# 25. HyperText Transfer Protocol

V knize najdete lecjaká témata, ale protokol jsme tu ještě neměli. K čemu je dobrý? Protokol obecně stanoví pravidla komunikace mezi dvěma partnery – jak jsou přenášené informace uspořádány, co všechno lze protějšku sdělit a jaké mohou být jeho reakce. Konkrétně v případě HTTP se definují zákonitosti dialogu mezi WWW klientem a serverem.

## 25.1 Základní vlastnosti

Již v úvodním seznámení s WWW jsem prozradil, že HTTP je protokol bezstavový. Vychází striktně z modelu dotaz–odpověď. Tato dvojice tvoří ucelenou jednotku komunikace. Jakmile server obdrží dotaz, odpoví na něj a tím pro něj dialog končí. Případné uchovávání stavových informací (typu nacházíme se právě na dokumentu $X$ a z něj jsme vybrali odkaz na $Y$, kdykoli se však můžeme vrátit k předchozímu dokumentu $Q$) je plně ponecháno na klientovi.

HTTP patří mezi protokoly aplikační vrstvy. To znamená, že se zaobírá výlučně otázkami, spojenými s WWW, a neřeší například transport dat mezi partnery. Předpokládá, že pro tento účel má k dispozici spolehlivý[1] transportní protokol. Standardně je používáno TCP, avšak teoreticky může posloužit libovolný protokol s odpovídajícími vlastnostmi. Server typicky bydlí na TCP portu číslo 80.

Situaci lehce komplikuje existence dvou verzí HTTP. V dobách jeho začátků se používal protokol verze 0.9. Jeho charakteristickými znaky jsou velmi omezené vyjadřovací možnosti – například server nemá možnost idenitifkovat typ odesílaných dat; musí jej hádat klient podle přípony nebo obsahu. Současné HTTP nese verzi 1.0 a může se pochlubit podstatně košatějšími vyjadřovacími schopnostmi. V definici je pamatováno i na zpětnou kompatibilitu. Zjednodušeně řečeno zní příslušná pravidla takto:

- Server musí odpovídat stejnou verzí protokolu, ve které byl formulován dotaz. Pokud to umí, samozřejmě.
- Klient musí být schopen vyrovnat se s odpovědí v obou formátech.

[1] To znamená, že se přenos chová jako „bitová roura“ – data, která do ní na jednom konci vkládáte, na druhém vytékají ven v nezměněné podobě. Ručí se za to, že se nic neztratí, nic nezpřehází, nic neduplikuje a podobně.

Servery, podporující jen HTTP 0.9, lze považovat za zastaralé a vřele doporučuji nahradit je novějšími verzemi. Naštěstí těchto staromilců zbývá v současné době již jen hrstka.

Dotaz i odpověď mají shodnou strukturu, která do značné míry připomíná dopisy elektronické pošty. Začíná speciálním dotazovým nebo stavovým řádkem, za ním následují hlavičky s podrobnějšími informacemi a nakonec, odděleno od hlaviček prázdným řádkem, tělo zprávy. Délka zprávy nemá předepsány žádné pevné meze a zmizelo i těch pár omezení, které RFC 822 pro E-dopisy zavádí – především omezení na znaky standardního ASCII kódu a omezená délka řádků. Tělo HTTP zprávy se obecně chová jako binární data. Řadu prvků převzal protokol od nového poštovního standardu MIME, například jeho typologii.

## 25.2 Dotaz

Dotazem všechno začíná. Typická HTTP transakce vypadá následovně:

1. Kdesi v Internetu sedí server a čeká, až jej někdo zavolá.
2. Klient naváže TCP spojení s programem, realizujícím server.
3. Zašle dotaz.
4. Server jej zpracuje, zašle odpověď a ukončí TCP spojení.

Díky předchozí verzi protokolu existují dva přípistné tvary dotazu.

jednoduchý dotaz

*Jednoduchý dotaz* je podporován výlučně pro zachování zpětné kompatibility s HTTP 0.9. Moderní klienti by jej neměli používat. Má tvar

`GET` ◁*URL*▷

a vyjadřuje touhu klienta získat dokument s uvedeným URL.

kompletní dotaz

*Kompletní dotaz* dává podstatně větší možnosti. Vypadá následovně:

◁*metoda*▷ ◁*URL*▷ ◁*verze HTTP*▷
◁*hlavičky*▷

◁*tělo*▷

Přítomnost prvku ◁*verze HTTP*▷ odlišuje kompletní dotaz od jednoduchého. V současné době na jejím místě stojí `HTTP/1.0`. ◁*metoda*▷ určuje druh služby, kterou klient od serveru požaduje. Standardní nabídku metod shrnuje tabulka 25.1 na protější straně.

Sortiment metod obecně není uzavřen a může se postupem času dynamicky měnit. Jestliže klient projeví zájem o metodu, kterou server neumí, odpoví chybovým kódem `501 not implemented`. Na použitou metodu se těsně váže ◁*tělo*▷ dotazu. Je obsaženo jen pokud metoda požaduje jeho přítomnost (ze

| | |
|---|---|
| `GET` | Klient chce získat informaci, určenou ◁*URL*▷. |
| `HEAD` | Má význam podobný, jako `GET`. Server však má poslat jen stavový řádek a hlavičky, nikoli tělo dokumentu. |
| `POST` | Server má předat ◁*tělo*▷ dotazu jako data pro prvek, identifikovaný ◁*URL*▷ dotazu. Metodu `POST` lze používat například pro vytváření nových souborů v adresáři nebo (nejčastěji) pro předání dat ke zpracování CGI skriptem. |

Tabulka 25.1: Metody HTTP

standardní trojice klade tento požadavek pouze `POST`). Drtivá většina současných dotazů používá metodu `GET`. Dotaz s tělem proto uvidíte jen velmi vzácně.

◁*URL*▷ dotazu musí být buď absolutní. Nanejvýš je přípustné vynechat v něm identifikaci serveru – pak se předpokládá, že serverem je ten, komu byl dotaz dopraven. Cesta v ◁*URL*▷ však musí být v každém případě absolutní. Pokud je dotaz předáván zprostředkujícímu (proxy) serveru, musí obsahovat kompletní absolutní ◁*URL*▷.

Nejsložitější části dotazu jsou nepochybně ◁*hlavičky*▷. Budu se jim věnovat o něco později v části 25.4 na straně 319. Všimněte si prádného řádku, kterým jsou ◁*hlavičky*▷ odděleny od těla.

## 25.3 Odpověď

jednoduchá odpověď

Takmé odpověď existuje ve dvou odrůdách. *Jednoduchá odpověď* má své kořeny v HTTP verze 0.9 a obsahuje pouhé ◁*tělo*▷. Server ji používá jen ve dvou případech: Jako odpověď na jednoduchý dotaz nebo pokud nepodporuje HTTP verze 1.0.

kompletní odpověď

Ve všech ostatních případech je povinen poskytnout *kompletní odpověď* ve tvaru

◁*verze HTTP*▷ ◁*kód*▷ ◁*vysvětlení*▷
◁*hlavičky*▷

◁*tělo*▷

První řádek je nazýván *stavový* a oznamuje klientovi výsledek zpracování jeho dotazu. Opět začíná identifikací ◁*verze HTTP*▷, tedy `HTTP/1.0`. Za ním následuje ◁*kód*▷ a ◁*vysvětlení*▷. Tyto dva údaje jsou stranami téže mince. Oba říkají totéž, ale každý je určen pro jiného čtenáře. ◁*kód*▷ slouží WWW klientovi (tedy programu). Jedná se o číselný kód, informující klienta o úspěšnosti či neúspěšnosti zpracování jeho dotazu. Nejčastějším kódem je 200, vyjadřující šťastný konec. ◁*vysvětlení*▷ obsahuje stejnou informaci jako kód, ale v

přirozeném jazyce. Je určen pro lidského čtenáře. Například ke zmíněnému ◁*kódu*▷ 200 se váže ◁*vysvětlení*▷ `OK`.

Kódy jsou rozděleny do pěti tématických skupin podle své první číslice.

**1xx – informační** Tato skupina je rezervována pro budoucí použití. Zatím žádný konkrétní kód nebyl definován.

**2xx – úspěch** Kód začínající dvojkou signalizuje, že dotaz byl serverem pochopen a akceptován. Patří sem

`200 OK` Všechno v pořádku, server posílá odpověď.

`201 Created` Výsedkem zpracování dotazu bylo vytvoření nového objektu, který lze identifikovat pomocí URL. Příslušné URL je tělem odpovědi.

`202 Accepted` Dotaz byl přijat, jeho zpracování však dosud neskončilo. Tento kód se používá zejména při dávkovém zpracování – například si představte, že výsledkem dotazu je zařazení úlohy do tiskové fronty. Klient nemusí čekat na jeho kompletní zpracování (v našem případě dokončení tisku), které se může protáhnout.

`204 No Content` Dotaz byl akceptován a v pořádku obsloužen, nevznikla však žádná data, která by server klientovi poslal.

**3xx – přesměrování** Tato skupina odpovědí signalizuje klientovi, že jeho práce neskončila. Bude muset podniknout další kroky, chce-li dostat odpověď.

`301 Moved Permanently` Objekt byl trvale přestěhován na nové URL (server je oznámí v hlavičce `Location`). Klient se musí zeptat na novém místě.

`302 Moved Temporarily` Objekt byl dočasně přesunut jinam. Klient se musí obrátit na nové místo, neměl by si však například přepisovat URL objektu ve svých záložkách, protože přemístění je jen dočasné.

`304 Not Modified` Tuto odpověď dává server na podmíněný dotaz, o kterém budu mluvit později.

**4xx – klientova chyba** Kód z této kategorie oznamuje, že klient položil chybný dotaz nebo nemá oprávnění, vyžadované k jeho zodpovězení.

`400 Bad Request` Chybná syntax dotazu. Server nerozumět.

`401 Unauthorized` Obsloužení dotazu je vázáno na určité identifikační požadavky, které klient nesplnil.

`403 Forbidden` Server by rád odpověděl, ale doktor mu to zakázal.

`404 Not Found` Objekt s požadovaným URL neexistuje. Tento chybový kód je nejčastější. Příčinou může být buď překlep v URL nebo zánik objektu.

**5xx – chyba serveru** Dotaz byl v pořádku, server však není z nějakého důvodu schopen jej obsloužit.

`500 Internal Server Error` Během zpracování dotazu došlo v programu realizujícím server k jakési blíže neurčené chybě.

`501 Not Implemented` Takto reaguje server, pokud po něm chcete metodu, kterou neovládá.

`502 Bad Gateway` Chybu 502 pošle zprostředkující server, pokud na váš dotaz dostal od původního serveru špatnou odpověď.

`503 Service Unavailable` Server momentálně nedokáže váš dotaz obsloužit – například je přetížen nebo právě probíhá jeho údržba. Chyba je dočasná a pokud později svůj dotaz zopakujete, máte šanci, že budete vyslyšeni.

## 25.4 Hlavičky

Formát, ve kterém jsou hlavičky zapisovány, si HTTP vypůjčil od elektronické pošty, přesněji z RFC 822. Každá položka hlavičky je zapisována na samostatný řádek. Začíná vždy jménem, identifikujícím hlavičku. Za ním následuje dvojtečka, mezera a vlastní hodnota hlavičky – například

```
Pragma: no-cache
```

Hlavičky jsou rozděleny do tří tematických skupin. *Obecné* poskytují univerzální informace o zprávě. *Hlavičky dotazu/odpovědi* poskytují informace, specifické pro dotaz nebo odpověď. V jejich sortimentu se odlišuje dotaz od odpovědi, zbývající dvě skupiny se shodují pro oba druhy zpráv. *Hlavičky těla* popisují tělo zprávy. Na pořadí samozřejmě nezáleží, nicméně ve standardu se doporučuje, aby hlavičky zprávy byly uspořádány podle svých tematických kategorií v uvedeném pořadí.

### 25.4.1 Obecné hlavičky

`Date` informuje o okamžiku vytvoření zprávy. Doporučuje se používat pro zápis času formát, definovaný v RFC 822:

```
Sun, 06 Nov 1994 08:49:37 GMT
```

`MIME-Version` která verze MIME byla použita k vytvoření zprávy. Implicitní hodnotou je `1.0`.

`Pragma` umožňuje přidávat ke zprávě instrukce, jejichž zpracování je implementačně závislé. Například zmíněné

```
Pragma: no-cache
```

zakazuje umístit zprávu ve vyrovnávací paměti.

### 25.4.2 Hlavičky dotazu

`Authorization` slouží pro autentifikaci uživatele.

`From` elektronická adresa uživatele, řídícího činnost klienta. Lze ji použít například pro zaznamenávání transakcí.

`If-Modified-Since` se používá především pro aktualizaci obsahu vyrovnávací paměti. Je-li přítomna tato hlavička, je dotaz považován za podmíněný. Hodnotou hlavičky je časový údaj a dotaz znamená „dokument mne zajímá jen pokud se změnil od tohoto okamžiku“. Jestliže v dokumentu není nic nového, server odpoví `304 Not Modified`. V takovém případě lze použít verzi z vyrovnávací paměti. Došlo-li ke změně dokumentu, chová se server stejně, jako při normálním nepodmíněném dotazu.

`Referer` oznamuje URL zdroje, ze kterého pochází URL právě kladeného dotazu. Pokud například ze stránky `http://www.kdesi.cz/home.html` uživatel vybere značku `<A HREF="top.html">`, vznikne dotaz

```
GET http://www.kdesi.cz/top.html HTTP/1.0
Referer: http://www.kdesi.cz/home.html
```

Tato informace může být užitečná například při chybných odkazech. Jestliže se změnilo URL některého dokumentu, můžete díky hlavičce `Referer` identifikovat stránky, které se odkazují na původní (nyní již neplatné) URL. Kromě toho využívají údaj z `Referer` některá počítadla přístupů, aby se chránila proti čítačovému pirátství (viz strana 162).

`User-Agent` umožňuje klientovi představit se. Pokud zalistujete zpět na stranu 155, kde byla ukázka proměnných prostředí, předávaných CGI skriptu, najdete obsah této hlavičky v proměnné `HTTP_USER_AGENT`. Z ní vidíte, že server obdržel

```
User-Agent: Mozilla/2.0b3 (X11; I; Linux 1.2.11 i586)
```

Dotaz byl vznesen klientem Netscape (v HTTP se představuje jako Mozilla, v dobách jeho počátků autoři doporučovali, aby se tak vyslovovalo i jméno „Netscape“) verze 2.0 beta 3. Jak vidíte, Netscape podal také informace o operačním systému, ve kterém pracuje.

### 25.4.3 Hlavičky odpovědi

`Location` správné umístění odpovědi. Tvoří doplněk kódů ze skupiny `3xx`, oznamujících přemístění dokumentu. Pokud došlo k přestěhování např. na adresu `http://www.jinde.cz/text.html`, odpoví server při dotazu na původní URL[2]

```
HTTP/1.0 301 Moved Permanently
Location: http://www.jinde.cz/text.html
```

`Server` identifikuje programové vybavení serveru. Pokud klient zjistí, že server používá „ten pravý“ program, může pak požadovat nějaké nestandardní služby.

[2]Přesněji řečeno správce může zpravidla server konfigurovat tak, aby odpověděl...

`WWW-Authenticate` stanoví způsob prověrky oprávnění pro daný cíl. Tato hlavička je povinným doplňkem návratového kódu `401 Unauthorized`.

### 25.4.4 Hlavičky těla

`Allow` seznam metod, které lze použít pro získání dokumentu – např.

```
Allow: GET, HEAD
```

`Expires` čili „spotřebujte do“. Hodnotou této hlavičky je časový údaj, udávající okamžik vypršení platnosti dokumentu. Po jeho uplynutí je dokument považován za neplatný a je zakázáno ukládat jej do vyrovnávacích pamětí.

`Last-Modified` okamžik poslední změny v datech.

`Content-Encoding` je jednou z hlaviček, převzatých od poštovního MIME standardu. Oznamuje, jakým způsobem je kódováno tělo dokumentu. Hodnotou je název některého z MIME kódování, řekněme

```
Content-Encoding: x-gzip
```

`Content-Length` udává délku těla v bajtech (přesněji oktetech).

```
Content-Length: 5216
```

`Content-Type` je velmi důležitá pro zpracování dokumentu. Ohlačuje klientovi, jakého druhu jsou zasílané informace a jak s nimi jest zacházeti. Hodnotou je oblíbená MIME dvojice ◁*typ*▷/◁*podtyp*▷. Odesílá-li server WWW stránku, ponese hlavičku

```
Content-Type: text/html
```

## 25.5 Stůj! Kdo tam?

V závěrečné části se podíváme, jak HTTP řeší otázky prokazování totožnosti a autorizace přístupu k dokumentů. Používá celkem jednoduchý model, založený na principu výzva–odpověď. Pokud se klient pokusí získat dokument, přístupný jen autorizovaným uživatelům, reaguje server známým zvoláním, které jsem použil pro nadpis kapitoly. Přesněji řečeno pošle odpověď, která vypadá asi takto:

```
HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="zamestnanci"
```

Základem odpovědi je návratový kód `401`, ozanující, že klient není oprávněn získat dokument. Jeho *povinným* doplňkem je hlavička `WWW-Authenticate`, poskytující klientovi údaje, potřebné k prokázání totožnosti. Obecný tvar této hlavičky je

◁*schéma*▷ `realm="`◁*oblast*▷`"`

◃*schéma*▹ definuje způsob prokazování totožnosti – především informace, které si mezi sebou vyměňují server s klientem. Jediným schématem, podporovaným v současné verzi HTTP, je `Basic`.

Definice ◃*oblasti*▹ umožňuje klientovi odpovídat na příští autentifikační dotazy automaticky. Dokumenty uvnitř serveru jsou totiž rozloženy do tak zvaných *chráněných prostorů*. Každý z nich má společnou databázi uživatelů a pro všechny dokumenty z jednoho prostoru používá klient stejné autentifikační informace. Uživatel by jistě příliš nejásal, kdyby se na ně klient ptal pokaždé znovu. Proto si klient zapamatuje, že pro daný chráněný prostor uživatel zadal určité jméno a heslo a pro příští dokumenty z téhož chráněného prostoru použije automaticky stejné údaje. K jednoznačné identifikaci chráněného prostoru slouží dvojice: kanonické kořenové URL serveru (obsahuje jen jméno/adresu serveru a číslo portu, chybí cesta a vše za ní) a ◃*oblast*▹, definovaná v hlavičce `WWW-Authenticate`.

Tím jsme se propracovali k reakci klienta. Ten zopakuje svůj předchozí dotaz, ovšem přidá k němu hlavičku `Authorization`. Její obsah zcela závisí na použitém ◃*schématu*▹.

Jediné v současnosti dostupné schéma `Basic` používá pro autorizaci klasické údaje – jméno uživatele a jemu příslušející heslo. Klient je ohlašuje v hlavičce `Authorization` takto:

`Basic` ◃*uživatel*▹`:`◃*heslo*▹

Závěrečná dvojice ◃*uživatel*▹`:`◃*heslo*▹ je přepravována v kódování BASE64, což je jedno ze standardních MIME kódování, používaných pro přenos binárních dat. Ve skutečnosti vypadá příslušná hlavička třeba následovně:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

**BASE64 je kódování přepravní, nikoli kryptografické!** Ačkoli text vypadá na první pohled zcela nesrozumitelně, algoritmy a programy pro jeho dekódování do původní podoby jsou veřejně přístupné. Rozhodně nepropadejte naději, že přenášené jméno a heslo je bezpečné. Pro zajištění zabezpečeného přenosu informací je třeba použít jiný mechanismus (např. Secure HTTP nebo Secure Socket Interface).

Když server získá autorizační informace (v našem případě jméno a heslo), nahlédne do databází, příslušejících danému chráněnému prostoru. Pokud v nich najde uživatele a zjistí, že heslo je správně zadáno, a pokud zjistí, že dotyčný uživatel je oprávněn získat požadovaný dokument, pošle standardní odpověď. V opačném případě odpoví `403 Forbidden` a dává tak na vědomí, že poskytnutá autentifikace k získání dokumentu nestačí.