



USP - Universidade  
de São Paulo



IME - Instituto de  
Matemática e Estatística

**MAC0332**  
**Engenharia de Software**

# Introdução ao Desenvolvimento para Web

Prof. Marco Aurélio Gerosa  
[gerosa@ime.usp.br](mailto:gerosa@ime.usp.br)



# Evolução da tecnologia

- Processamento de dados  
↓  
Sistemas de Informação  
↓  
Sistemas de comunicação
- Douglas Engelbart [1968] já previu
- Software hoje é social.



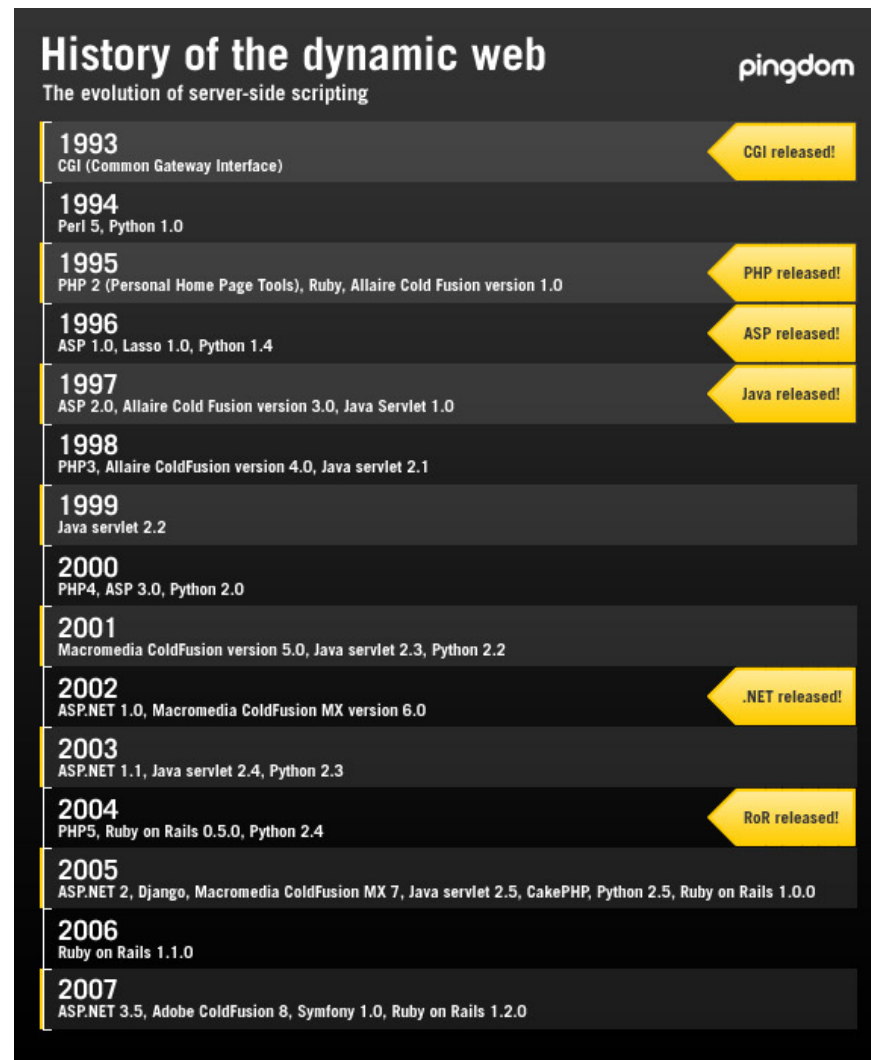


# Histórico

- Ted Nelson concebe o hipertexto (**1963**)
- Douglas Engelbart desenvolve o primeiro sistema de hipertexto (**1966**)
- Surgimento da Web – Tim Berners-Lee (CERN - European Organization for Nuclear Research/Suíça) (**1990/1991**)
- A CERN anuncia que a Web iria ser aberta e gratuita para todos. Como o protocolo do Gopher deixou de ser gratuito, ocorreu rápida migração para tecnologia Web (**1993**)
- Surgimento do Mosaic – navegador Web (**1993**)
- Estabelecimento do W3C (World Wide Web Consortium) (**1994**)
- Netscape 1.0 primeiro browser comercial (Final de **1994**)
- O tráfego de Web na rede supera o de outros protocolos, como Telnet e FTP (**1995**)
- Lançamento do Microsoft Internet Explorer (**1995**)
- Liberação do código do Netscape Navigator (**1998**)
- Estouro da Bolha Dot Com (**2001**)
- Popularização da banda larga
- Surgimento do termo Web 2.0 (O'Reilly Media) (**2003**)



# História da Web dinâmica



<http://royal.pingdom.com/2007/12/07/a-history-of-the-dynamic-web/>



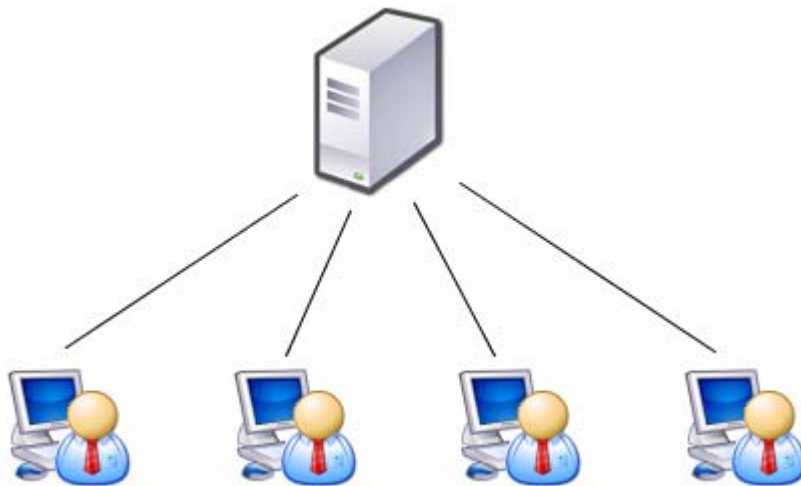
# Pilha de protocolos da Web

- A Web
  - Markup language para hipertexto
  - Notação uniforme para acesso de recursos
  - Um protocolo para transportar dados
- HyperText Markup Language (HTML)
- Uniform Resource Locator (URL) ou Uniform Resource Identifier (URI)
- URL
  - `scheme://host[:port]/path/.../[/;url-params][?query-string][#anchor]`
- Protocolo HTTP
  - Modelo OSI vs TCP/IP





# Arquitetura cliente-servidor



C:\> Administrator: C:\Windows\system32\cmd.exe

```
220 galena.ime.usp.br ESMTP
helo
250 galena.ime.usp.br
mail gerosa@ime.usp.br
250 ok
rcpt profgerosa@gmail.com
250 ok
data
354 go ahead
Esta e' uma mensagem.
.
250 ok 1281665769 qp 11533
quit
221 galena.ime.usp.br
```

Connection to host lost.

C:\Users\MarcoGerosa>



# Arquitetura

- Protocolos
  - Stateful vs Stateless
  - Ex: SMTP vs HTTP
  - Serviços session-based vs session-less
- Cliente
  - Lightweight (ou thin-client)
- Como o cliente e servidor se comunicam?
  - Sockets (TCP connection)
  - Portas
- Portas padrões
  - HTTP 80
  - SMTP 25
  - FTP 21
  - Telnet 23
  - IMAP 143
  - Etc.



# MIME types

- Originalmente email só era usado para transmissão de ASCII
- Criação de uma padronização para instruir o cliente sobre o tipo de anexo (sequência de bytes enviadas)
  - Multipurpose Internet Mail Extension
  - Originalmente definida na RFC 1341. As especificações mais recentes estão nas RFCs 2045 a 2049
  - Ex de MIMEs types padronizados:
    - text/html
    - text/plain
    - image/jpeg
    - audio/mp3
    - application/pdf
    - video/quicktime





# O protocolo HTTP

- Simplicidade !
- requests e responses
- Estrutura das mensagens HTTP:
  - Cabeçalhos
  - Linha em branco
  - Corpo
- Exemplo de request:

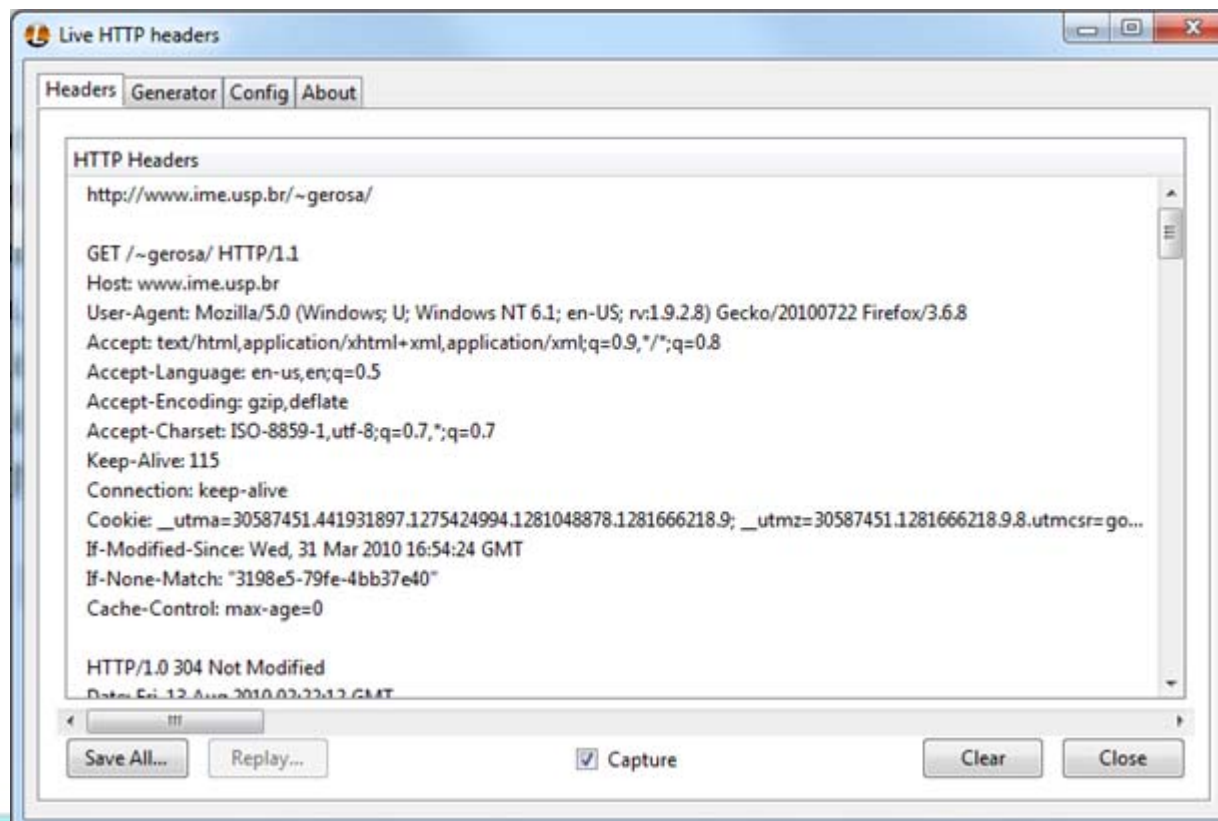
```
GET /diretorio/index.html HTTP 1.1
Host: www.mywebsite.com
```
- Exemplo de response:

```
HTTP/1.1 200 OK
Content-type: text/html
Content-length: 9934
(...)
<html><head><title>Minha página</title></head>
<body>
(...)
</body>
</html>
```



# Método GET

- GET
  - Desde as primeiras versões do HTTP. Não precisa de corpo e no HTTP 1.0 não era necessário cabeçalhos
  - No 1.1 passou a exigir o cabeçalho Host para possibilitar o virtual hosting nos servidores Web





# Método POST

- POST
  - A diferença fundamental para o GET é que os parâmetros são enviados no corpo da mensagem HTTP e não na URL
  - Exemplo:  
POST /recurso HTTP/1.1  
Host: site.com.br  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 6  
  
S=YHOO
- OBS: Algumas aplicações podem tratar diferentemente as requisições dos dois métodos, um deles pode não ser implementado ou pode tratar de maneira indiferente



# Método HEAD

- HEAD
- Retorna somente os cabeçalhos da resposta. Anteriormente era usado para fins de cache.

- Request:

```
HEAD /index.html HTTP/1.1
```

```
Host: www.meusite.com.br
```

- Response:

```
HTTP/1.1 200 OK
```

```
Date: Tue, 08 Apr 2008 15:55:04 GMT
```

```
Server: Apache/2.2.4 (Unix)
```

```
Last-Modified: Tue, 29 Oct 2002 04:22:52 GMT
```

```
Content-Length: 2111
```

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



## Outros Métodos

- PUT – gravar um recurso no servidor
- DELETE – remover um recurso no servidor
- OPTIONS – retorna quais são os métodos com suporte para a data URL
- CONNECT, TRACE, PATCH
- Métodos seguros (não devem alterar o servidor): HEAD, GET, OPTIONS e TRACE



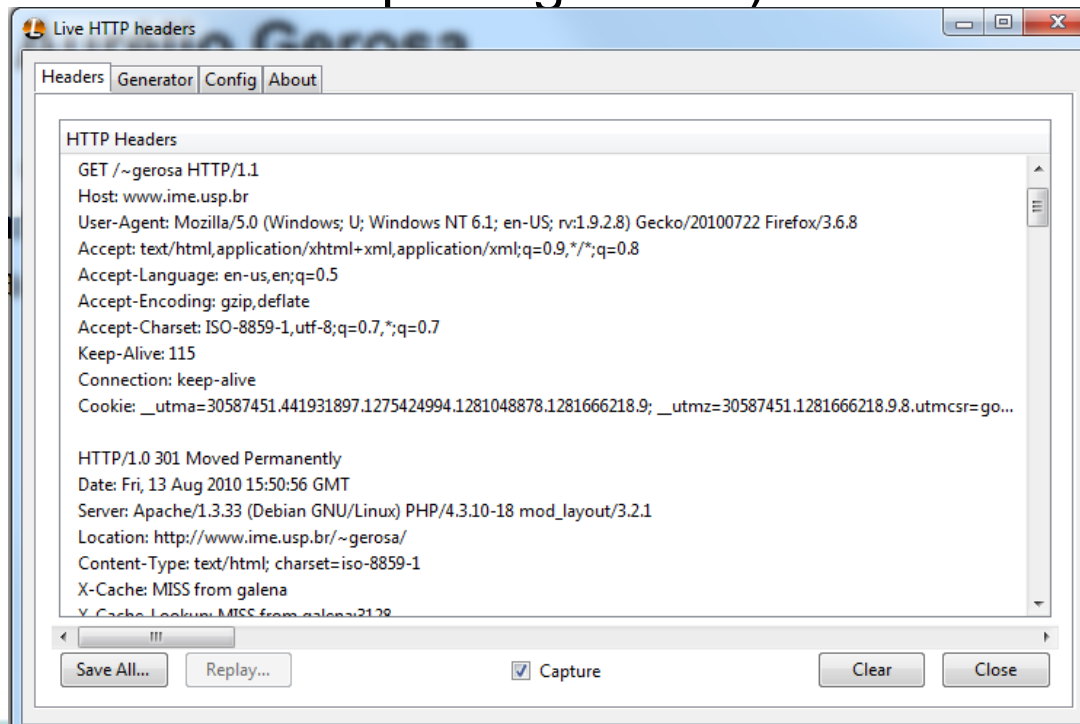
# Status Code

- A primeira linha da resposta contém um código de status de 3 dígitos.
  - 1xx: resposta informacional
  - 2xx: resposta bem sucedida
  - 3xx: pede para o cliente realizar alguma ação adicional
  - 4xx: erro na requisição do cliente
  - 5xx: erro no servidor
- Status 1xx
  - 100: Continue (usado em resposta ao cabeçalho Expect: 100-continue da requisição, usado quando o cliente quer uma confirmação do servidor se pode enviar uma requisição).
- Status 2xx
  - 200: Ok
  - 201: Ok, com a criação de um recurso no servidor



# Status code

- Status 3xx:
  - 301: recurso movido permanentemente
  - 302: recurso movido temporariamente(por exemplo, ao acessar o site:  
<http://www.ime.usp.br/~gerosa> o servidor retorna um 301 para  
<http://www.ime.usp.br/~gerosa/> )





# Status code

- **Status 4xx:**
  - 400 Bad Request
  - 401 Not Authorized
  - 403 Forbidden
  - 404 Not found
  - Etc.
- Exemplo para uso por caches:  
GET /~gerosa/ HTTP 1.1  
Host: www.ime.usp.br  
If-Unmodified-Since: Fri, 11 Feb 2000 22:28:00 GMT  
  
HTTP/1.1 412 Precondition Failed  
Date: Tue, 29 Apr 2008 22:28:31 GMT
- **Status 5xx:**
  - 500 Internal Server Error
  - 501 Not Implemented





# Cabeçalho

- Gerais
  - Date: data de criação da mensagem
  - Connection: {close/keep-alive} – keep-alive é o default no HTTP/1.1
  - Warning: mensagem de depuração – não é tratada pelo software
- Requisição
  - User-Agent: software que fez a requisição
  - Host: possibilita virtualização de domínios
  - Referer: Página onde o usuário estava quando clicou no link
  - Authorization: Transmite as credenciais do usuário (login e senha) em resposta a um 401 (authorization challenge) . Continua transmitindo enquanto acessar recursos daquele subdomínio. Ex: Authorization: Basic eNCoDEd-uSErId:pASsORd
  - OBS: A informação no Authorization é codificada, mas não criptografada. (não há uma chave de segurança). Para aumentar a segurança, a autorização deve ser usada em conjunto com HTTPS



# Cabeçalho

- Resposta
  - Location: acompanha um 301 ou 302 indicando o novo local do recurso
  - WWW-Authenticate: acompanha um 401 (authorization challenge) .  
Exemplo: WWW-Authenticate: Basic realm="Minha Aplicação"
  - Server: Indica o software servidor
- Sobre o corpo
  - Content-Type: MIME type do corpo
  - Content-Length: qtde de bytes do corpo
  - Last-Modified: data da última modificação. Usado para possibilitar o cache no cliente e nos proxies
- Cache
  - Cache-Control: no-cache, private (um proxy deve fazer cache somente para o usuário que requisitou), public (a informação no cache pode ser usada por qualquer usuário)
  - Pragma: no-cache (deprecated – era usado no HTTP 1.0)



# Cabeçalho

- Cookies:
  - Set-Cookie: <name=value> [; Comment=<value>] [; Max-Age=<value>] [; Expires=<date>][; Path=<path>][; Domain=<domain name>][; Secure][; Version=<version>]
  - Set-Cookie2: similar ao Set-Cookie com algumas flexibilidades a mais, como definir a porta
  - Cookie: \$Version="1"; name="value" [; (...)]; \$Path="/path"



# Tecnologias

- Lado Cliente
  - JavaScript
  - Java (applets)
  - Adobe Flash
  - Adobe Flex
- Lado Servidor
  - Java (servets)
  - Porlets
  - ASP.Net
  - PHP
  - Ruby on Rails
  - Python