:44 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (dcb/7F18)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 648

<!doctype html>
<html>
...
```

HTTP státuszkódok

„Hasznos tudni” státuszkódok (legalább a létezésükre érdemes emlékezni)

`100 Continue`	folytatom a végrehajtást, ez csak egy köztes jelzés
`200 OK`	minden rendben, itt a válasz
`201 Created`	minden rendben, létrehoztam, amit kértél
`301 Moved Premanently`	amit kerestél, az elköltözött ide: …
`400 Bad Request`	nem értem a kérdést
`401 Unauthorized`	tessék bejelentkezni
`402 Payment Required`	tessék fizetni
`403 Forbidden`	nem engedem a hozzáférést a jelenlegi felhasználónak
`404 Not Found`	nincs ilyen erőforrás
`501 Not Implemented`	nincs implementálva a szerveren a kérés feldolgozása

HTTP metódusok

A HTTP kérés többféle metódus alapján történhet attól függően, hogy mi a célja a kérésnek, milyen hatást akarunk kiváltani vele.

A leggyakrabban használt metódusok:

`GET`	a kliens valamilyen erőforrást akar letölteni a szerverről
`HEAD`	mint a GET, de az adat nem érdekel minket, csak a fejléc
`POST`	a kliens adatot küld, és kéri ennek feldolgozását a szervertől
`PATCH`	a kliens egy kisebb módosítást akar végrehajtatni a szerveren lévő adaton
`DELETE`	a kliens kéri a szervert, hogy töröljön egy szerveren tárolt valamit

HTTP GET

A GET kérésben csak egy fejléc kerül küldésre, adat nem.

A GET-re kapott válaszban fejléc és jellemzően adat is van.

A GET célja, hogy letöltsük egy állomány (hiperszöveg, kép stb.) tartalmát.

A GET kérés nem okozhat változást a szerveren; akkor használjuk a GET metódust, ha a művelet idempotens (vagyis $A * A = A$, azaz a két operandus és az eredmény megegyezik => a művelet nem változtat semmin).

GET kérés például: egy oldal betöltése, egy kép letöltése, ...

HTTP HEAD

A HEAD kérésben csak egy fejléc kerül küldésre, adat nem.

A HEAD-re kapott válaszban is csak egy fejléc szerepel, adat nem.

A HEAD célja leginkább az, hogy megtudjuk az erőforrás jellemzőit azelőtt, hogy az adatot ténylegesen letöltöttük volna.

Tehát pl. tudni szeretnénk a fájl méretét a letöltés előtt, ehhez intézhetünk egy HEAD kérést a fájlra, és a visszaadott fejlécben a Content-Length mező megadja a választ a feltett kérdésre.

A GET-hez hasonlóan ez sem idézheti elő a szerveren az adat megváltozását.

HTTP POST

A POST kérésben fejléc és adat is eküldésre kerül.

A POST-ra kapott válaszban általában fejléc és többnyire adat is szerepel.

A POST célja hogy adatot küldjünk a szerver számára. Szeretnénk, ha ez az adat változást idézne elő a szerveren tárolt adatokban.

Tipikus POST eset például az űrlapok elküldése, vagyis pl. a bejelentkezés vagy egy online vásárlásnál a kosár megrendelése.

A POST kérés leggyakrabban létrehoz valamit (pl. munkamenetet a felhasználó számára, megrendelés sort az adatbázisban), és a válaszban az imént létrejött objektumot/erőforrást szokás visszaadni.

HTTP PATCH

A PATCH kérésben fejléc és adat is eküldésre kerül.

A PATCH-re kapott válaszban általában fejléc és többnyire adat is szerepel.

A PATCH célja hogy adatot küldjünk a szerver számára. Szeretnénk, ha ez az adat változást idézne elő a szerveren tárolt adatokban.

Tipikus PATCH eset például az, ha módosítani szeretnénk valamit, például növelni egy kosárban lévő tétel darabszámát.

A PATCH kérés ekkor az adatbázisban tárolt „kosár” objektumban megadja, hogy melyik elemet kell növelni, az erre adott válasz pedig rendszerint a módosított erőforrás lesz, azaz ebben az esetben a módosult kosár-tartalom.

HTTP DELETE

A DELETE kérésben fejléc és adat is eküldésre kerül.

A DELETE-re kapott válaszban általában csak egy fejléc.

A DELETE célja hogy a szerveren meglévő adatot töröljünk.

Tipikus DELETE eset például az, ha ki akarjuk törölni a kosár tartalmát.

A DELETE kérés ekkor az adatbázisban tárolt „kosár” objektumot törlni, és például a 204 No Content státuszkóddal tér vissza, ami jelzi, hogy „rendben, nincs ilyen erőforrás”.

Sütik

Motiváció:
A HTTP állapot nélküli protokoll, vagyis a kérdések nem hozhatók összefüggésbe egymással; nem tudhatjuk, hogy mik azok a kérések, amiket logikailag egybe tartozónak gondolunk.

Megoldás:
Tegyünk egy egyedi azonosítót a HTTP kérések és válaszok fejrészébe, ami minden kérdésben és válaszban ott lesz.

Sütik

A látogató meglátogat egy oldalt. Ehhez HTTP kérést intéz a szerverhez.

A szerver gyárt egy választ, és a válasz fejrészébe beleteszi az egyedi azonosítót. (Süt egy sütit, és az elküldi a kliensnek.)

A kliens a következő HTTP kérésnél visszaküldi a szervertől korábban kapott egyedi azonosítót (a szerver visszakapja a sütit).

Mivel a szerver tudja, hogy melyik korábbi kérésre válaszolt ezzel a sütivel, a két kérést össze tudja kapcsolni; tudhatja, hogy azok ugyanattól a látogatótól jöttek.

Ha a kliens a továbbiakban is elküldi a sütit, akkor minden kérést hozzá tudunk kötni az eredetihez; tudni fogjuk, hogy milyen kérések és válaszok követték egymást a HTTP kommunikáció során.

Sütik

Miért akarja mindenki, hogy fogadjam el a sütiket? Egyáltalán milyen sütik ezek?

Működéshez szükséges sütik
Ezzel tudja azonosítani a szerver a bejelentkezett felhasználót.

Kényelmi sütik
Ezek segítenek abban, hogy megmaradjanak az oldalon tett beállítások, pl. hogy milyen volt az oszlopok rendezése, vagy hogy milyen pénznemben akarok fizetni.

Marketing sütik
Ezeket a sütiket a reklámokat megjelenítő oldalak teszik le, és nyomon tudják követni vele, hogy a felhasználótól milyen kérések érkeznek, mikre keres rá és hol.

#08/6 – Összefoglalás

Fogalmak

HTTP rendeltetése

Státuszkódok

Metódusok

Süti

VÉGE

PÁZMÁNY

Pázmány Péter Katolikus Egyetem

Információs Technológiai és Bionikai Kar