

# Redes de Computadores

## O protocolo HTTP

Departamento de Informática da  
FCT/UNL

# Objetivo do Capítulo

- O protocolo HTTP (*Hyper Text Transfer Protocol*) é o principal protocolo usado para a distribuição de recursos informativos e para suporte de aplicações na Internet
- É um protocolo cliente / servidor relativamente simples mas muito extensível
- O objetivo desta lição é estudar os mecanismos de base do HTTP

*Anyone who has lost track of time when using a computer knows the propensity to dream, the urge to make dreams come true and the tendency to miss lunch.*

*- Autor: Sir Tim Berners-Lee, inventor da Web*

# História da Web

- Até aos anos 80
  - A Internet só era usada por académicos (*login* remoto, e-mail, transferência de ficheiros) e os computadores eram muito caros
  - As interfaces eram predominantemente na linha de comando
- Fim dos anos 80 e princípio dos anos 90
  - A ideia de base da WEB e a linguagem HTML são propostas por Sir Tim Berners-Lee (CERN, Suíça)
  - Com o objetivo de servir as necessidades de acesso e distribuição de informação dentro do CERN
- Anos 90
  - 1991: primeira versão de um browser (NeXT machine)
  - 1993: o primeiro browser para PC - Mosaic
  - Os PCs tornam-se populares e relativamente potentes
  - O acesso comercial à Internet generaliza-se

# Objetivo

- Definir um protocolo de transferência de recursos informativos entre um cliente e um servidor
  - capaz de permitir a transferência de qualquer tipo de informação
  - independente dos sistemas de operação
  - genérico
  - capaz de também transferir a meta - informação associada ao recurso informativo
- Recurso é aqui tomado num sentido muito genérico
  - uma sequência de dados com um tipo (o que permitia saber como a interpretar)

# Problemas a Resolver

- Designar um documento de forma normalizada e independente da sua localização;
- Estabelecer referências normalizadas de um documento para outro documento de forma independente da sua localização;
- Definir uma linguagem de descrição de documentos normalizada na qual os diversos documentos fossem codificados;
- Definir um protocolo de acesso a documentos remotos genérico e normalizado; e
- Implementar um demonstrador do sistema independente do sistema de operação.

# Soluções

- Introduziu os URLs (Uniform Resource Locators)
- Hyper texto para permitir o estabelecimento de relações entre documentos ou recursos usando os URLs
- Introduziu a linguagem HTML (Hyper Text Markup Language) uma linguagem de descrição genérica de documentos formatados
- Introduziu o protocolo HTTP (Hyper Text Transfer Protocol)

# Hello World em HTML

```
<!DOCTYPE html>  
<html>  
<head> <title>Page Title</title> </head>  
<body>  
<h1>HTML Hello World</h1>  
<p>Isto parece excitante!</p>  
<a href="http://www.w3schools.com/html/">Siga um tutorial  
sobre HTML</a>  
</body>  
</html>
```

## HTML Hello World

Isto parece excitante!

[Siga um tutorial sobre HTML](http://www.w3schools.com/html/)



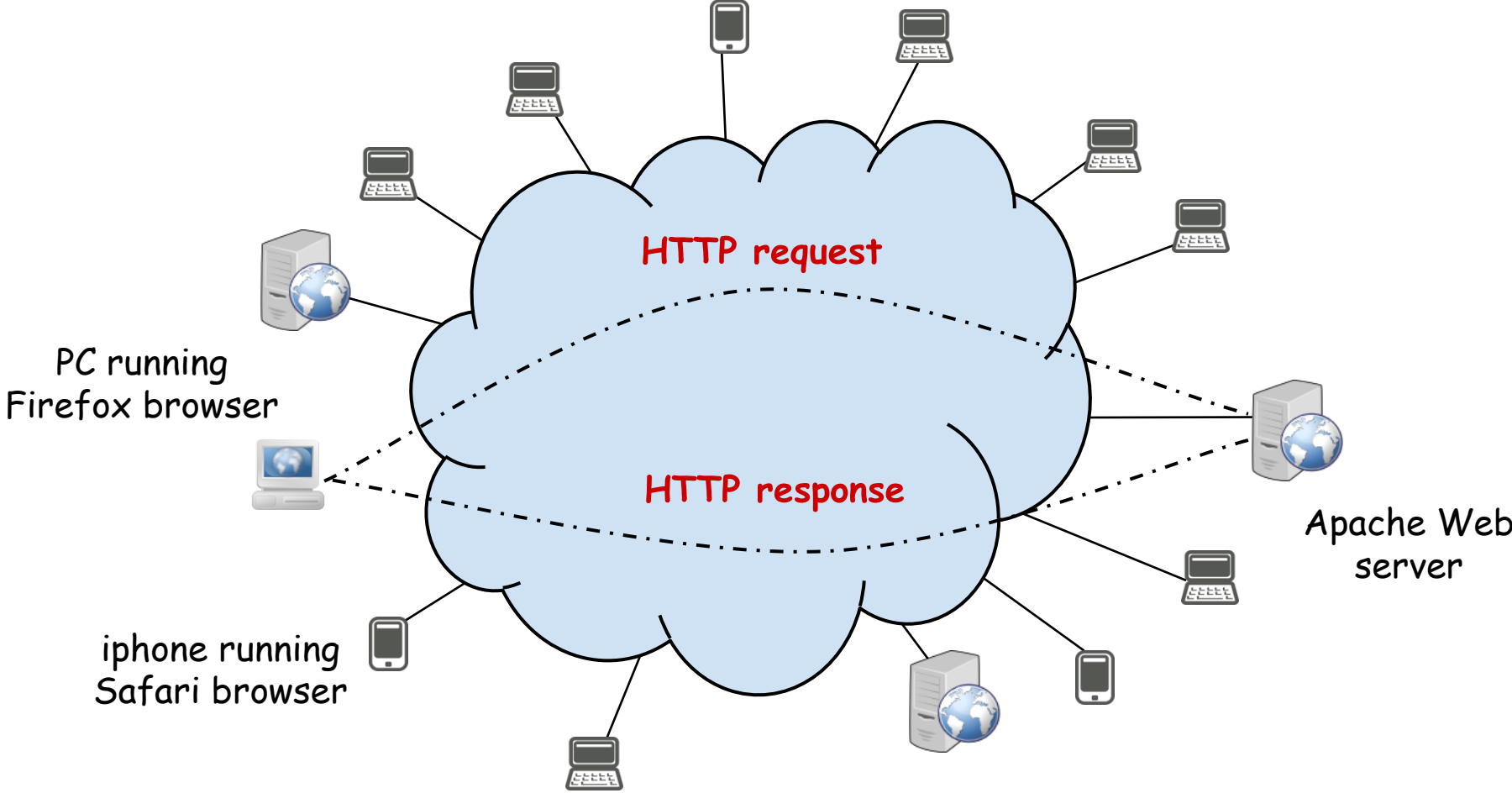
# Terminologia

- Uma página WEB é composta por vários objetos ou recursos informativos
  - Geralmente um conteúdo de base, codificado na linguagem HTML, contendo referências a outros objetos como: imagens, ficheiros, áudio ou vídeo, código executável no browser, outras páginas contendo código HTML, etc.
- Qualquer um desses recursos tem um endereço, as referências HTML, designado popularmente por URL
  - Que contém o nome do servidor onde o objeto reside, seguido do nome do objeto local ao servidor, eventualmente seguido de parâmetros

# Nomes, identificadores e endereços

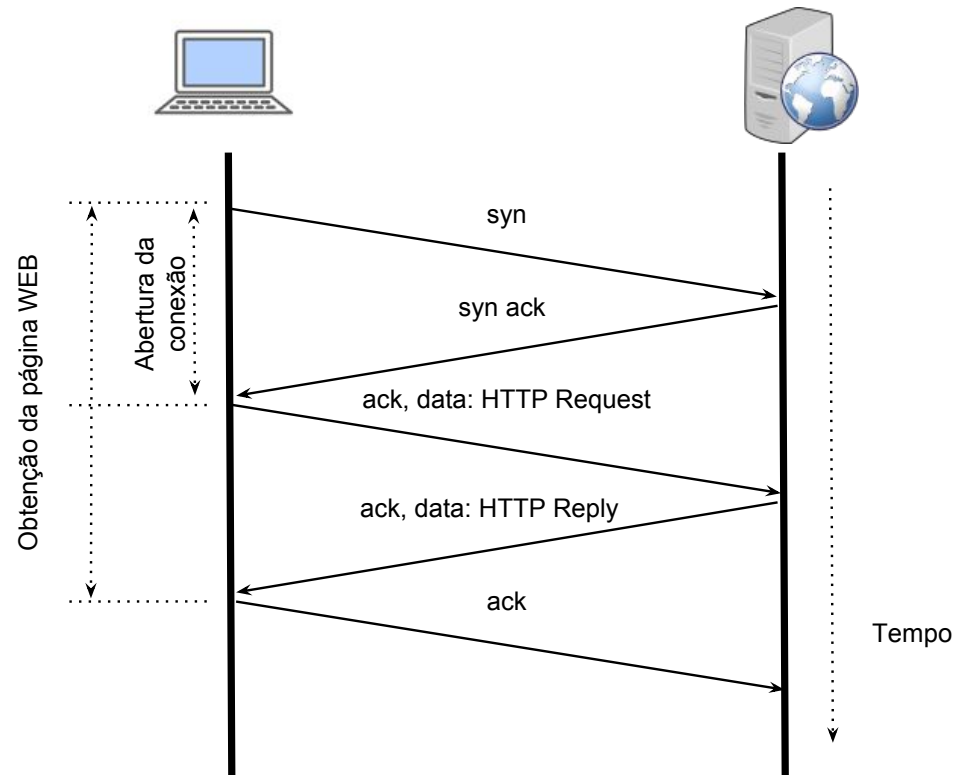
- Nome, Localizador (URL)
  - Nome: um nome único global para um recurso que o identifica inequivocamente, e.g. ISBN de um livro
  - URL: uma localização do conteúdo do livro (e.g. cópia pdf)
- Os URLs são escritos de uma forma bem definida
  - Protocolo para comunicar com um servidor (e.g., http)
  - Nome do servidor (e.g. www.fct.unl.pt)
  - Nome do recurso (e.g., coolpicture.gif)
- Sintaxe
  - Protocolo de acesso://nome do servidor[:porta]/nome-do-objecto
- Exemplo
  - `http://www.fct.unl.pt/coolpicture.gif`

# Hyper Text Transfer Protocol



# O HTTP Usa uma Conexão TCP

- O browser abre uma conexão TCP para o servidor
- Envia então a mensagem *HTTP Request*
- O servidor responde com a mensagem *HTTP Reply*



# Funcionamento

O que se passa quando é solicitado a um browser que aceda a um recurso indicado por um URL?

Exemplo: `http://en.wikipedia.org/wiki/HTML`

- 1) O browser faz a análise do URL
- 2) Solicita ao DNS o endereço do servidor (`en.wikipedia.org`)
- 3) O DNS responde com `145.97.39.155` por exemplo
- 4) O browser abre uma conexão TCP para o porto 80 de `145.97.39.155`
- 5) Envia então o comando: `"GET /wiki/HTML HTTP/1.0"` seguido de uma linha em branco
- 6) O servidor responde com esse documento
- 7) O browser lê o documento através do canal TCP
- 8) O servidor e o browser fecham a conexão
- 9) O browser começa a interpretação do documento e abre novas conexões para ir buscar as imagens e outros documentos indicados no mesmo

# Mensagens HTTP

**GET /index HTTP/1.0 CRLF**

Host: [www.some-example.com](http://www.some-example.com) CRLF

User-Agent: Mozilla/40.01 CRLF

CRLF

**HTTP Request  
message**

**HTTP/1.1 200 OK CRLF**

Date: Mon, 21 Feb 2015 17:51:37 GMT CRLF

Server: Apache/2.4.10 (Debian) CRLF

Last-Modified: Wed, 17 Aug 2014 17:46:43 GMT CRLF

ETag: "1267-5272fc1726880" CRLF

Accept-Ranges: bytes CRLF

Content-Length: 4768 CRLF

Vary: Accept-Encoding CRLF

Connection: close CRLF

Content-Type: text/html CRLF

CRLF

.....

**HTTP Reply  
message**



Tempo

# Meta Dados ou Atributos dos Recursos

- Meta dados ou atributos
  - Informações sobre um recurso ou atributos do recurso
  - ... mas que não fazem parte do mesmo
- Exemplos
  - Dimensão
  - Tipo e codificação do conteúdo
  - Data da última modificação
- Inspirado dos protocolos desenvolvidos para o e-mail
  - *Multipurpose Internet Mail Extensions (MIME)*
  - Tipo do conteúdo (e.g., Content-Type: text/html) que permite ao *browser* lançar imediatamente um visualizador adequado ao tipo (e.g. PDF, MPEG, ...)

# Estrutura da Mensagem HTTP Request

- *Request headers*

- Legíveis pelos humanos (ASCII), de comprimento variável

- Utilizações:

- Authorization - dados para autenticação do utilizador
    - Tipo e codificação do conteúdo
    - If-Modified-Since
    - User-Agent - o software do cliente
    - Etc etc etc

- *Linha em branco*

- *Body (corpo)*



# Formato do *HTTP Request*

*Request line or  
method  
(GET, POST,  
HEAD, ....)*

*Header  
lines*

*Dois carriage  
return,  
line feeds  
indicam o fim da  
mensagem (linha em  
branco)*

*Notas: CRLF representa os  
códigos dos caracteres de  
controlo carriage return e  
line feed*

**método /nome-local-do-objecto versão do  
protocolo CRLF  
header field name: field value CRLF  
header field name: field value CRLF  
.....  
header field name: field value CRLF  
CRLF  
[ entity body ]**

Exemplo:

```
GET /wiki/Main_Page HTTP/1.0 CRLF
Host: en.wikipedia.org CRLF
User-Agent: Mozilla/44.01 CRLF
CRLF
```

# Utilização das Linhas do Cabeçalho

- Exemplo: o cliente só pretende o objecto caso este tenha sido modificado

```
GET /HTTP/1.1 CRLF
Host: asc.di.fct.unl.pt CRLF
User-Agent: Mozilla/4.03 CRLF
If-Modified-Since: Mon, 6 Feb 2009 11:12:23 GMT CRLF
CRLF
```

- O servidor evita assim enviar dados inúteis
  - O servidor analisa a data da última atualização do recurso
  - ... e compara-o com a data a seguir a "if-modified-since"
  - Responde "304 Not Modified" se o recurso não foi alterado
  - ... ou "200 OK" seguido do conteúdo se este foi modificado após a data indicada

# Alguns Header-fields

<b>Header-fields</b>	<b>Exemplo</b>
<b>User-Agent</b>	<b>User-Agent: Mozilla/40.0</b>
<b>Accept-Charset</b>	<b>Accept-Charset: utf-8</b>
<b>Accept-Encoding</b>	<b>Accept-Encoding: gzip</b>
<b>Accept-Language</b>	<b>Accept-Language: en-UK</b>
<b>If-Modified-Since</b>	<b>If-Modified-Since: Tue, 02 Feb 2016 14:25:41 GMT</b>
<b>If-Match</b>	<b>If-Match: "756154ad8d .... 3102f1349317f"</b>
<b>Range</b>	<b>Range: bytes=500-999</b>
<b>.....</b>	

# Formato Geral do *HTTP Reply*

*Status line*

(Version, status code, optional status message)

*Zero ou mais header lines*

Separador  
(2 CRLF)

Object data

```
versão código frase CRLF
header field name: field value CRLF
header field name: field value CRLF
header field name: field value CRLF
.....
header field name: field value CRLF
CRLF
[ entity body ]
```

Notas: CRLF representa os códigos dos caracteres de controlo *carriage return* e *line feed*

# Exemplo de *HTTP Reply*

status line  
(protocol  
status  
code or  
status  
phrase)

header  
lines

Separador  
(2 CRLF)

dados, e.g.,  
requested  
HTML file

```
HTTP/1.1 200 OK CRLF
Date: Mon, 21 Feb 2015 17:51:37 GMT CRLF
Server: Apache/2.4.10 (Debian) CRLF
Last-Modified: Wed, 17 Aug 2014 17:46:43 GMT CRLF
ETag: "1267-5272fc172688" CRLF
Accept-Ranges: bytes CRLF
Content-Length: 4768 CRLF
Vary: Accept-Encoding CRLF
Connection: close CRLF
Content-Type: text/html CRLF
CRLF
dados, dados, ...
```

# Alguns Header-fields

<b><i>Header-fields</i></b>	<b>Exemplo</b>
<b>Server</b>	<b>Server: Apache</b>
<b>Last-Modified</b>	<b>Last-Modified: Tue, 02 Feb 2016 14:25:41 GMT</b>
<b>Content-Type</b>	<b>Content-Type: text/html; charset=utf-8</b>
<b>Content-Length</b>	<b>Content-Length: 348</b>
<b>Content-Encoding</b>	<b>Content-Encoding: gzip</b>
<b>ETag</b>	<b>ETag: "3d2-52aca46b79fd9"</b>
<b>Accept-Ranges: bytes</b>	<b>Accept-Ranges: bytes</b>
<b>.....</b>	<b>.....</b>

# Métodos ou Comandos

Nome	Descrição
GET	Pedido do objeto identificado no URL
HEAD	Pedido do objeto identificado no URL, mas o cliente apenas pretende obter o cabeçalho da resposta
TRACE	Ecoa o pedido do cliente para apoio ao <i>debug</i>
OPTIONS	Ecoa os comandos e opções que o servidor aceita para um certo URL (muitos servidores ignoram este comando por segurança)
POST	Permite ao cliente transmitir um conjunto de dados para o servidor, geralmente contendo parâmetros de uma operação (e.g., envio de um formulário HTML, comentário numa discussão, valor a colocar numa base de dados, . . . )
PUT	Introdução ou substituição do objeto identificado no URL
DELETE	Supressão do objeto identificado no URL

# Exemplos de códigos de resposta

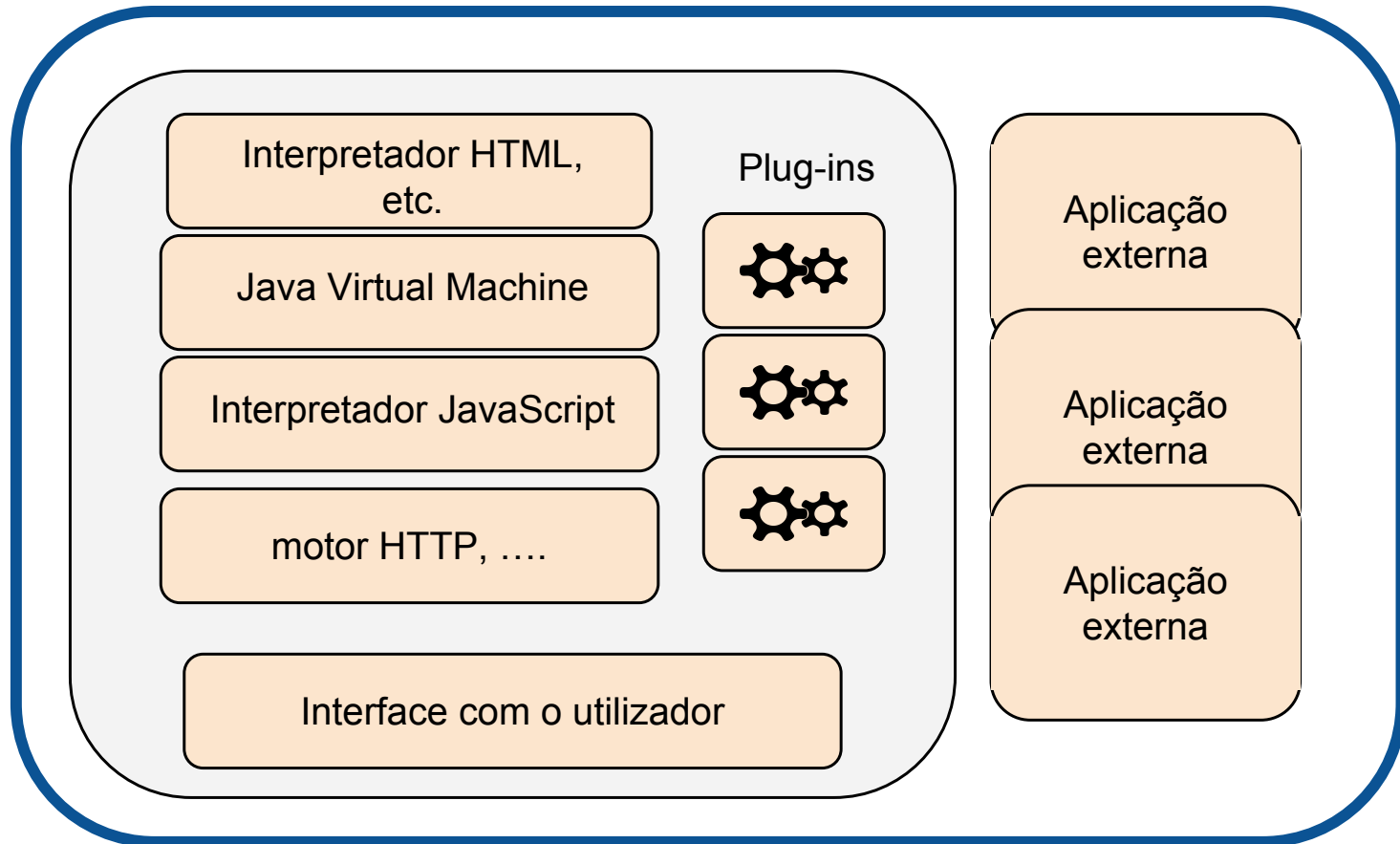
<b>Código</b>	<b>Exemplo</b>	<b>Descrição</b>
<b>1xx</b>	<b>Códigos informativos</b>	
<b>100</b>	<b>Continue</b>	<b>Um objeto vai ser devolvido mas ainda está a ser gerado</b>
<b>2xx</b>	<b>Códigos de sucesso</b>	
<b>200</b>	<b>OK</b>	<b>Sucesso</b>
<b>204</b>	<b>OK but no content</b>	<b>O objeto existe mas está vazio</b>
<b>3xx</b>	<b>Códigos de redirecção</b>	
<b>301</b>	<b>Moved permanently</b>	<b>O objeto tem um novo URL</b>
<b>4xx</b>	<b>Códigos de erro do cliente</b>	
<b>400</b>	<b>Bad request</b>	<b>Pedido não reconhecido (sintaxe ....)</b>
<b>5xx</b>	<b>Códigos de erro do servidor</b>	
<b>501</b>	<b>Not implemented</b>	<b>O servidor não suporta o método</b>



# Alguns Tipos dos Objetos

<b>Tipo</b>	<b>Descrição</b>
<b>Text/plain</b>	<b>Texto não formatado</b>
<b>Text/html</b>	<b>Texto HTML</b>
<b>Image/jpeg</b>	<b>Imagem em JPEG</b>
<b>Video/mpeg</b>	<b>Vídeo em MPEG</b>
<b>Application/octetstream</b>	<b>Objeto opaco correspondente geralmente a algo executável por um programa, muitas vezes externo</b>
<b>Application/postscript</b>	<b>Documento codificado em postscript</b>
<b>Application/java-vm</b>	<b>Java bytecode file</b>
<b>Application/javascript</b>	<b>JavaScript file</b>
<b>Application/vnd.ms-excel</b>	<b>Ficheiro Excel</b>
<b>Application/json</b>	<b>JavaScript Object Notation (JSON)</b>
<b>.....</b>	

# Tipos dos Objetos e Browsers



# HTTP Request Manual com Telnet

```
$ telnet en.wikipedia.org 80  
Trying 91.198.174.192...  
Connected to en.wikipedia.org.  
Escape character is '^]'.  
  
GET / HTTP/1.0  
HTTP/1.1 200 OK  
Server: Apache X-Powered-By: HHVM/3.3.0-static  
Last-Modified: Tue, 02 Feb 2016 14:25:41 GMT  
ETag: "3d2-52aca46b79fd9"  
Content-Type: text/html  
  
.....
```

# Onde Termina a Mensagem ?

- Content-Length
  - O servidor necessita de saber ou calcular dinamicamente a dimensão do recurso
- Fechar a conexão
  - Após o envio, o servidor fecha a conexão
- Dimensão implícita
  - E.g., a mensagem de código 304 não tem dimensão

# Operações Idempotentes

Uma operação idempotente (*idempotent operation*) é uma operação cuja execução repetida, mesmo que devolva um valor diferente em cada caso, não conduz a erros aplicativos, nem no servidor, nem no cliente.

# Servidores Sem Estado (Stateless)

- Se um servidor tem uma interface exclusivamente baseada em operações idempotentes, diz-se um **servidor sem estado** (*stateless server*) e torna-se mais simples porque não tem de detetar operações executadas mais do que uma vez.
- Os servidores DNS só executam operações idempotentes. Parte das operações executadas pelos servidores HTTP também são idempotentes, nomeadamente os métodos GET e PUT.

# O HTTP é Sem Estado (*Stateless*)

- Cada interação é independente das outras e tem de ter todas as indicações necessárias para ser satisfeita
  - Servidor simples - esquece os pedidos a que já respondeu
  - O servidor pode ser substituído entre pedidos (*crash recovery*, distribuição de carga, ...)
  - Pode servir mais clientes e não necessita de se preocupar em saber se eles "ainda lá estão"
  - Se o recurso é estático, pode-se fazer *caching* do mesmo e replicá-lo sem limites e sem problemas
  - Estas características também justificam o sucesso do HTTP
- Mas tal complica o desenvolvimento das aplicações
  - Certas aplicações necessitam de estado (e.g. carrinho de compras, ...)
  - Que fazer ? Veremos a seguir

# Conclusões

- HyperText Transfer Protocol (HTTP)
  - Protocolo cliente servidor para acesso a recursos remotos
  - O cliente envia o pedido, o servidor a resposta
- Propriedades importantes
  - Cliente servidor
  - Sobre TCP
  - Utilização de identificadores e localizadores de recursos (URI e URL)
  - Mensagens com comandos, respostas e cabeçalhos em ASCII e extensíveis
  - Agnóstico ao conteúdo transportado
  - Meta dados dos recursos nas mensagens
- Protocolo sem estado (*stateless*)