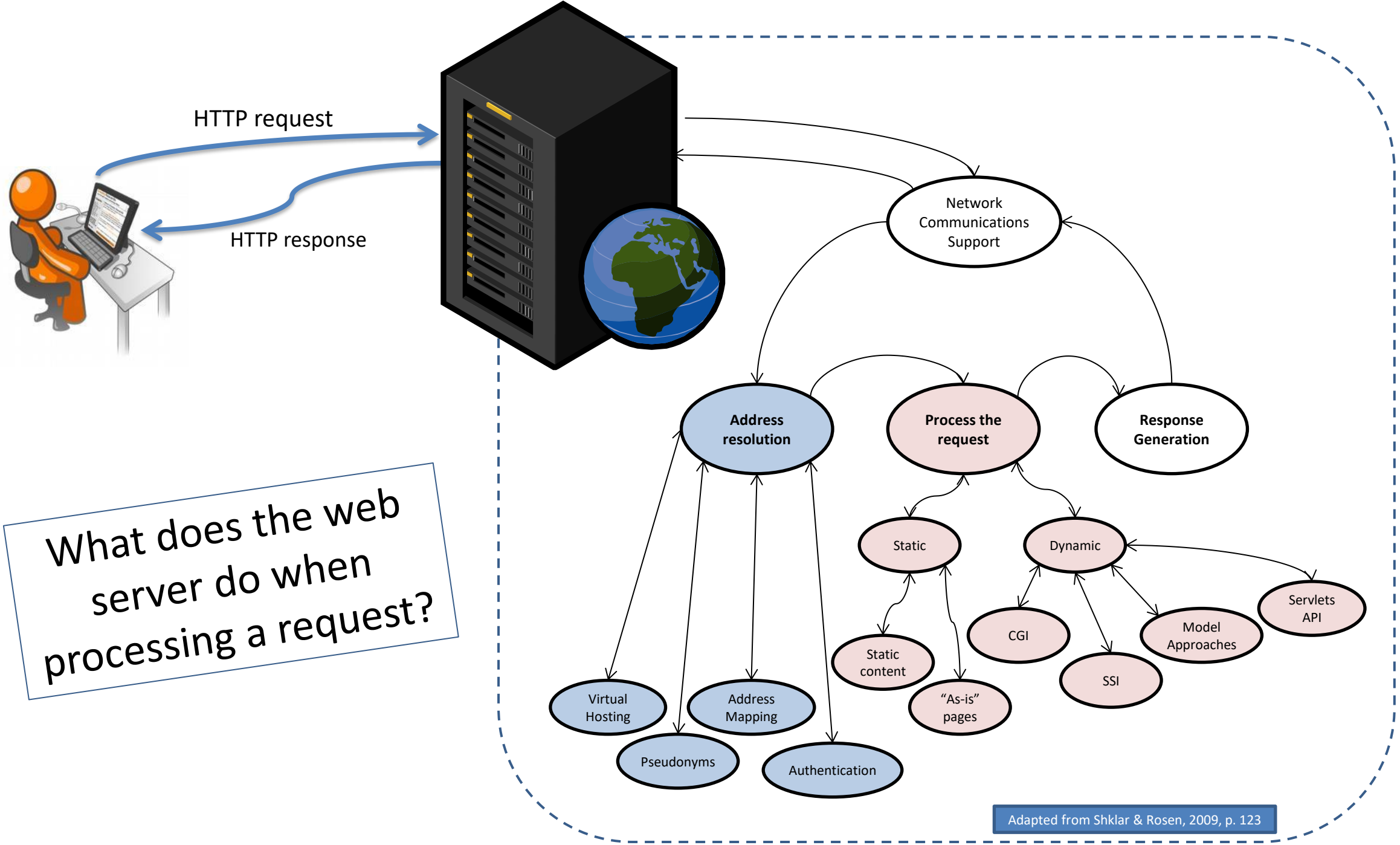
Address resolution in HTTP servers

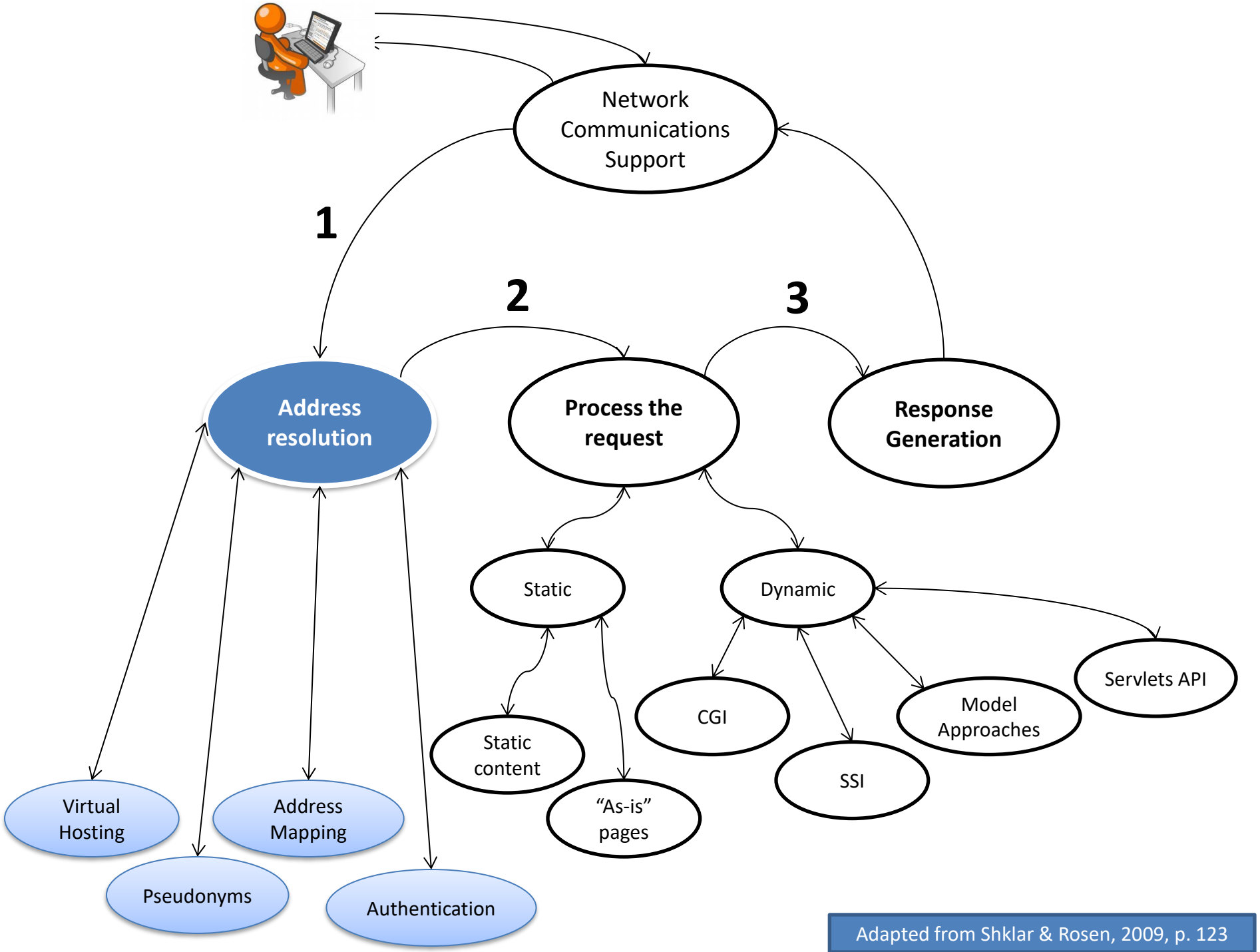
Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO



What does the web server do when processing a request?

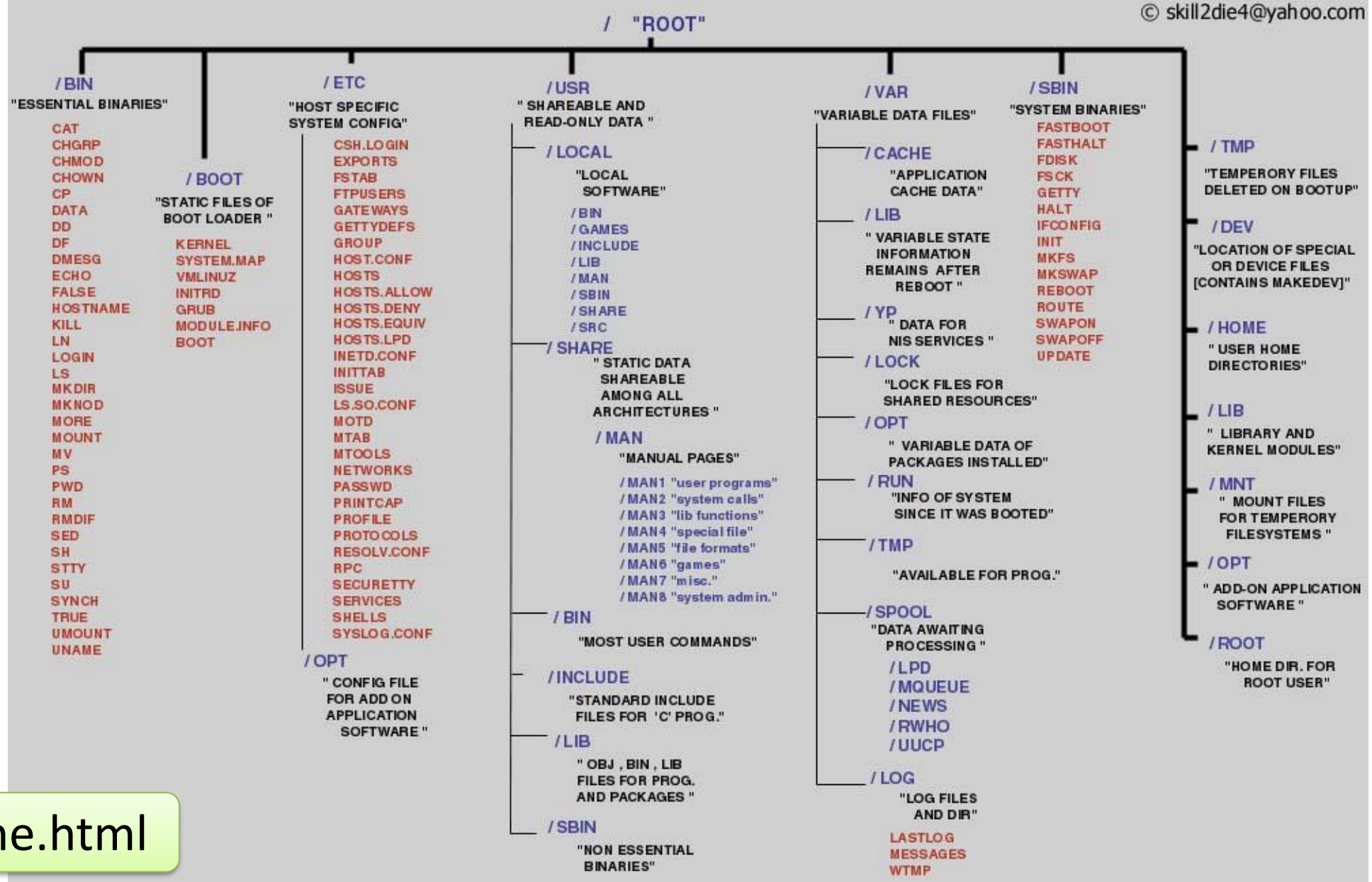
Adapted from Shklar & Rosen, 2009, p. 123



Adapted from Shklar & Rosen, 2009, p. 123

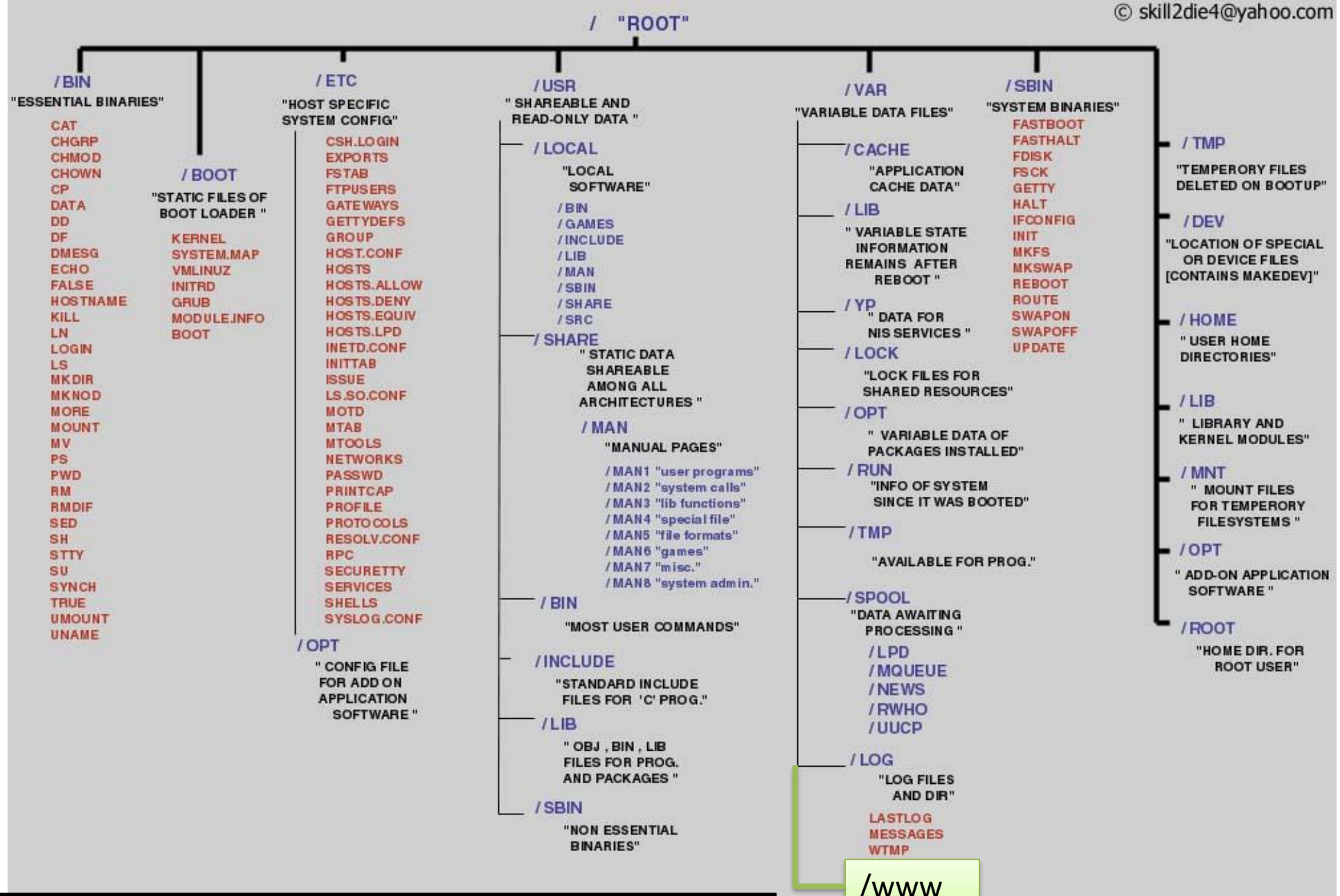
GET /tc/home.html HTTP/1.1
Host: www.tribunalconstitucional.pt
Date: Tue, 30 Sep 2008 13:45:29 GMT
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.3) Gecko/2008092417 Firefox/3.0.3
Referer: http://home.utad.pt/~lfb/teste.htm
If-Modified-Since: Tue, 30 Sep 2008 13:40:29 GMT

On the server disk, where is the file indicated by the address `/tc/home.html` **?**

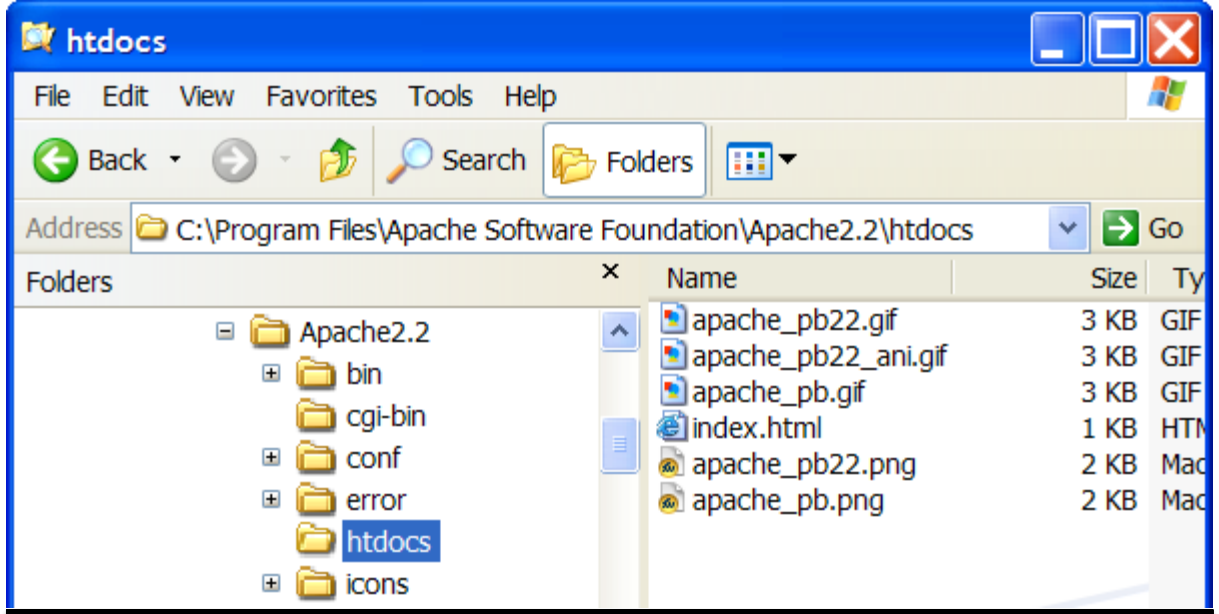


/tc/home.html

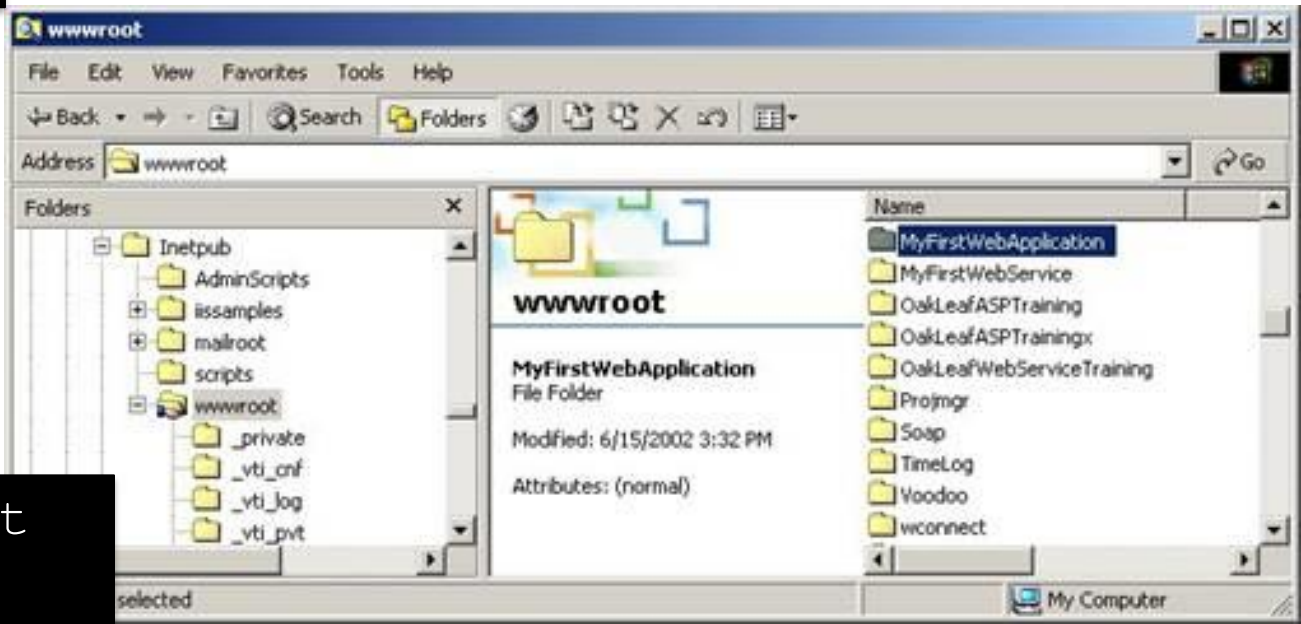
…make sense that the root of the web address matches any root of the local file system?



Apache's default web root on Linux, for example, is `/var/www`



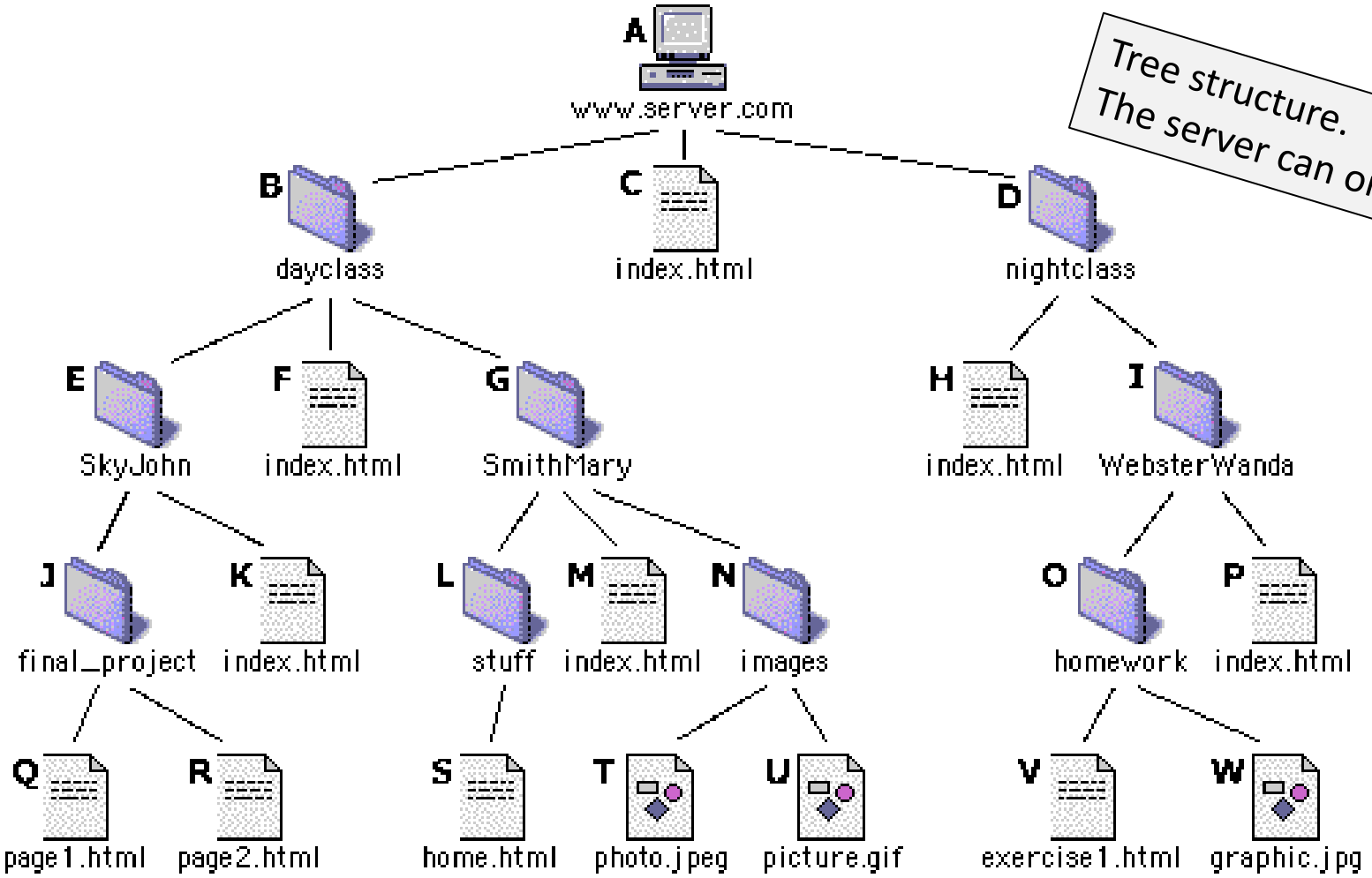
In Apache for Windows, it's `C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/`



In IIS (Windows), it is `c:\Inetpub\wwwroot`

… etc.

Website file structure example



Tree structure.
The server can only deliver files within this structure.

Network Communications Support

/tc/home.html

Address resolution

Process the request

Response Generation

GET /tc/home.html HTTP/1.1
Host: **www.tribunalconstitucional.pt**

Static

Dynamic

Static content

"As-is" pages

CGI

SSI

Model Approaches

Servlets API

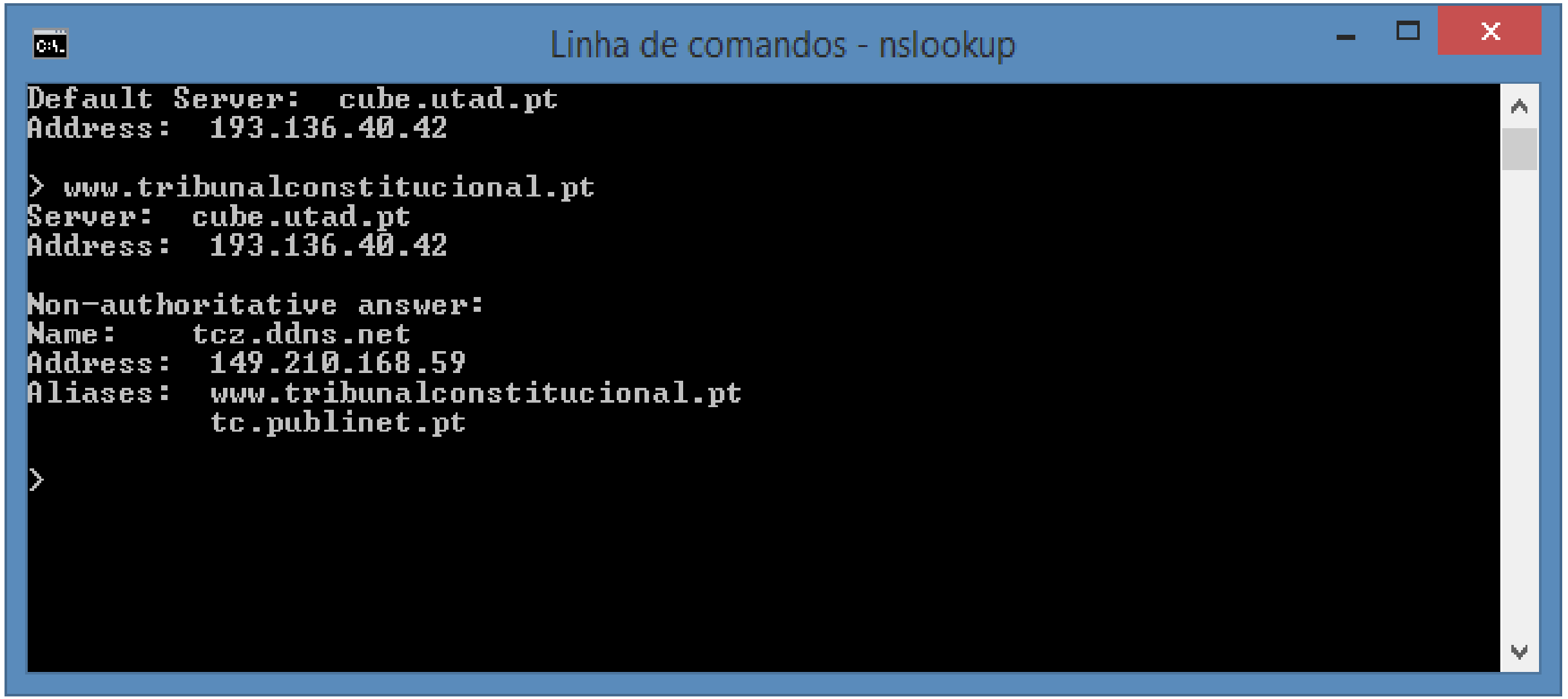
Virtual Hosting

Address Mapping

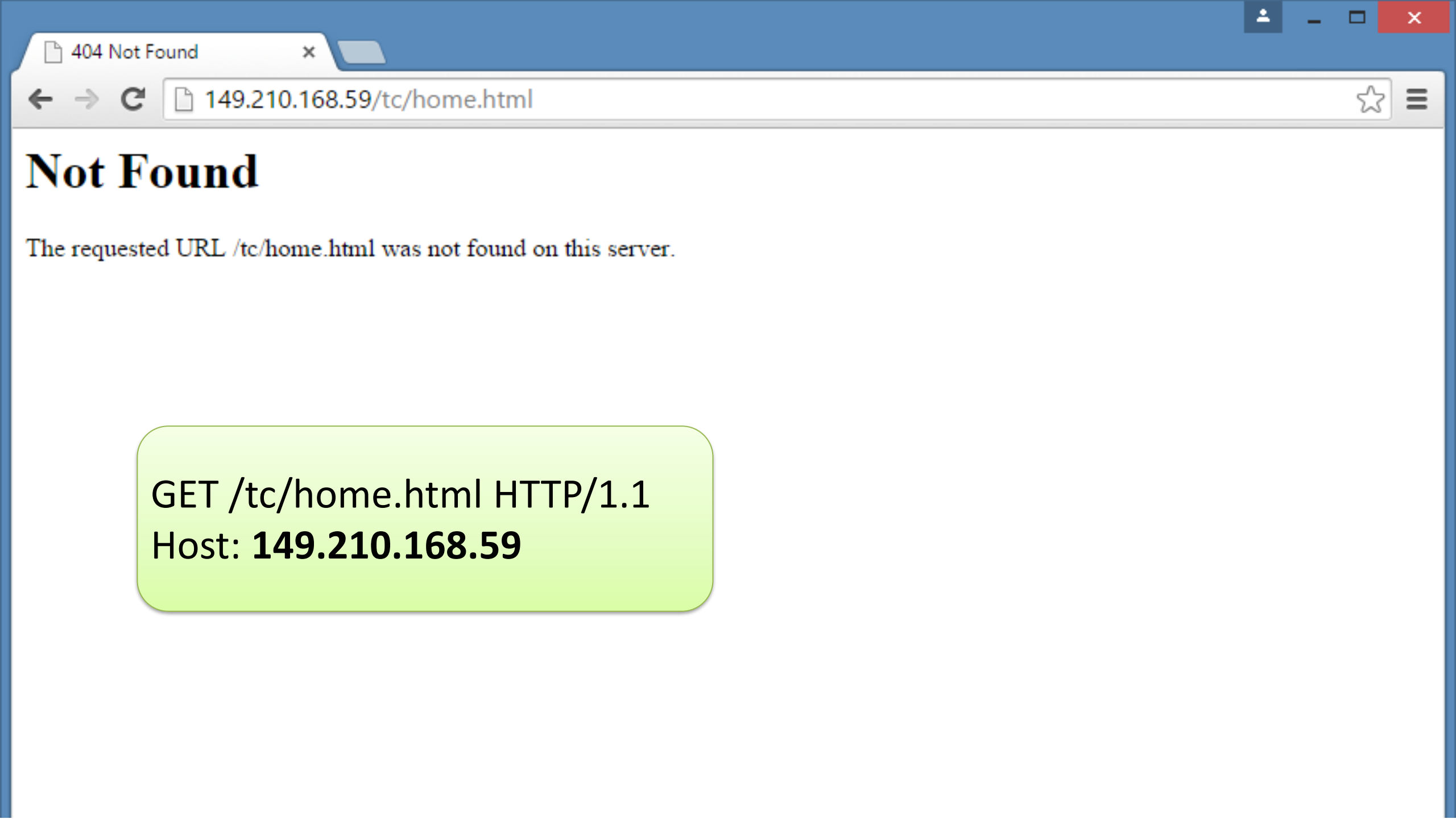
Pseudonyms

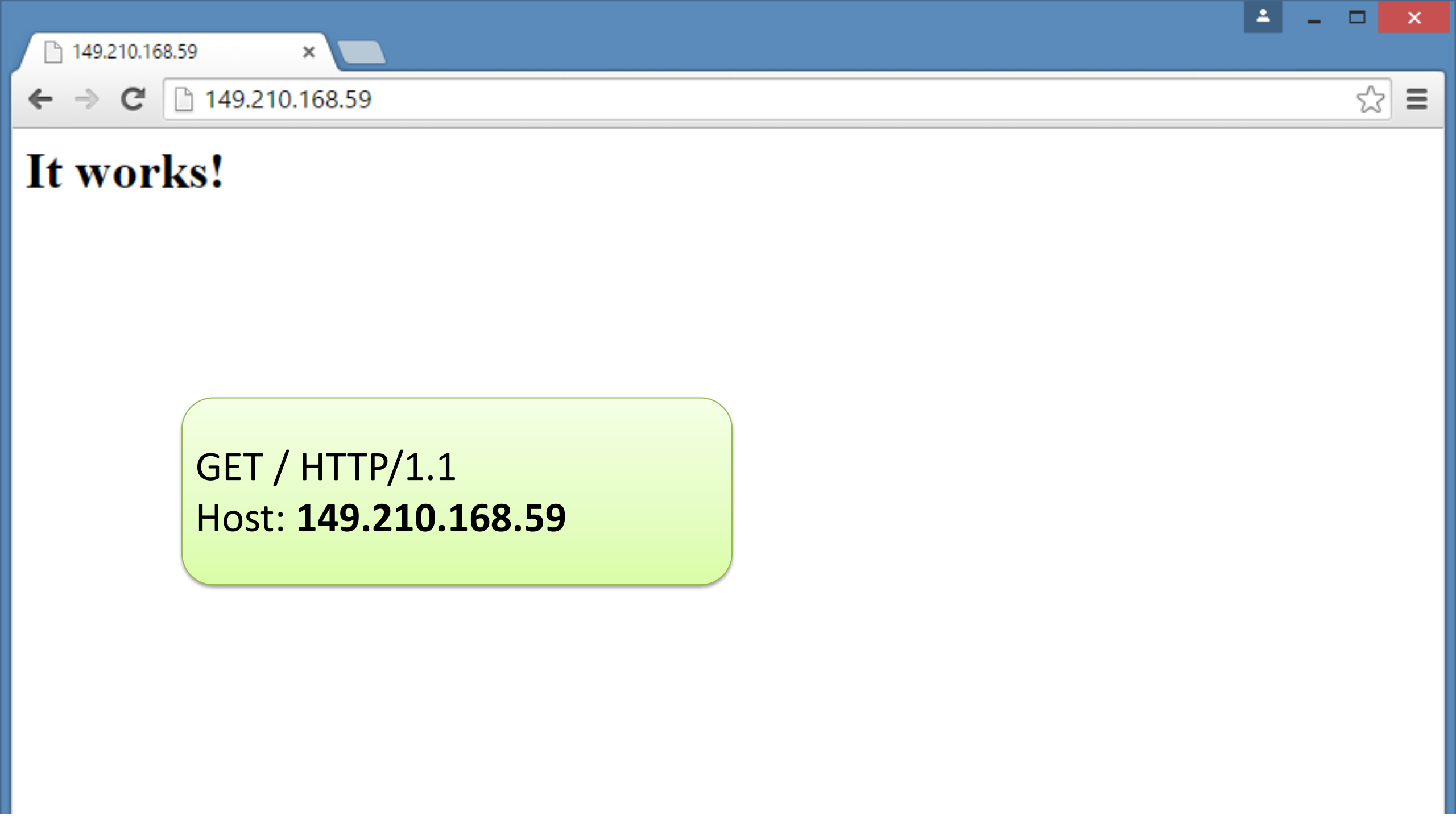
Authentication

Adapted from Shklar & Rosen, 2009, p. 123


Linha de comandos - nslookup
Default Server: cube.utad.pt
Address: 193.136.40.42
> www.tribunalconstitucional.pt
Server: cube.utad.pt
Address: 193.136.40.42
Non-authoritative answer:
Name: tcz.ddns.net
Address: 149.210.168.59
Aliases: www.tribunalconstitucional.pt
          tc.publinet.pt
>

TC > Página Inicial
www.tribunalconstitucional.pt/tc/home.html
TRIBUNAL CONSTITUCIONAL Portugal
JURISPRUDÊNCIA
BIBLIOTECA
INTERVENÇÕES
COMUNICADOS
ENGLISH VERSION
GET /tc/home.html HTTP/1.1
Host: www.tribunalconstitucional.pt
PÁGINA INICIAL
BEM-VINDOS
INSTRUMENTOS DE GESTÃO
CONTACTOS
LIGAÇÕES
INFORMAÇÃO LEGAL
Biblioteca do Tribunal Constitucional - Aviso
Informa-se que a Biblioteca do Tribunal Constitucional estará encerrada aos Leitores externos, por motivo de obras, a partir do dia 21 de setembro, por um período previsível de dois meses.
Audiência
O Presidente do Tribunal Constitucional, Conselheiro Joaquim de Sousa Ribeiro, recebeu em audiência, no dia 1 de


404 Not Found
149.210.168.59/tc/home.html
Not Found
The requested URL /tc/home.html was not found on this server.
GET /tc/home.html HTTP/1.1
Host: 149.210.168.59


149.210.168.59
149.210.168.59
It works!
GET / HTTP/1.1
Host: 149.210.168.59

Network Communications Support

Address resolution

Process the request

GET /tc/home.html HTTP/1.1
Host: **www.tribunalconstitucional.pt**

Virtual Hosting

Address Mapping

Pseudonyms

Authentication

Static

Static content

"As-is" pages

CGI

SSI

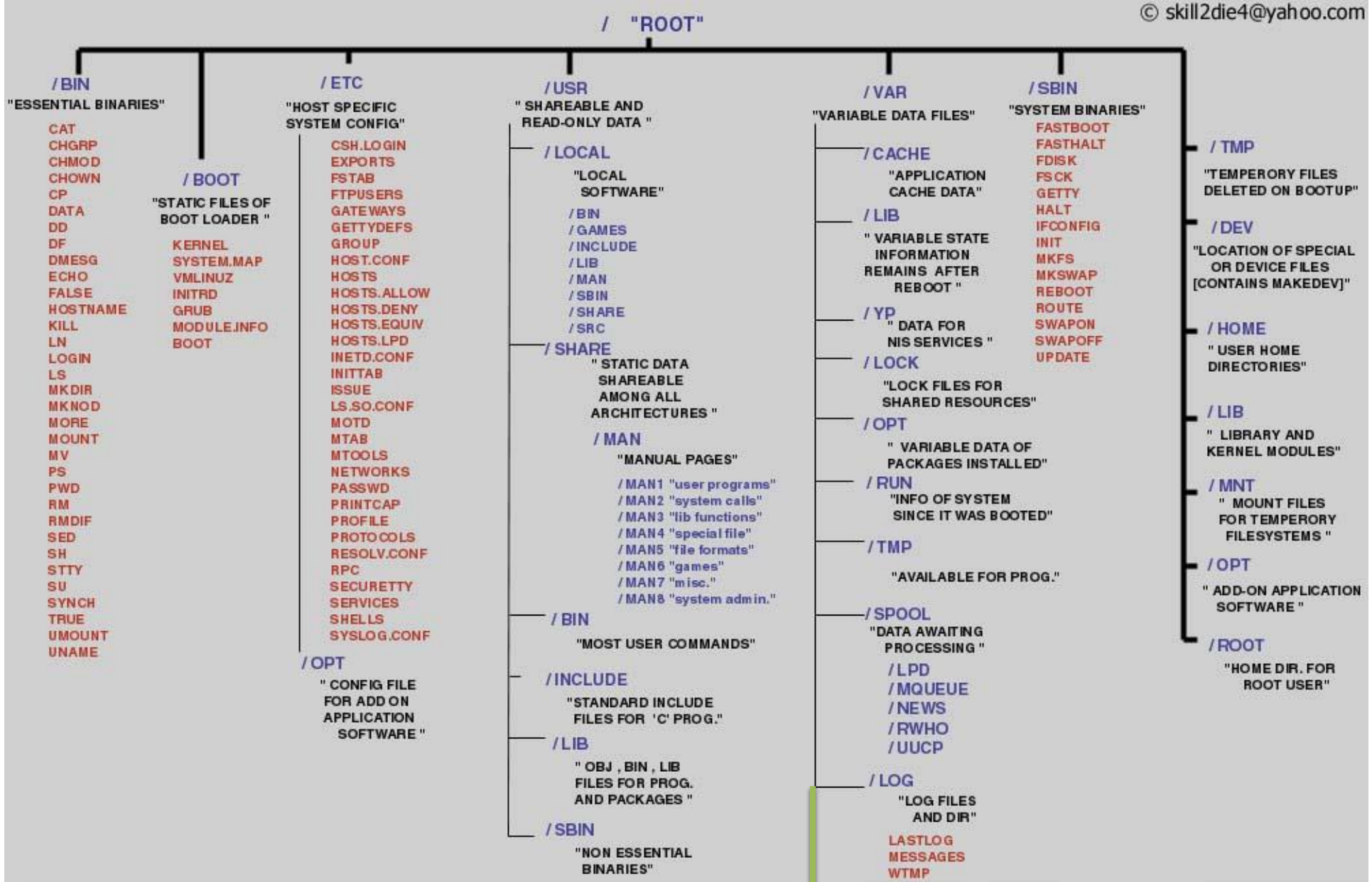
Model Approaches

Servlets API

Example in Apache server

```
<VirtualHost *:80>
    ServerName zf2-tutorial.localhost
    DocumentRoot /path/to/somewhere/public
    SetEnv APPLICATION_ENV "development"
    <Directory /path/to/somewhere/public>
        DirectoryIndex index.php
        AllowOverride
        All Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Adapted from Shklar & Rosen, 2009, p. 123



Is there somewhere a folder `tc`?
And inside it a file `home.html`?

/www /tc home.html

/tc home.html

Network Communications Support

Address resolution

Process the request

Response Generation

/tc/home.html

Virtual Hosting

Address Mapping

Pseudonyms

Authentication

Static

Static content

"As-is" pages

Dynamic

CGI

SSI

Model Approaches

Servlets API

Adapted from Shklar & Rosen, 2009, p. 123

Network Communications Support

Example in Apache server

```
<VirtualHost 172.20.30.40>
    DocumentRoot /www/example3
    ServerName www.example.net
    ServerAlias *.example.net
    Alias "/image" "/ftp/pub/image"
    # ...
</VirtualHost>
```

Address resolution

Process the request

Alias /tc/ /var/tc/

Alias /tc/home.html /site/index.asp?id=tribunal

Static

Dynamic

Servlets API

Static content

SSI

Virtual Hosting

Address Mapping

"As-is" pages

Pseudonyms

Authentication

Adapted from Shklar & Rosen, 2009, p. 123

Network Communications Support

Example in Apache server

```
<VirtualHost 172.20.30.40>
   DocumentRoot /www/example3
   ServerName www.example.net
   ServerAlias *.example.net
   # ...
</VirtualHost>
```

Address resolution

Process the request

GET /tc/home.html HTTP/1.1
Host: www.tribunalconstitucional.pt...

What is the file anyway? Is it static or dynamic?

Dynamic

CGI

Servlets API

Model Approaches

SSI

content

"As-is" pages

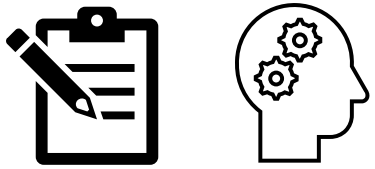
Virtual Hosting

Address Mapping

Pseudonyms

Authentication

Adapted from Shklar & Rosen, 2009, p. 123



- Pseudonyms can allow server to "attach" content to the virtual hosting's file tree that de would not normally be able to access.

- It even allow sharing files between virtual hostings.

Network Communications Support

GET /tc/home.html HTTP/1.1
Host: www.tribunalconstitucional.pt...

Is it necessary a header `Authorization`?

Status code 401
Header WWW-Authenticate: : Basic realm="User Visible Realm"
Header Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l

Servlets API
Model Approaches
CGI
SSI
Static content
"As-is" pages
Virtual Hosting
Address Mapping
Pseudonyms
Authentication

Adapted from Shklar & Rosen, 2009, p. 123

Network Communications Support

Address resolution

Process the request

Response Generation

Resource (file) identified.
Authentication verified.

Virtual Hosting

Address Mapping

Pseudonyms

Authentication

Static content

"As-is" pages

CGI

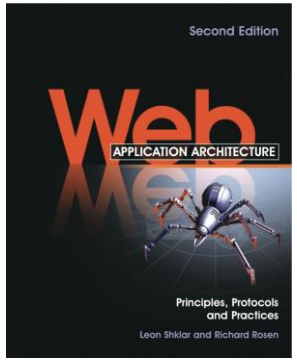
SSI

Model Approaches

Servlets API

Adapted from Shklar & Rosen, 2009, p. 123

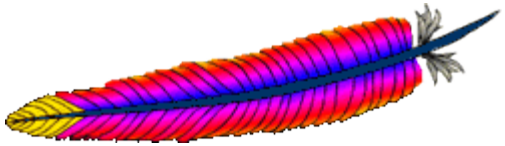
Bibliography



Shklar, Leon & Rosen, Rich (2009). *Web Application Architecture: Principles, Protocols and Pratices*. Chichester, Reino Unido: John Wiley & Sons.

Address Processing: pages 123-125.
Virtual Hosting: pages 57-58 and 140-141.
Authentication: pages 51-53.

About pseudonyms / aliases:



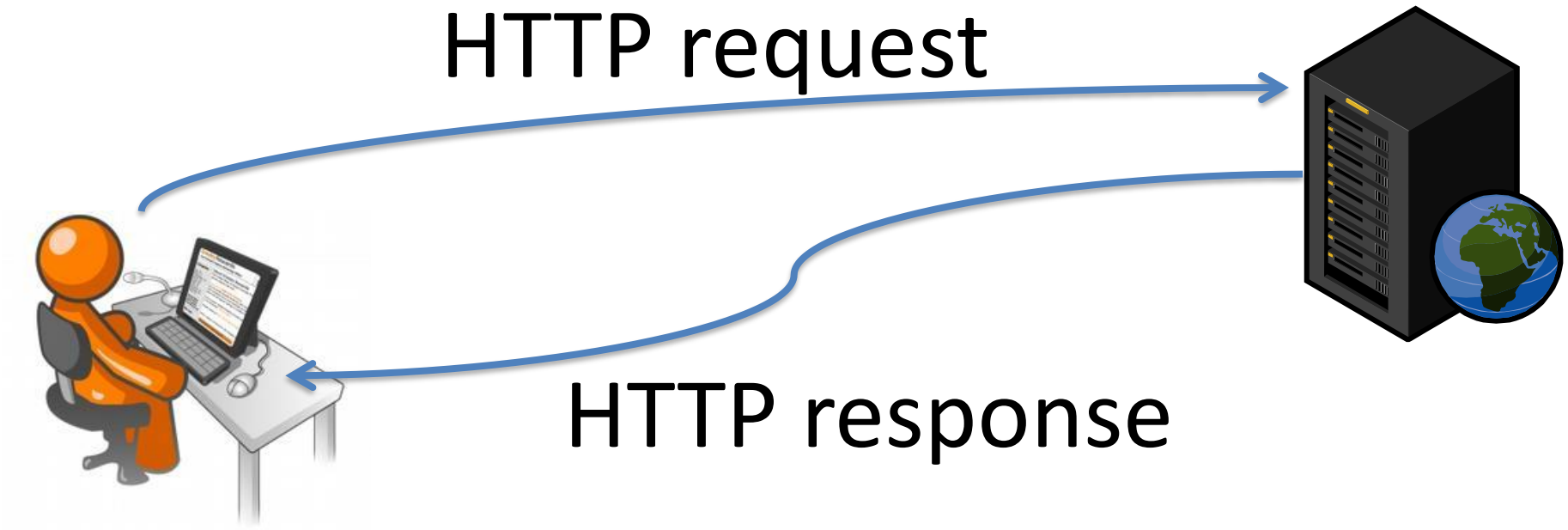
mod_alias - Apache HTTP Server
http://httpd.apache.org/docs/current/mod/mod_alias.html

About Address Mapping:
http://httpd.apache.org/docs/current/urlmapping.html

Static and dynamic content delivery

Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO


HTTP request
HTTP response

Network Communications Support

Address resolution

Process the request

Response Generation

Static

Dynamic

Static content

"As-is" pages

CGI

SSI

Model Approaches

Servlets API

Virtual Hosting

Address Mapping

Pseudonyms

Authentication

Adapted from Shklar & Rosen, 2009, p. 123

Static Content Delivery

- Static content pages
  - Combines the generation of the required response header with the content of a hosted file as the message body.
- “As-is” pages
  - Uses the full contents of a file as the entire HTTP message used as a reply.

Dynamic content delivery

- CGI – Common Gateway Interface
- SSI – Server Side Includes
- More advanced mechanisms
  - Native APIs
  - FastCGI
  - Template Processing
  - Servlets
  - Java Server Pages

Dynamic content delivery

- CGI – Common Gateway Interface

Table 4.1 Environment variables set from sources of information other then HTTP headers

SERVER_PROTOCOL	HTTP version as defined on the request line following HTTP method and URL.
SERVER_PORT	Server port used for submitting the request, set by the server based on the connection parameters.
REQUEST_METHOD	HTTP method as defined on the request line.
PATH_INFO	Extra path information in the URL. For example, if the URL is `http://mysite.org/cgi-bin/zip.cgi/test.html`, and `http://mysite.org/cgi-bin/zip.cgi` is the location of a CGI script, then `/test.html` is the extra path information.
PATH_TRANSLATED	Physical location of the CGI script on the server. In our example, it would be `/www/cgi-bin/zip.cgi` assuming that the server is configured to map the `/cgi-bin` path to the `/www/cgi-bin` directory.
SCRIPT_NAME	Set to the path portion of the URL, excluding the extra path information. In the same example, it's `/cgi-bin/zip.cgi`
QUERY_STRING	Information that follows the '?' in the URL.

Dynamic content delivery

- CGI – Common Gateway Interface

```
<FORM action = "/cgi-bin/hello_post.cgi" method = "POST">
First Name: <input type = "text" name = "first_name">  <br>

Last Name: <input type = "text" name = "last_name">

<input type = "submit" value = "Submit">
</FORM>
```

Steps:

- Determines if it is a CGI program.
- Translates to the corresponding file
- Checks if it is a valid file
- Check permissions
- Set environment variables
- Create child process to run CGI program
- Process CGI Program Response

Dynamic content delivery

- CGI – Common Gateway Interface

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST") {
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
} else {
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$first_name = $FORM{first_name};
$last_name  = $FORM{last_name};
```

Example (perl)
Process form data from stdin…

Dynamic content delivery

- CGI – Common Gateway Interface

```
$first_name = $FORM{first_name};
$last_name  = $FORM{last_name};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Hello - Second CGI Program</title>";
print "</head>";
print "<body>";
print "<h2>Hello $first_name $last_name - Second CGI Program</h2>";
print "</body>";
print "</html>";

1;
```

Example (perl)
…and produces the information in a HTML structure

Dynamic content delivery

- SSI – Server Side Includes
  - Mechanisms for including helper files in an HTML page
  - May include results of running scripts
  - Defined through macros

```
<!--#command attr1="value1" attr2="value2" -->
```

Dynamic content delivery

- SSI – Server Side Includes

Most common directives

Directive	Parameters			Example
include	file or virtual	This is probably the most used SSI page, text file, script, etc.) to be inc the file or execute the script, the inc the current file. When using "file" it i Apache documentation recommends	ameters specify the file (HTML does not have access to read ath relative to the directory of explicitly configured. The	`<!--#include virtual="menu.cgi" -->` or `<!--#include file="footer.html" -->` Apache tutorial on SSI stipulates the format requires a space character before the "-->" that closes the element.
exec	cgi or cmd	This directive executes a program, s path to a CGI script. The PATH_INF instead of "include virtual".	he cgi parameter specifies the exec cgi" should be used	`<!--#exec cgi="/cgi-bin/foo.cgi" -->` or `<!--#exec cmd="ls -l" -->`
echo	var	This directive displays the contents	IFIED, and HTTP_ACCEPT.	`<!--#echo var="REMOTE_ADDR" -->`
config	timefmt, sizefmt, or errmsg	This directive configures the display		`<!--#config timefmt="%y %m %d" -->` or `<!--#config sizefmt="bytes" -->` or `<!--#config errmsg="SSI command failed!" -->`
flastmod or fsize	file or virtual	These directives display the date wh document to use. The file parameter document root.	al parameters specify the ment as relative to the	`<!--#flastmod virtual="index.html" -->` or `<!--#fsize file="script.pl" -->`
printenv		This directive outputs a list of all var		`<!--#printenv -->`

Dynamic content delivery

- SSI – Server Side Includes

Control directives

Directive	Parameters		Example
if	expr	Used for condition tests that may dete n one single physical page.	<!--#if expr="${Sec_Nav}" --> <!--#include virtual="" --> <!--#endif -->
elif	expr	Serves the same purpose as further co	<!--#if expr="${Sec_Nav}" --> <!--#include virtual="secondary_nav.txt" --> <!--#elif expr="${Pri_Nav}" --> <!--#include virtual="primary_nav.txt" --> <!--#endif -->
else		If none of the if and elif directive catche happen.	<!--#if expr="${Sec_Nav}" --> <!--#include virtual="secondary_nav.txt" --> <!--#else --> <!--#include virtual="article.txt" --> <!--#endif -->
endif			See above for example.
set	var, value	Sets the value of a SSI variable. (Not s	<!--#set var="foo" value="bar" -->

Dynamic content delivery

- Native APIs
  - Uses compiled code optimized for use in context of a specific web server environment
    - Apache Server API
    - ISAPI (Microsoft IIS)
- FastCGI
  - Reuse CGI execution processes

Dynamic content delivery

- Template processing
  - PHP
  - Cold Fusion
  - Active Server Pages
  - …

```
<CFQUERY NAME="query1" DATASOURCE="oracle" ...>
   SELECT id, columnX, columnY, columnZ
     FROM TABLE1
    WHERE id = #substitution-parameter#
</CFQUERY>

<CFIF query1.recordcount GT 0>
   <TABLE>
      <CFOUTPUT QUERY="query1">
         <TR>
            <TD>#columnX#</TD>
            <TD>#columnY#</TD>
            <TD>#columnZ#</TD>
         </TR>
      </CFOUTPUT>
   </TABLE>
</CFIF>
```

Figure 4.11 Sample template (Cold Fusion)

Dynamic content delivery

- Servlets

Java classes used to extend the functionality of a server.

(requires a container - eg Apache Tomcat)

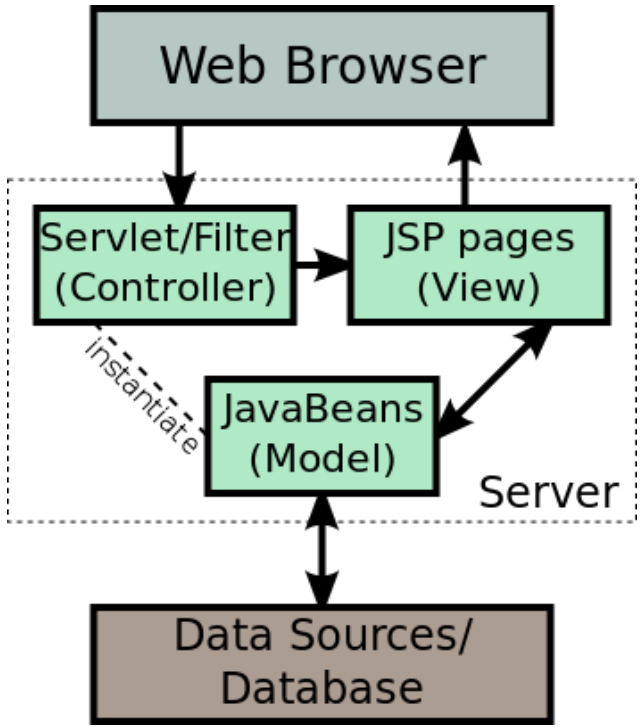
Server Extensions

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FormServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>\n<head><title>hello</title></head>");
        out.println("<body>");

        Enumeration e = request.getParameterNames();
        while (e.hasMoreElements()) {
            String name = (String)e.nextElement();
            String value = request.getParameter(name);
            out.println("<h3>" + name + ": " + value + "</h3>");
        }
        out.println("</body>\n</html>");
    }

    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException
    {
        doGet(request, response);
    }
}
```

Figure 4.12 Parameter processing in Servlets

Dynamic content delivery

- Java Server Pages

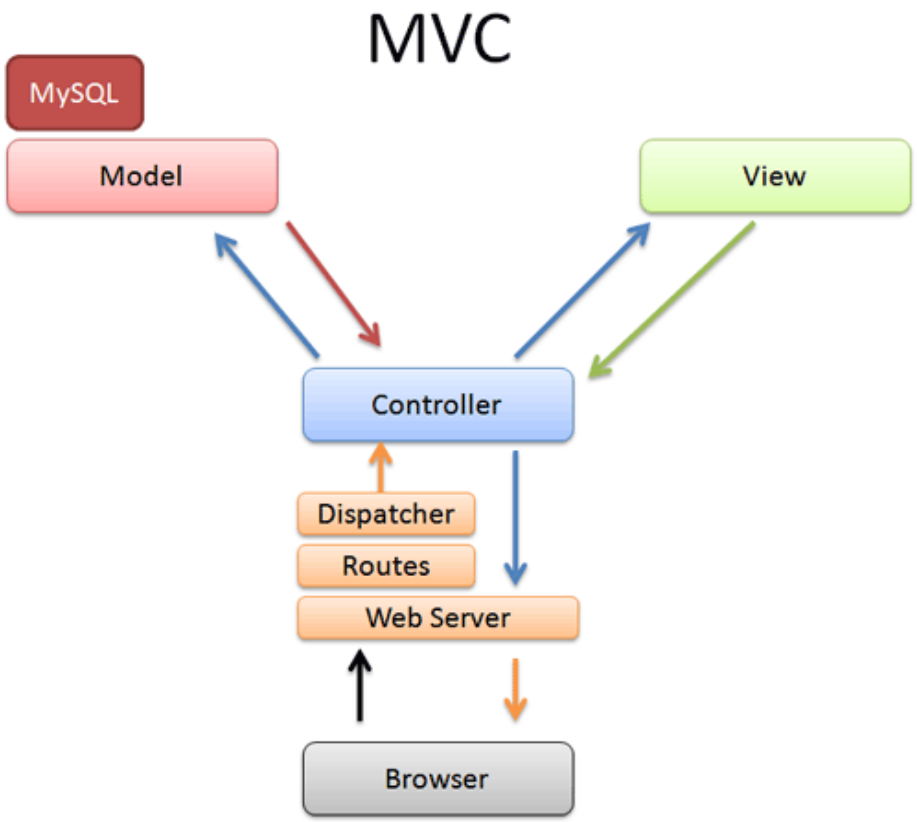


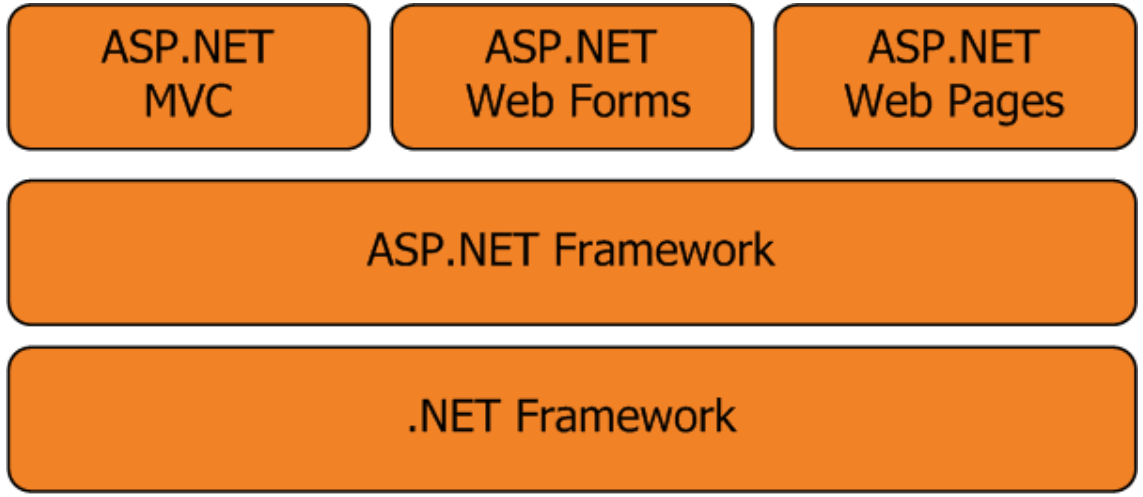

```
<html>
<head><title>hello!</title></head>
<body>
<%@ page import ="java.util.*" %>
<% Enumeration e = request.getParameterNames();
   while (e.hasMoreElements()) {
      String name  = (String)e.nextElement();
      String value = request.getParameter(name);
%>
<h3><%=name%>:<%=value%></h3>
<% } %>
</body>
</html>
```

Figure 4.13 Parameter processing in JSP

Dynamic content delivery

- Future directions?

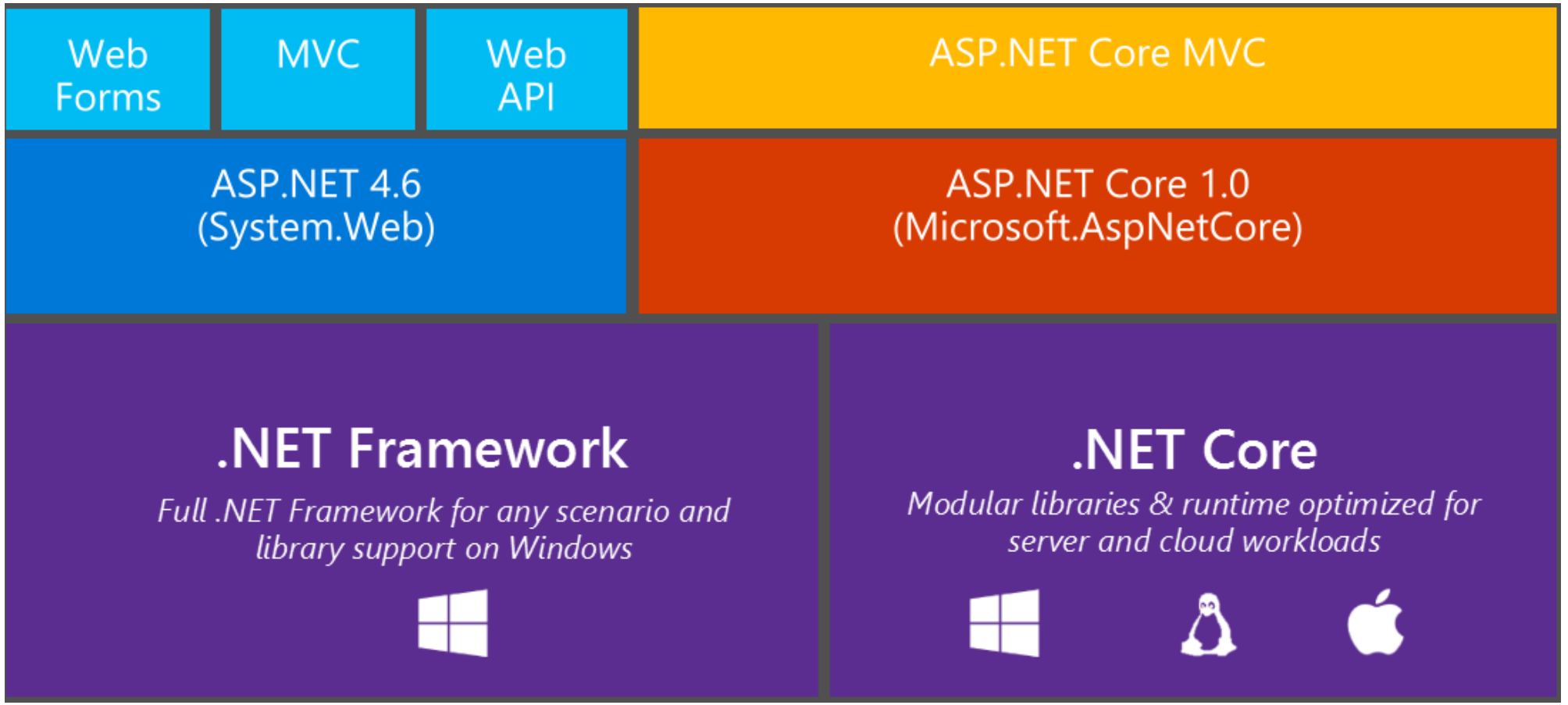




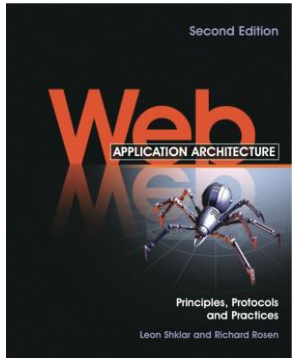
ASP.NET Frameworks

Dynamic content delivery

- Future directions?

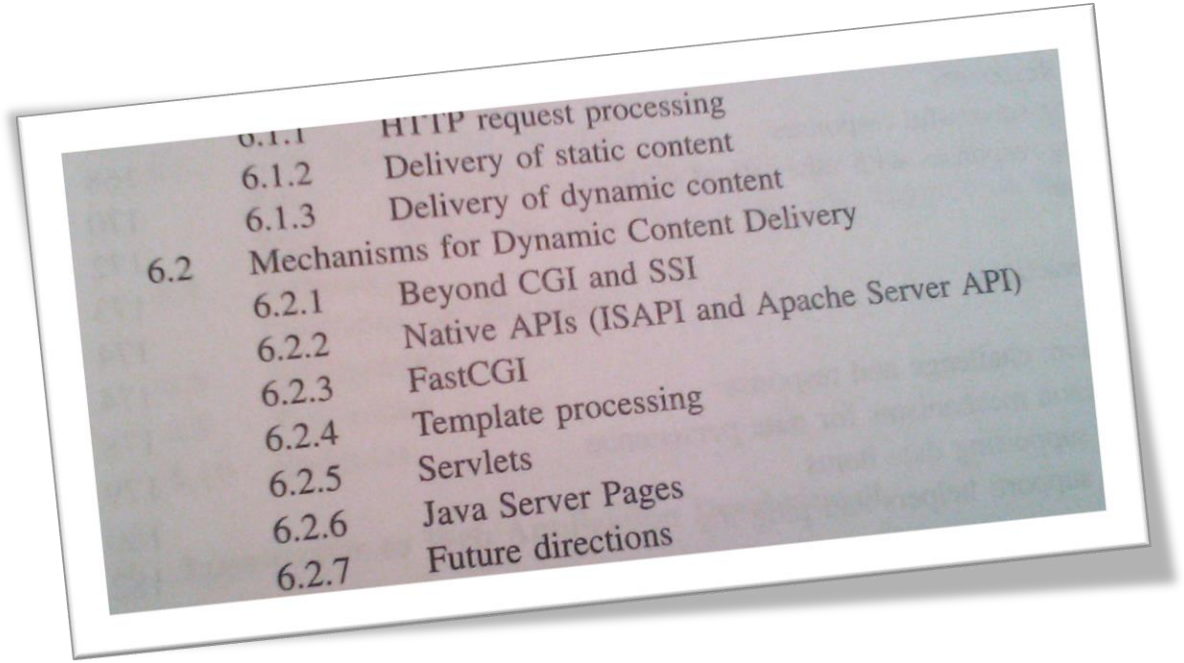


Bibliography

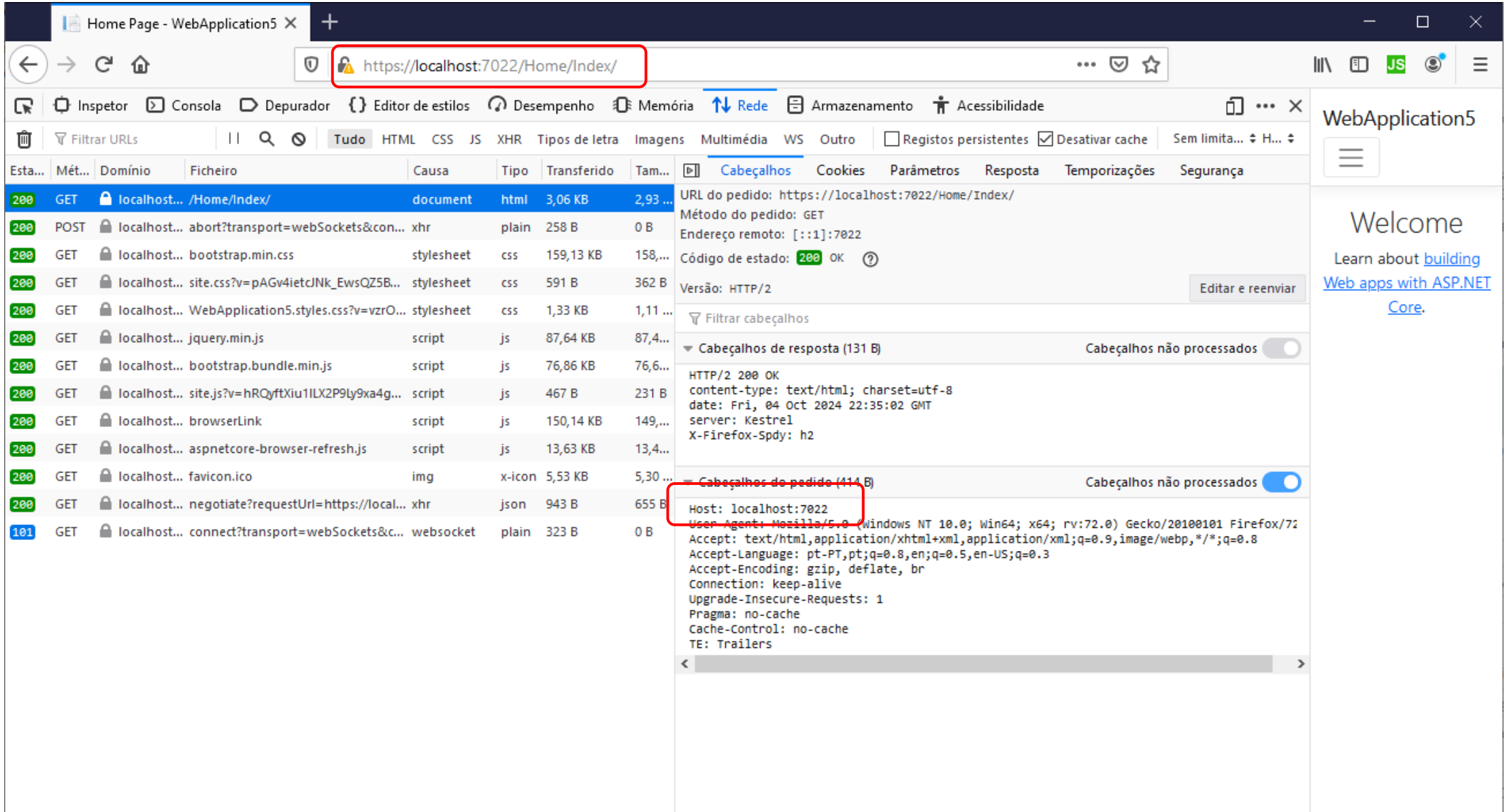


Shklar, Leon & Rosen, Rich (2009). *Web Application Architecture: Principles, Protocols and Pratices*. Chichester, Reino Unido: John Wiley & Sons.

Pages: 123 to 139



Routing


Home Page - WebApplication5
https://localhost:7022/Home/Index/
Inspetor
Consola
Depurador
Editor de estilos
Desempenho
Memória
Rede
Armazenamento
Acessibilidade
Filtrar URLs
Tudo
HTML
CSS
JS
XHR
Tipos de letra
Imagens
Multimédia
WS
Outro
Registos persistentes
Desativar cache
Sem limita...
H...
Esta...
Mét...
Domínio
Ficheiro
Causa
Tipo
Transferido
Tam...
200 GET localhost... /Home/Index/ document html 3,06 KB 2,93 ...
200 POST localhost... abort?transport=webSockets&con... xhr plain 258 B 0 B
200 GET localhost... bootstrap.min.css stylesheet css 159,13 KB 158,...
200 GET localhost... site.css?v=pAGv4ietcJNk_EwsQZ5B... stylesheet css 591 B 362 B
200 GET localhost... WebApplication5.styles.css?v=vzrO... stylesheet css 1,33 KB 1,11 ...
200 GET localhost... jquery.min.js script js 87,64 KB 87,4...
200 GET localhost... bootstrap.bundle.min.js script js 76,86 KB 76,6...
200 GET localhost... site.js?v=hRQyftXiu1ILX2P9Ly9xa4g... script js 467 B 231 B
200 GET localhost... browserLink script js 150,14 KB 149,...
200 GET localhost... aspnetcore-browser-refresh.js script js 13,63 KB 13,4...
200 GET localhost... favicon.ico img x-icon 5,53 KB 5,30 ...
200 GET localhost... negotiate?requestUrl=https://local... xhr json 943 B 655 B
101 GET localhost... connect?transport=webSockets&c... websocket plain 323 B 0 B
Cabeçalhos
Cookies
Parâmetros
Resposta
Temporizações
Segurança
URL do pedido: https://localhost:7022/Home/Index/
Método do pedido: GET
Endereço remoto: [::1]:7022
Código de estado: 200 OK
Versão: HTTP/2
Editar e reenviar
Filtrar cabeçalhos
Cabeçalhos de resposta (131 B)
Cabeçalhos não processados
HTTP/2 200 OK
content-type: text/html; charset=utf-8
date: Fri, 04 Oct 2024 22:35:02 GMT
server: Kestrel
X-Firefox-Spdy: h2
Cabeçalhos do pedido (414 B)
Cabeçalhos não processados
Host: localhost:7022
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pt-PT,pt;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
TE: Trailers
WebApplication5
Welcome
Learn about building Web apps with ASP.NET Core.

ASP.NET MVC - Routing example

```
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Examples:

https://localhost/Home/Index

https://localhost/Persons/Create

https://localhost/Persons/Edit/4

https://localhost/

More Examples:

https://localhost/Home/Index/All

https://localhost/Home/Index?whom=All

https://localhost/Persons/Edit?id=4

X https://localhost/Home/Index/All/Paged

Resume

ADDRESS RESOLUTION IN HTTP SERVERS

HOW IT WORKS?
WHAT DO THE STEPS MEAN: VIRTUAL HOSTING, PSEUDONYMS, ADDRESS MAPPING, AUTHENTICATION
HOW SERVER IDENTIFY THE FILES TO PRODUCE THE RESPONSE? (LOCATION AND NAME)
HOW CAN THE SERVER USE DIFFERENT TECHNOLOGIES TO GENERATE DYNAMIC CONTENT?

STATIC AND DYNAMIC CONTENT DELIVERY

WHAT IS THE DIFFERENCE BETWEEN DELIVERING STATIC OR DYNAMICALLY GENERATED CONTENT?
WHAT ARE THE MOST COMMON TECHNOLOGIES FOR GENERATING DYNAMIC CONTENT?
WHAT IS THE DIFFERENCE BETWEEN DYNAMICALLY GENERATED AND STATIC CONTENT?
WHAT INTERFERENCE DO SERVER TECHNOLOGIES HAVE ON THE BEHAVIOR OF WEB BROWSERS?

Readings through October 14th Class

7 Web Browsers

7.1 Overview of Browser Functionality

7.2 Architectural Considerations

7.3 Overview of Processing Flow in a Browser

7.3.1 Transmitting a request

7.3.2 Receiving a response

7.4 Processing HTTP Requests

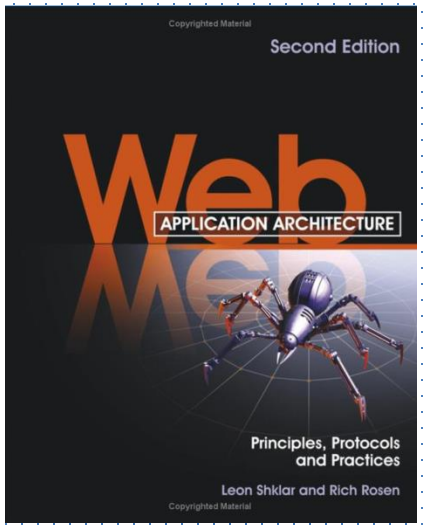
7.4.1 Constructing the request line

7.4.2 Constructing the headers

7.4.3 Constructing the request body

7.4.4 Transmitting the request

7.5 Processing HTTP Responses

7.5.1 Processing successful responses

7.5.2 Processing responses with other status codes