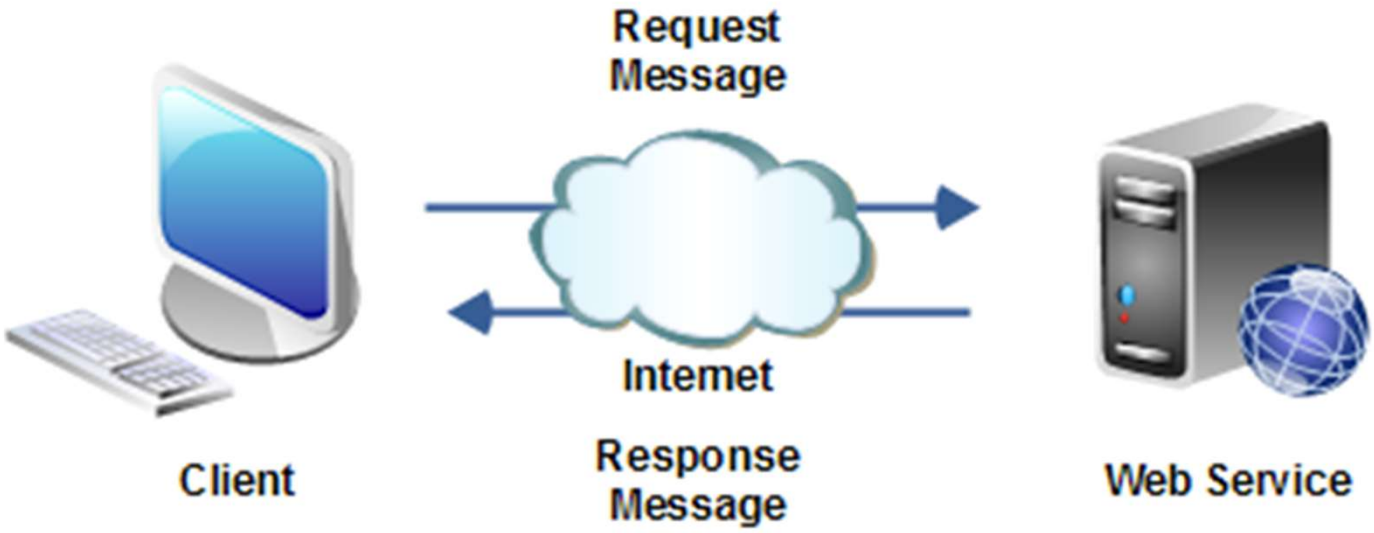



http://tutorials.jenkov.com/web-services/message-formats.html

# APLICAÇÕES E SERVIÇOS WEB

universidade de aveiro
departamento de eletrónica,
telecomunicações e informática
deti

Laboratórios de Informática

# Aplicações WEB

- A maioria de nós utiliza hoje aplicações que não residem no nosso computador
  - Google (Gmail, Docs)
  - Microsoft (Outlook.com, Office 365)
  - Facebook
  - elearning.ua.pt

- Disponibilizadas sobre um interface Web

# Como funciona uma aplicação web

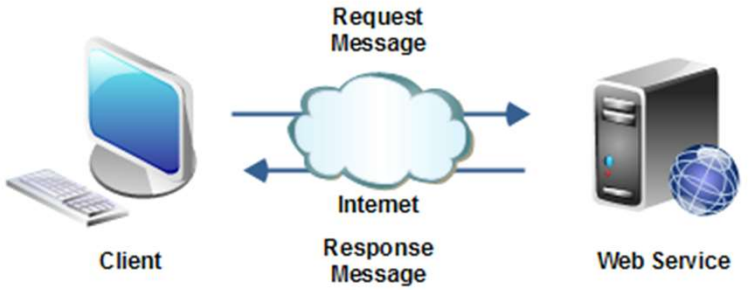
- Através de uma aplicação genérica residente no nosso computador
  - IE, Chrome, Firefox, Opera, etc
- Comunicação com um servidor remoto através do protocolo HTTP
- Servidor remoto capaz de servir centenas/milhares de clientes
  - Apache, IIS, nginx

# O que é um Servidor WEB?

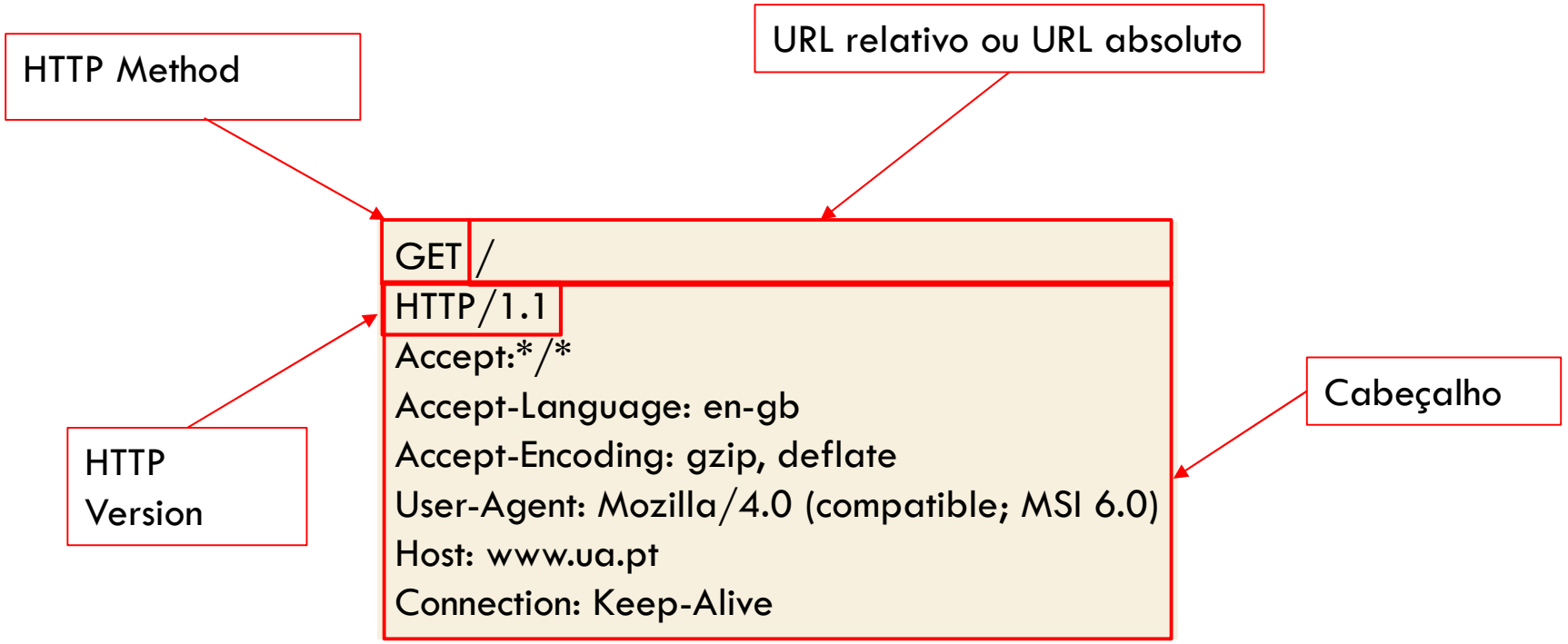
- É um programa de computador que comunica um recurso (página HTML, imagem, vídeo) a um programa cliente (Web Browser) através do uso do protocolo HTTP

# Protocolo HTTP

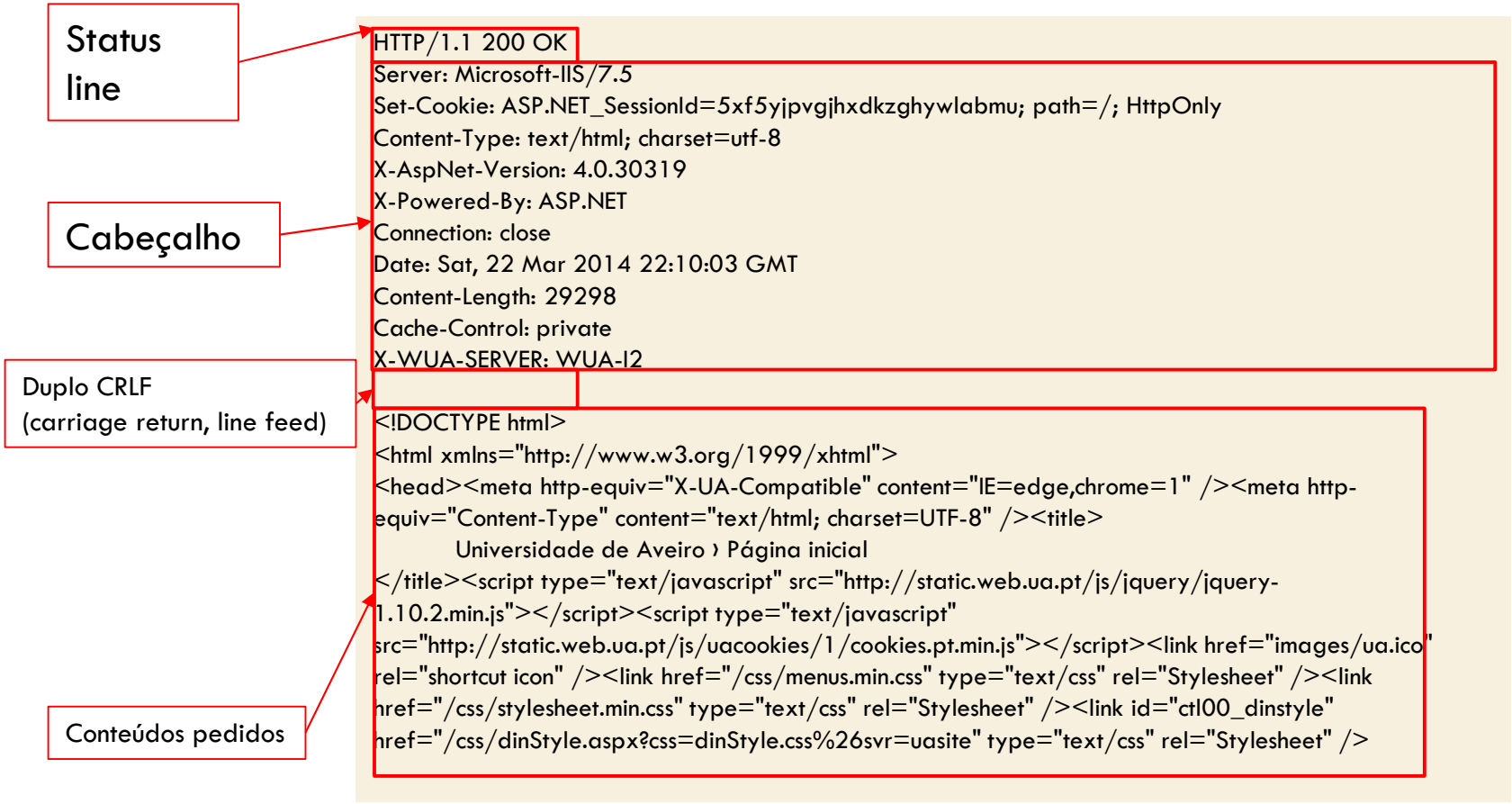
- Protocolo Cliente-Servidor
- Suportado em TCP
- Popularizado em 1990 por Tim Berners-Lee
  - Criação da World Wide Web
- Protocolo baseado em Texto



# HTTP Request



# HTTP Response



# Principais métodos HTTP

- **GET** permite aceder a qualquer informação identificada pelo Request-URI
- **POST** utilizado para enviar informação para o servidor
- **HEAD** idêntico ao GET mas o servidor não deverá enviar o conteúdo do recurso pedido
  - serve para desencadear ações no servidor

# Códigos de Status e Erro

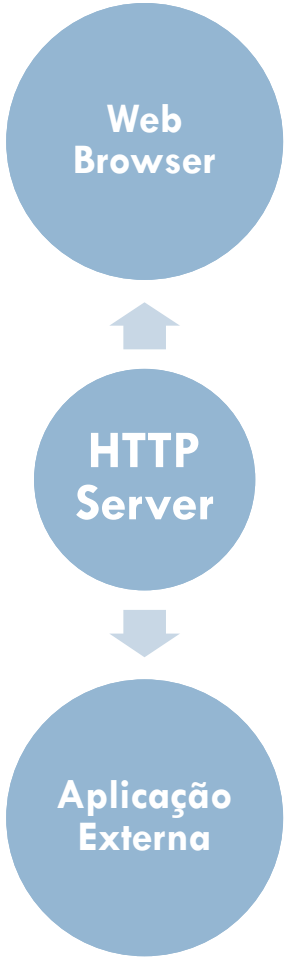
- **1xx** – Informacional – resposta intermédia que indica que o servidor ainda não acabou de processar o pedido
- **2xx** – Bem sucedido
  - 200 OK
- **3xx** – Redireccionamento do cliente para outra localização
  - 301-permanent, 302-temporary
- **4xx** – Erro provocado pelo Cliente
  - 400-bad request, 403-forbidden, 404-not found
- **5xx** – Erro do Servidor
  - 500 Internal Server Error, 503-Service Unavailable,504-Gateway Timeout

# Web App/Service

- A criação de sites dinâmicos que se adaptam ao cliente podem ser alcançados:
  - Manipulação local usando JS do DOM
  - Servidor serve conteúdos criados em função dos pedidos do cliente

- Folhas de estilos (CSS) adequadas ao dispositivo
  - Podem fazer aplicações Web parecer nativas

# CGI – Common Gateway Interface




- O Servidor HTTP recorre a uma aplicação externa para criar os conteúdos a servir ao Cliente
  - Não é um protocolo
  - O programa externo pode ser escrito em qualquer linguagem
- O programa devolve conteúdos (stdout), que são enviados por HTTP

# Problemas associados ao uso de CGI's

- Performance e Segurança
  - Cada pedido feito ao Servidor despoleta um novo processo que executa o programa externo
  - Processo do CGI é terminado quando o programa acaba de executar pelo que não há manutenção de estado de uma execução para outra

# Servidores Aplicacionais

- Principais alternativas ao uso de CGI's
  - Aplicação integrada com o servidor HTTP
    - exemplos: PHP
  - Servidor HTTP comunica por IPC com uma aplicação externa
    - Exemplos: ruby, Python (WSGI)

# WSGI

- Web Service Gateway Interface
  - Interface entre servidores web e aplicações
  - Para Python!
- Interface que permite aplicações Python receber o *Environment* e responder com conteúdos do recurso.
  - Environment=contexto do pedido HTTP

# CherryPy

- Framework Web Minimalista (Python)
- Permite um desenvolvimento isolado
  - Sem recorrer a um servidor Web
- "Tão simples como:"

```
import cherrypy
class HelloWorld(object):
    @cherrypy.expose
    def index(self):
        return "Hello World!"

cherrypy.quickstart(HelloWorld())
```

# CherryPy

- Métodos são expostos no URL
- Toda a complexidade de sockets e HTTP é abstraída

```
import cherrypy
from datetime import datetime

class HelloWorld(object):
    @cherrypy.expose
    def index(self, name):
        return "Hello " + name

    @cherrypy.expose
    def now(self):
        return str(datetime.now())

cherrypy.quickstart(HelloWorld())
```

http://server**/?name=John**

http://server/**now**

# CherryPy

- Cabeçalhos HTTP do pedido são expostos à aplicação

```
class HelloWorld(object):
    @cherrypy.expose
    def index(self, name):
        addr=cherrypy.request.headers["Remote-Addr"]
        return "Hello " + name + " at " + addr
```

- Aplicação pode controlar cabeçalhos da resposta
  - Por exemplo, o tipo de dados devolvido

```
class HelloWorld(object):
    @cherrypy.expose
    def index(self, name):
        addr=cherrypy.request.headers["Remote-Addr"]
        cherrypy.response.headers["Content-Type"]="application/javascript"
        m = json.dumps({"message": "Hello " + name + " at " + addr})
        return m.encode("utf-8")
```

# Web Forms

- Web Form permite recolher informação no cliente a enviar para o servidor
  - Formulário web

- Pode conter diversos elementos gráficos tais como:
  - text fields, checkboxes, radio-buttons, submit buttons, select lists, textarea

- Ex: diálogo de login, envio de mensagem

# Web Forms (2)

- Todos elementos devem estar delimitados por um único <form></form>
  - Atributo essencial: **action** que deve conter o URL do recurso no servidor web que irá processar os dados enviados
  - Elemento essencial:
    - <input type="submit" value="Enviar">

# Web Forms

```
<form action="/login">

        <input type="text" name="user"></input>

        <input type="password" name="password"></input>

        <input type="submit" value="Enviar">

</form>
```

# Aplicação Web

- O servidor recebe os dados vindos do cliente e processa os mesmos.
- Pode gerar:
  - Página Web com conteúdos personalizados
  - Documento JSON
    - Que pode ser processado no Web Browser
  - Documento XML
    - Que pode ser processado por outra aplicação (caso normal de um Web Service)

# Cherrypy

- Campos do formulário são convertidos em argumentos da função

```
class HelloWorld(object):
    @cherrypy.expose
    def login(self, user, password):
        if user == "labi" and password == "batatinhas":
            return "Acesso concedido"
        else:
            return "Acesso Negado"
```

# Referências

- https://www.ietf.org/rfc/rfc2616.txt
- http://docs.cherrypy.org/en/latest/