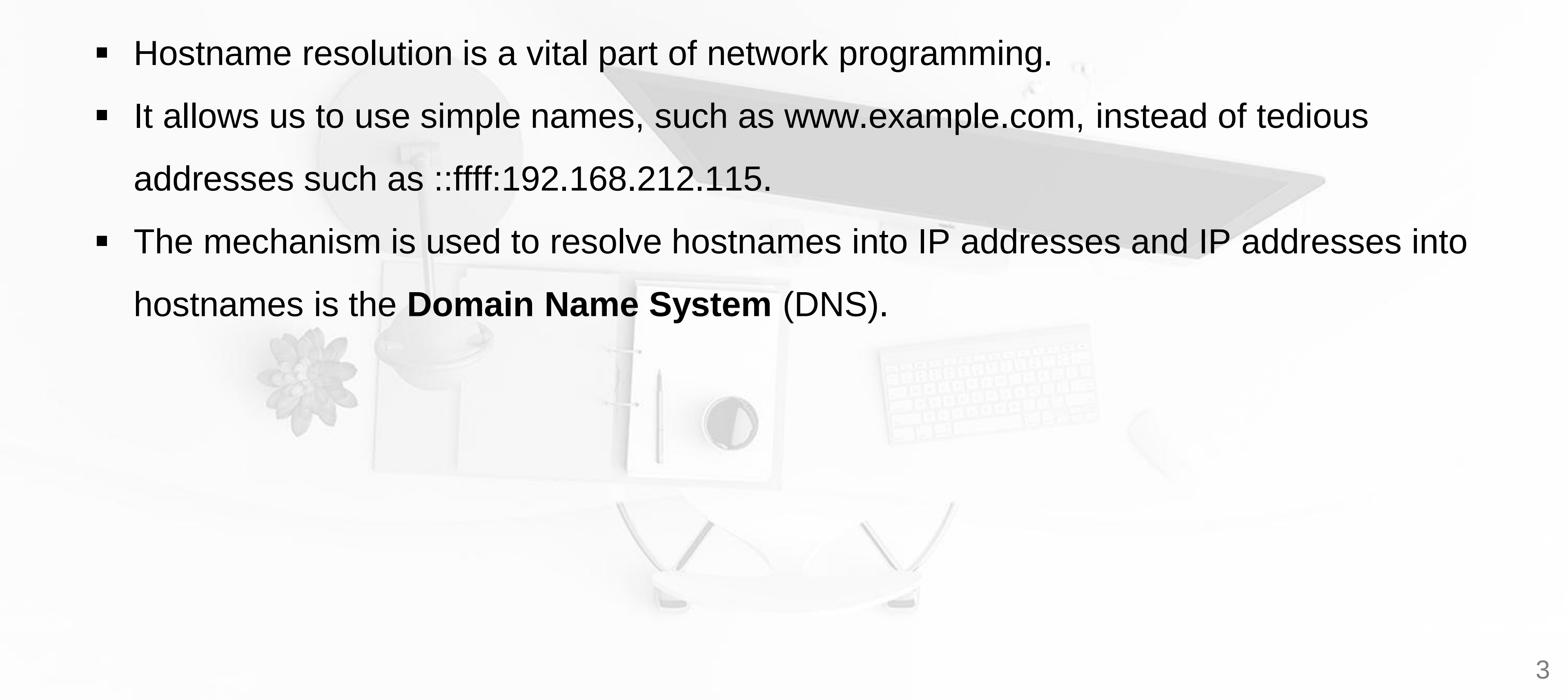
Network Programming

Ung Văn Giàu
Email: giau.ung@eiu.edu.vn

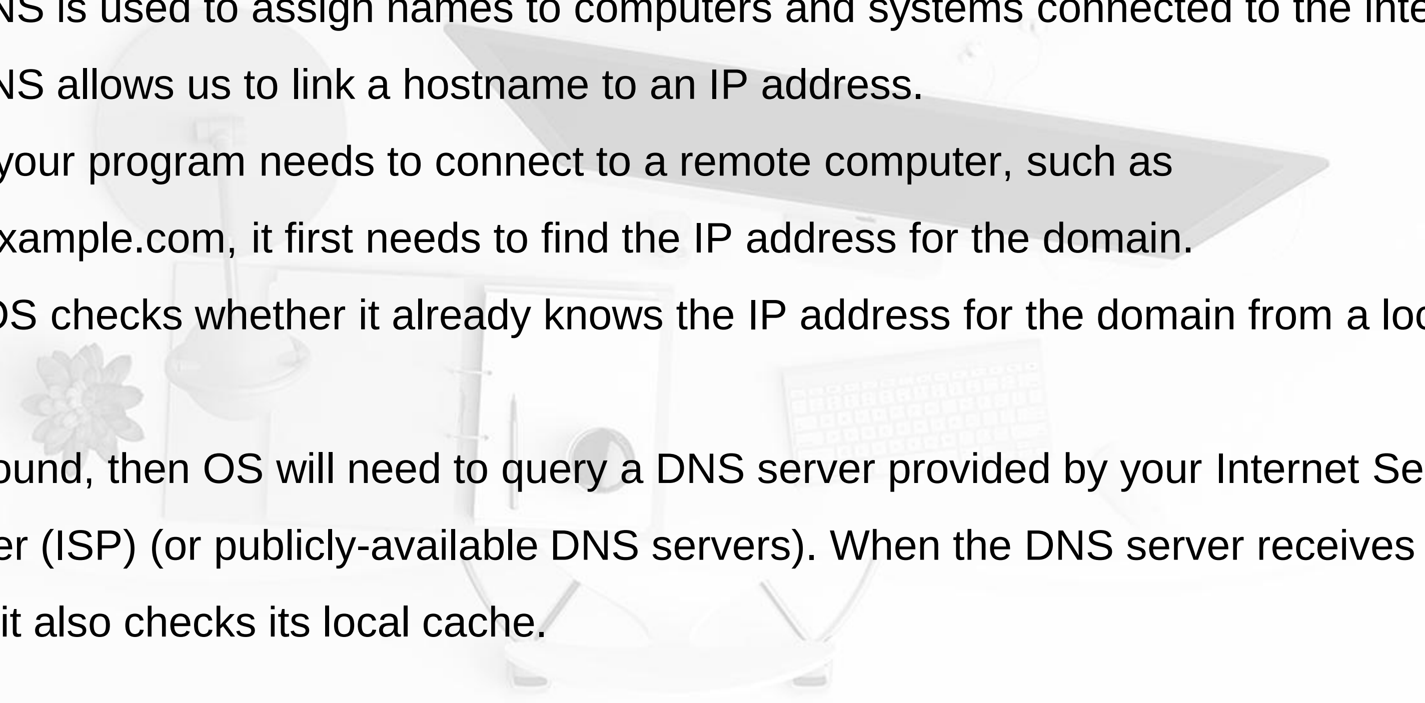
Hostname Resolution and DNS

1. Introduction

- Hostname resolution is a vital part of network programming.
- It allows us to use simple names, such as www.example.com, instead of tedious addresses such as ::ffff:192.168.212.115.
- The mechanism is used to resolve hostnames into IP addresses and IP addresses into hostnames is the **Domain Name System** (DNS).



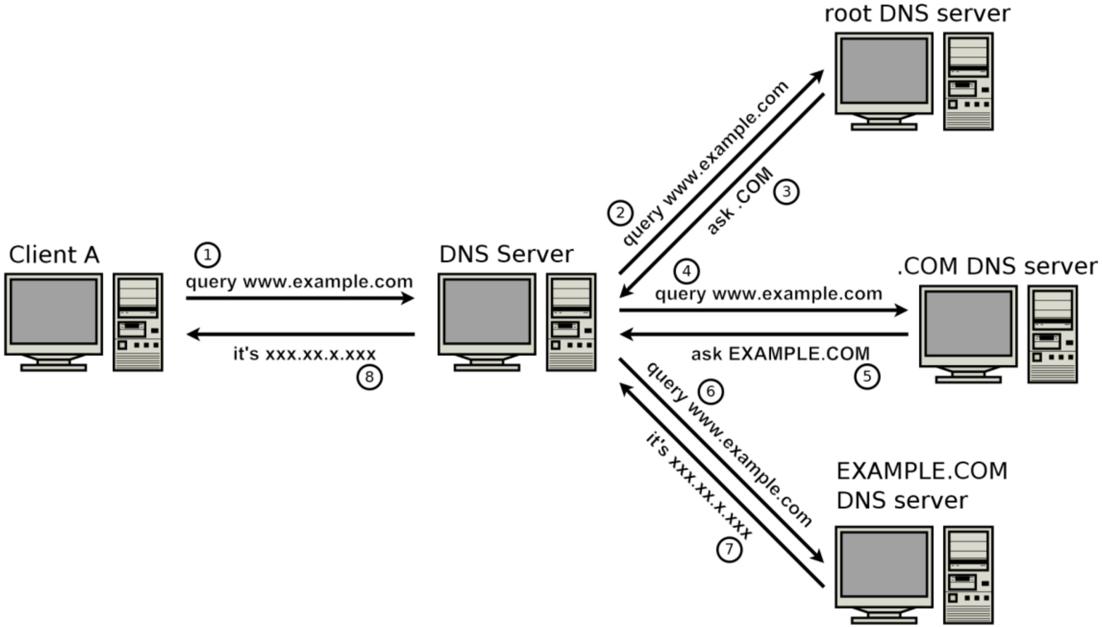
2. How hostname resolution works



- The DNS is used to assign names to computers and systems connected to the internet. The DNS allows us to link a hostname to an IP address.
- When your program needs to connect to a remote computer, such as www.example.com, it first needs to find the IP address for the domain.
- First, OS checks whether it already knows the IP address for the domain from a local cache.
- If not found, then OS will need to query a DNS server provided by your Internet Service Provider (ISP) (or publicly-available DNS servers). When the DNS server receives a query, it also checks its local cache.

2. How hostname resolution works

- If the DNS server doesn’t have the requested DNS record in its cache, then it needs to query other DNS servers until it connects directly to the DNS server responsible for the target system.

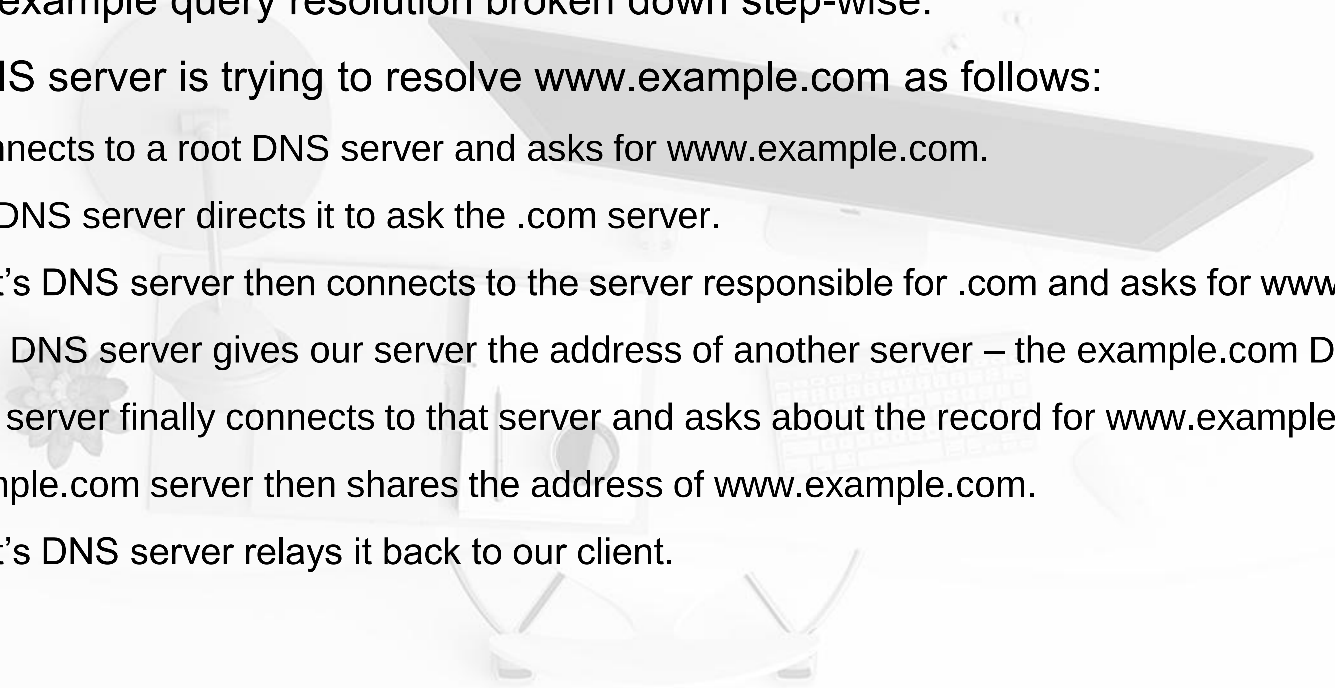


2. How hostname resolution works

- Here’s an example query resolution broken down step-wise:

Client A’s DNS server is trying to resolve www.example.com as follows:

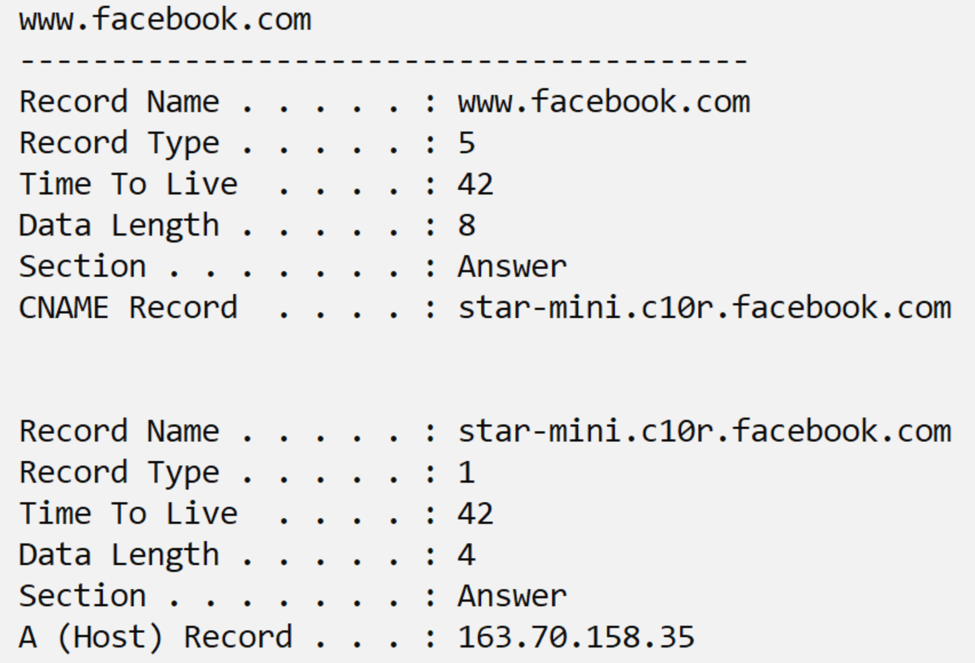
1. It first connects to a root DNS server and asks for www.example.com.
2. The root DNS server directs it to ask the .com server.
3. Our client’s DNS server then connects to the server responsible for .com and asks for www.example.com.
4. The .com DNS server gives our server the address of another server – the example.com DNS server.
5. Our DNS server finally connects to that server and asks about the record for www.example.com.
6. The example.com server then shares the address of www.example.com.
7. Our client’s DNS server relays it back to our client.



2. How hostname resolution works

- On Windows, you can show your local DNS cache with the following command:

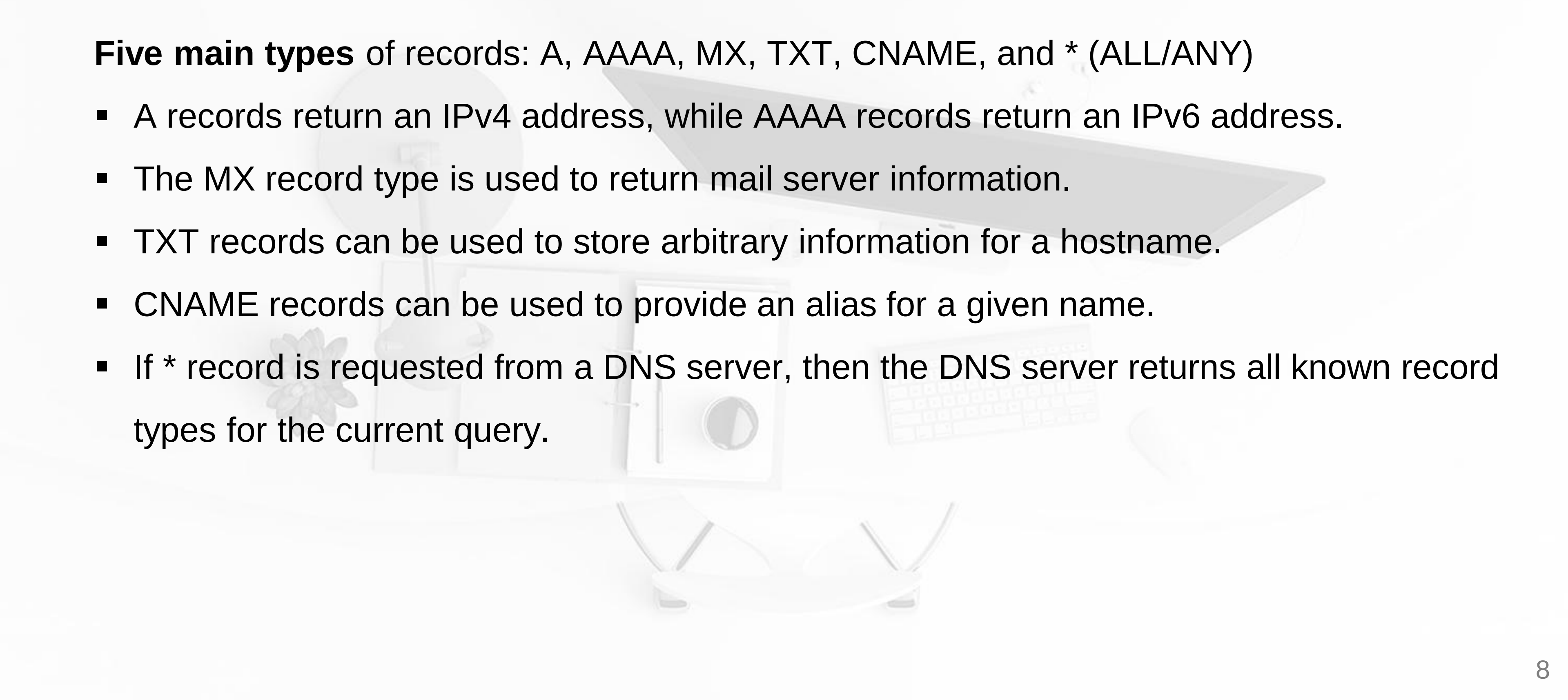
ipconfig /displaydns



```
www.facebook.com
----------------------------------------
Record Name . . . . . : www.facebook.com
Record Type . . . . . : 5
Time To Live  . . . . : 42
Data Length . . . . . : 8
Section . . . . . . . : Answer
CNAME Record  . . . . : star-mini.c10r.facebook.com


Record Name . . . . . : star-mini.c10r.facebook.com
Record Type . . . . . : 1
Time To Live  . . . . : 42
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 163.70.158.35
```

3. DNS record types

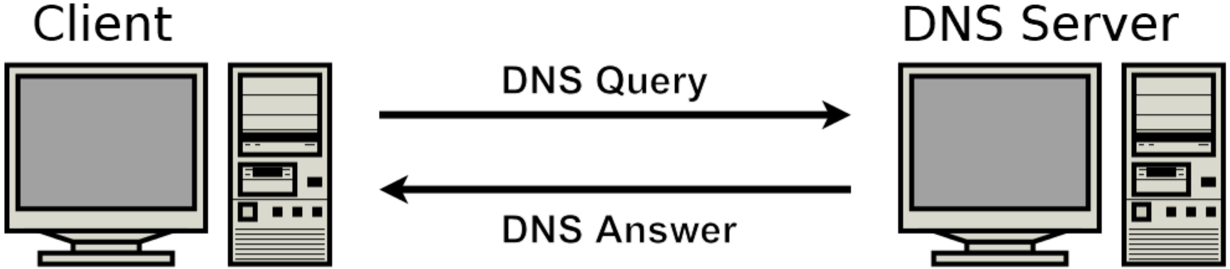


Five main types of records: A, AAAA, MX, TXT, CNAME, and * (ALL/ANY)

- A records return an IPv4 address, while AAAA records return an IPv6 address.
- The MX record type is used to return mail server information.
- TXT records can be used to store arbitrary information for a hostname.
- CNAME records can be used to provide an alias for a given name.
- If * record is requested from a DNS server, then the DNS server returns all known record types for the current query.

4. The DNS protocol

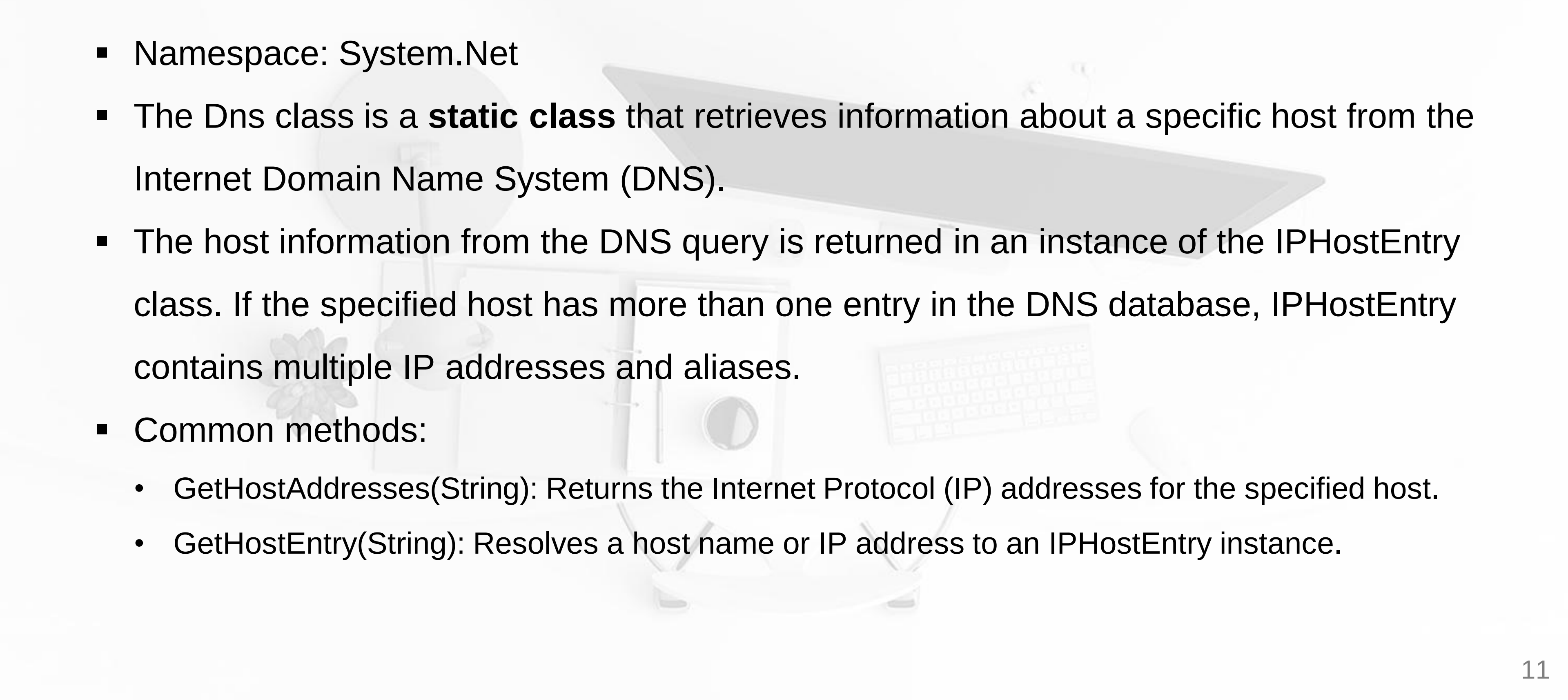
- When a client wants to resolve a hostname into an IP address, it sends a DNS query to a DNS server over UDP using port 53.
- The DNS server then performs the lookup, if possible, and returns an answer.



4. The DNS protocol

- If the query (or, more commonly, the answer) is too large to fit into one UDP packet, then the query can be performed over TCP instead of UDP. This is called **TCP fallback or DNS transport over TCP**.
- UDP works for most cases, and used the vast majority of the time.
- The client must know the IP address of at least one DNS server. DNS servers are usually provided by your ISP.

Dns Class



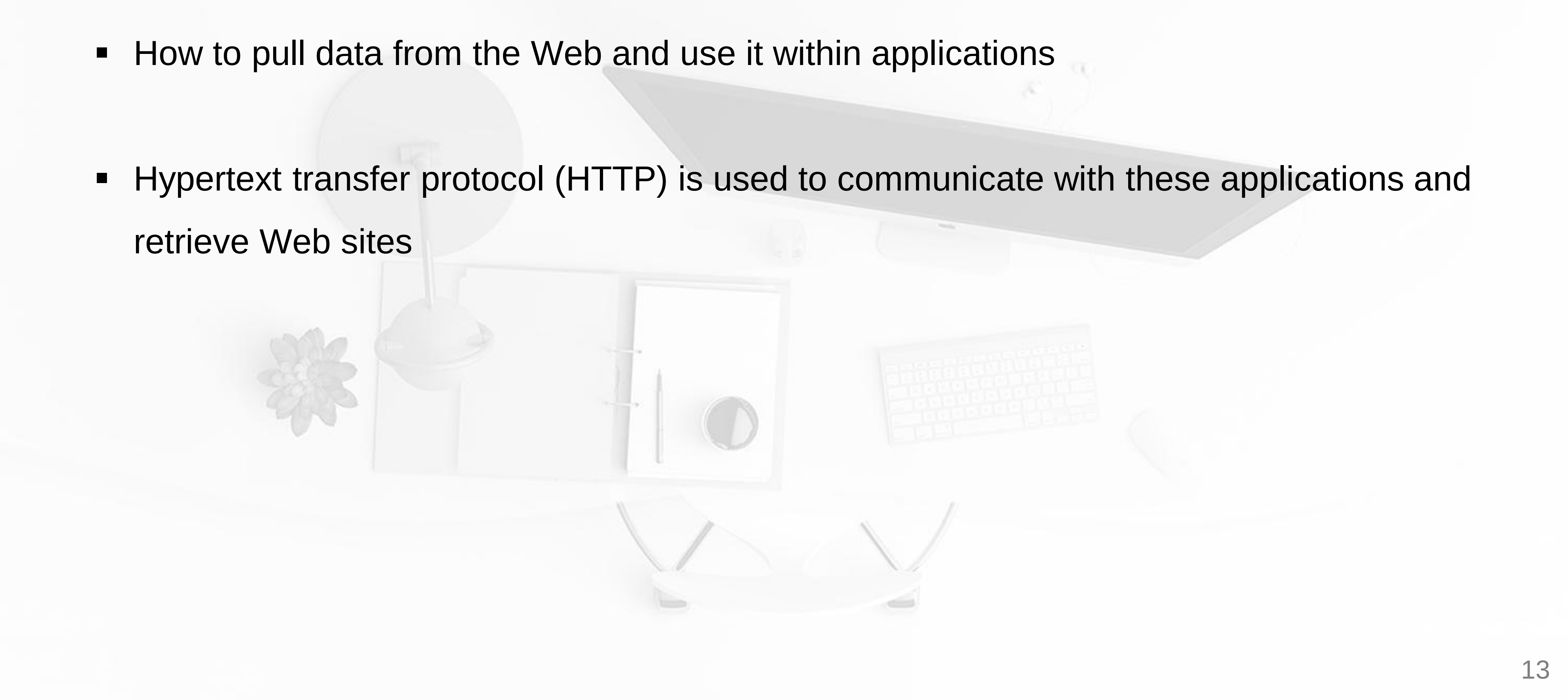
- Namespace: System.Net
- The Dns class is a **static class** that retrieves information about a specific host from the Internet Domain Name System (DNS).
- The host information from the DNS query is returned in an instance of the IPHostEntry class. If the specified host has more than one entry in the DNS database, IPHostEntry contains multiple IP addresses and aliases.
- Common methods:
  - GetHostAddresses(String): Returns the Internet Protocol (IP) addresses for the specified host.
  - GetHostEntry(String): Resolves a host name or IP address to an IPHostEntry instance.

HTTP
Communicating with Web Servers

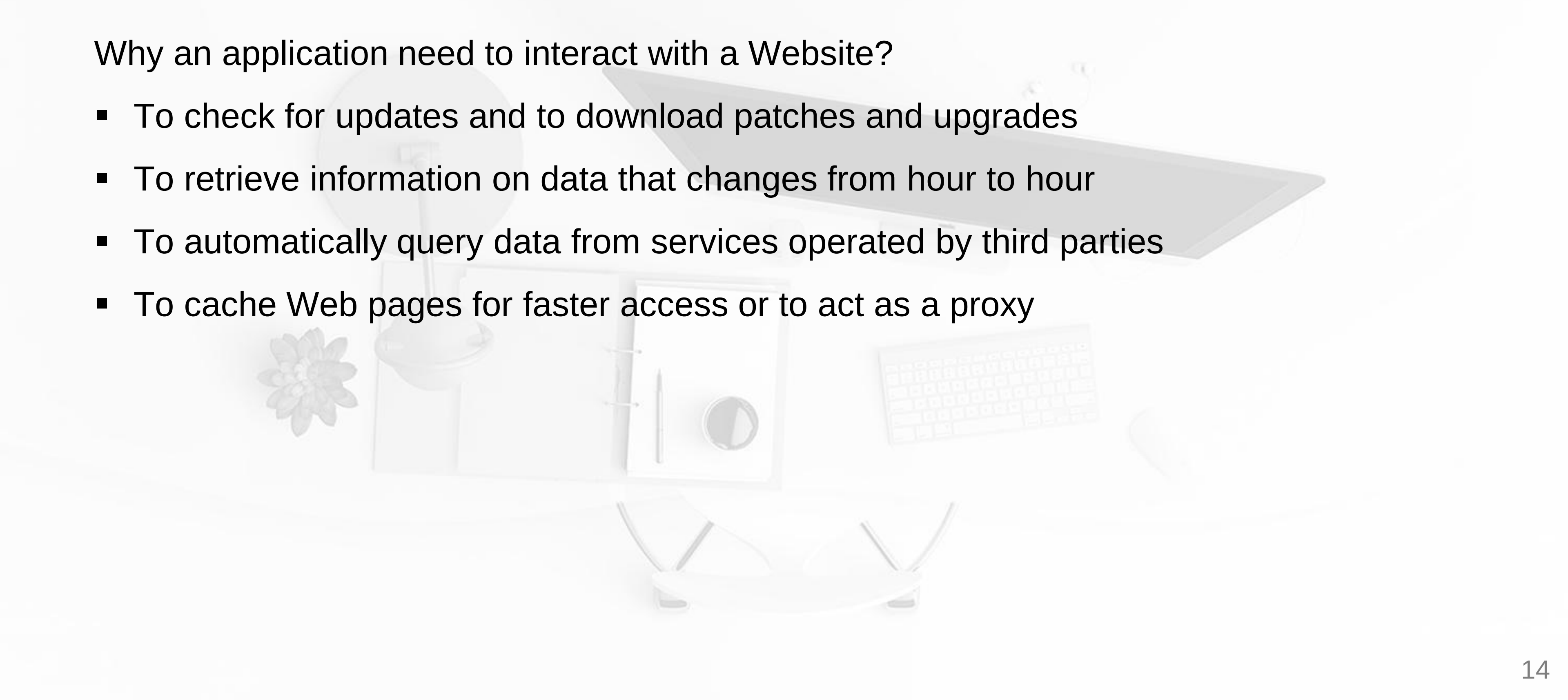
1. Introduction

- How to pull data from the Web and use it within applications

- Hypertext transfer protocol (HTTP) is used to communicate with these applications and retrieve Web sites



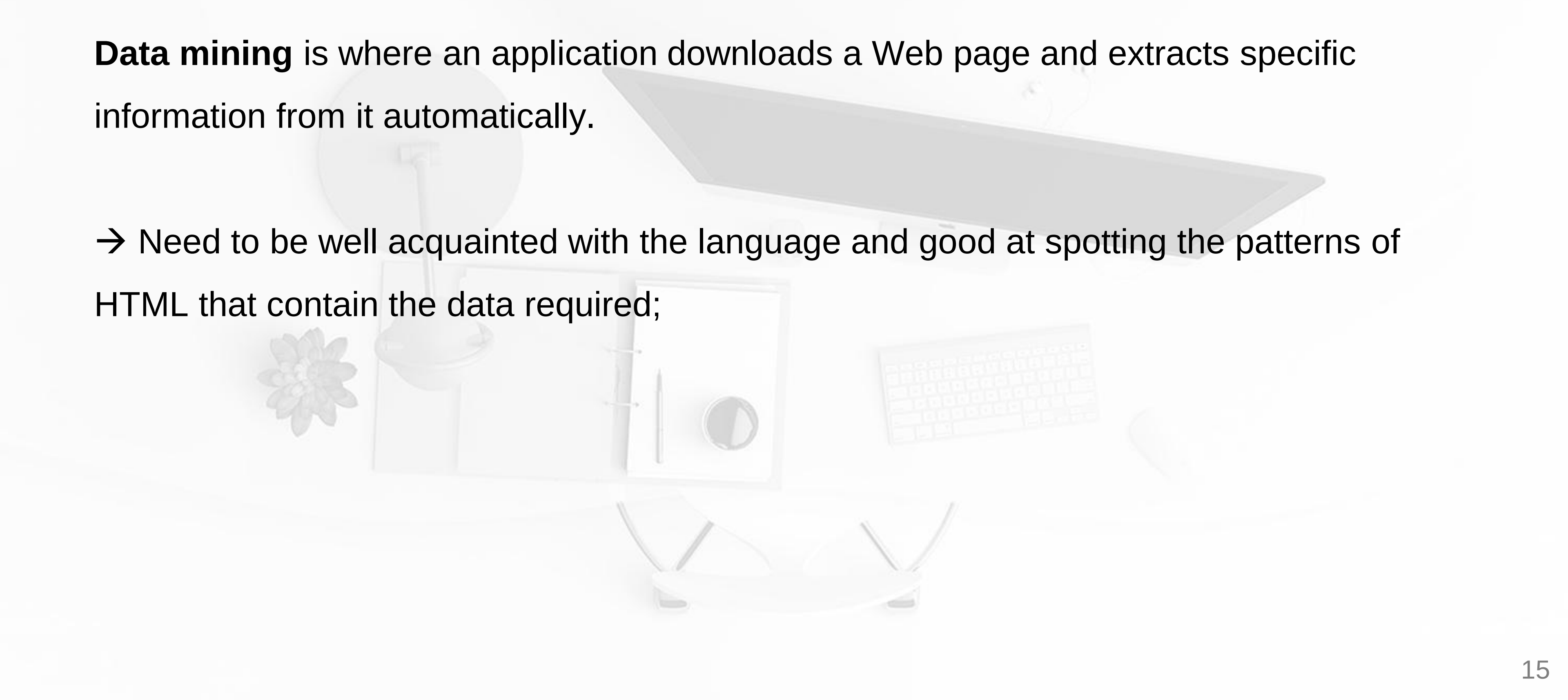
1. Introduction



Why an application need to interact with a Website?

- To check for updates and to download patches and upgrades
- To retrieve information on data that changes from hour to hour
- To automatically query data from services operated by third parties
- To cache Web pages for faster access or to act as a proxy

1. Introduction



Data mining is where an application downloads a Web page and extracts specific information from it automatically.

→ Need to be well acquainted with the language and good at spotting the patterns of HTML that contain the data required;

2. HTTP

- HTTP is the standard protocol for communication between web browsers and web servers.
- HTTP specifies how a client and server establish a connection, how the client requests data from the server, how the server responds to that request, and finally, how the connection is closed.
- HTTP connections use the TCP/IP protocol for data transfer.
- Plain HTTP is mostly deprecated for security reasons.
- Today, sites should use HTTPS, the secure version of HTTP.

2. HTTP

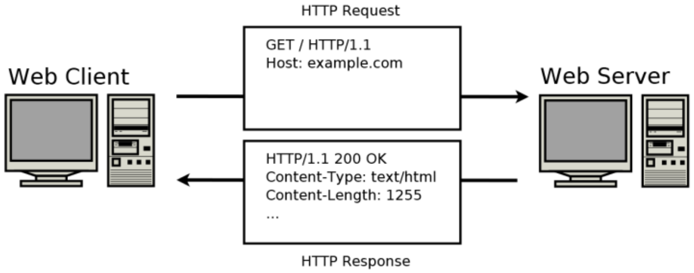
- For each request from client to server, there is a sequence of four steps:

1. The client opens a TCP connection to the server on port 80, by default; other ports may be specified in the URL.

2. The client sends a message to the server requesting the resource at a specified path. The request includes a header, and optionally (depending on the nature of the request) a blank line followed by data for the request.

3. The server sends a response to the client. The response begins with a response code, followed by a header full of metadata, a blank line, and the requested document or an error message.

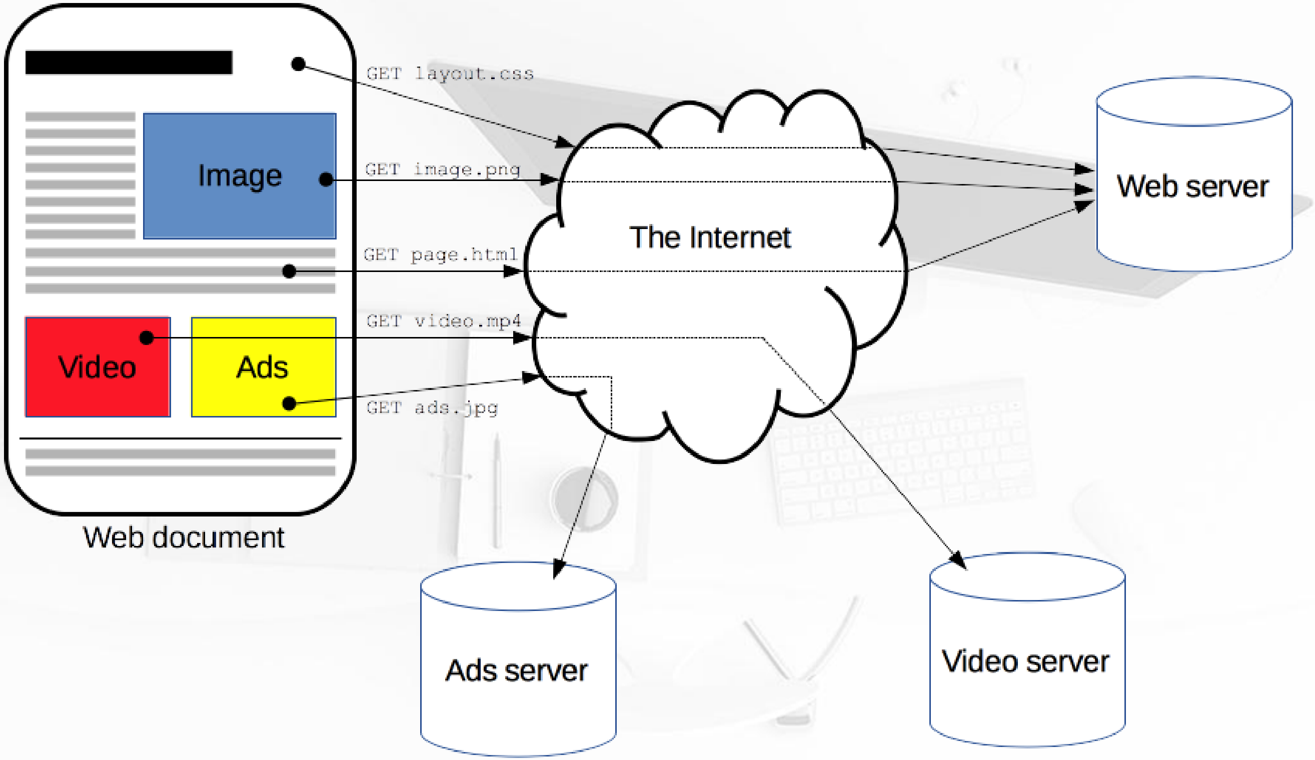
4. The server closes the connection.



```
GET / HTTP/1.1
Host: example.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1255
...
```

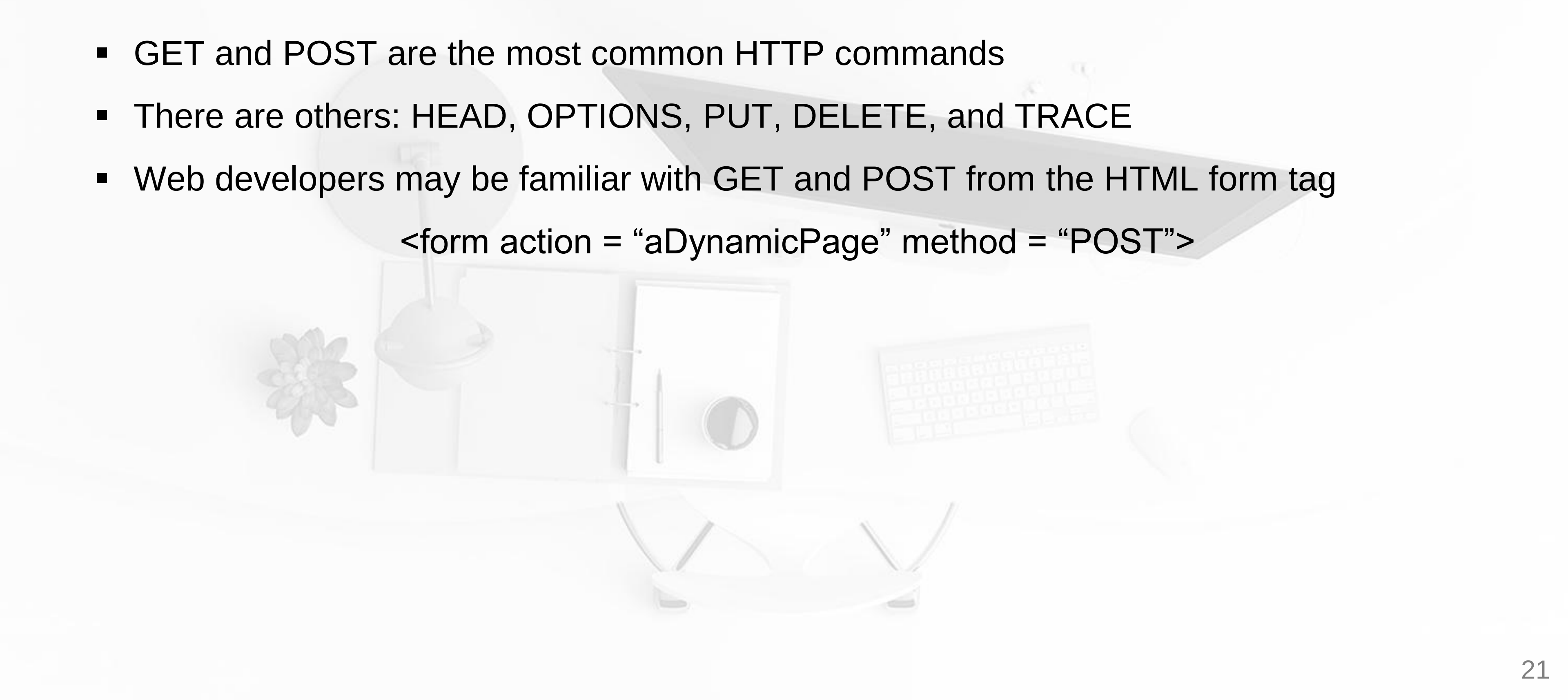
2. HTTP


GET layout.css
Image
GET image.png
GET page.html
GET video.mp4
Video
Ads
GET ads.jpg
Web document
The Internet
Web server
Ads server
Video server

2.1. HTTP request

- GET and POST are the most common HTTP commands
- There are others: HEAD, OPTIONS, PUT, DELETE, and TRACE
- Web developers may be familiar with GET and POST from the HTML form tag

<form action = “aDynamicPage” method = “POST”>

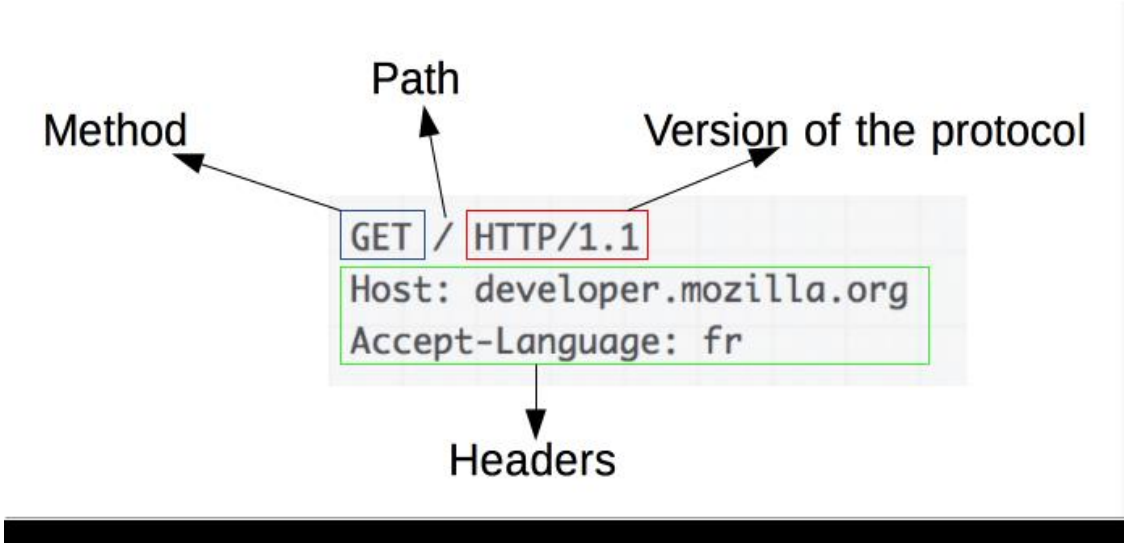


2.1. HTTP request

- The simplest HTTP request:

  GET /

  <enter><enter>



```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

2.1. HTTP request

- HTTP requests are generally more complex:

  GET / HTTP/1.1

  Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*

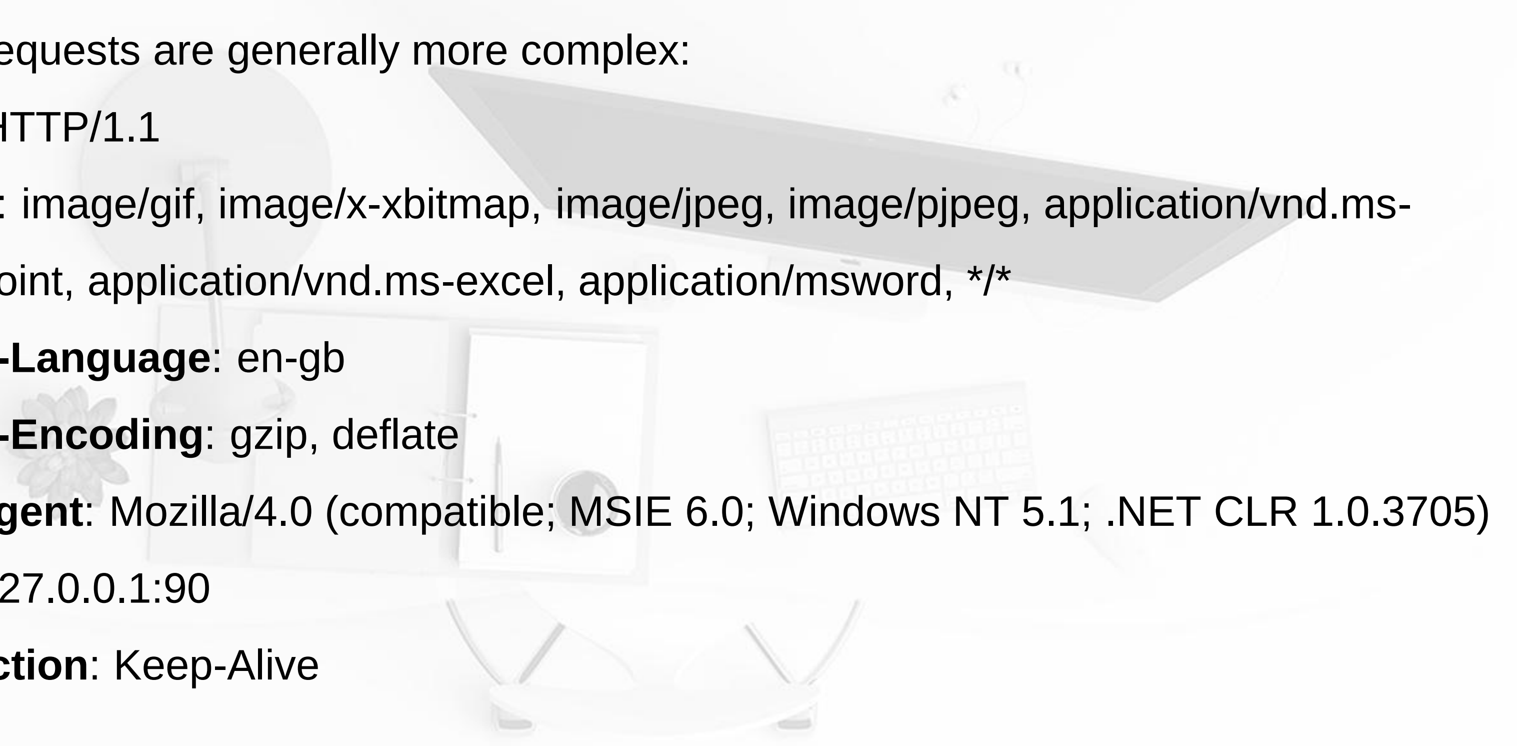
  Accept-Language: en-gb

  Accept-Encoding: gzip, deflate

  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)

  Host: 127.0.0.1:90

  Connection: Keep-Alive



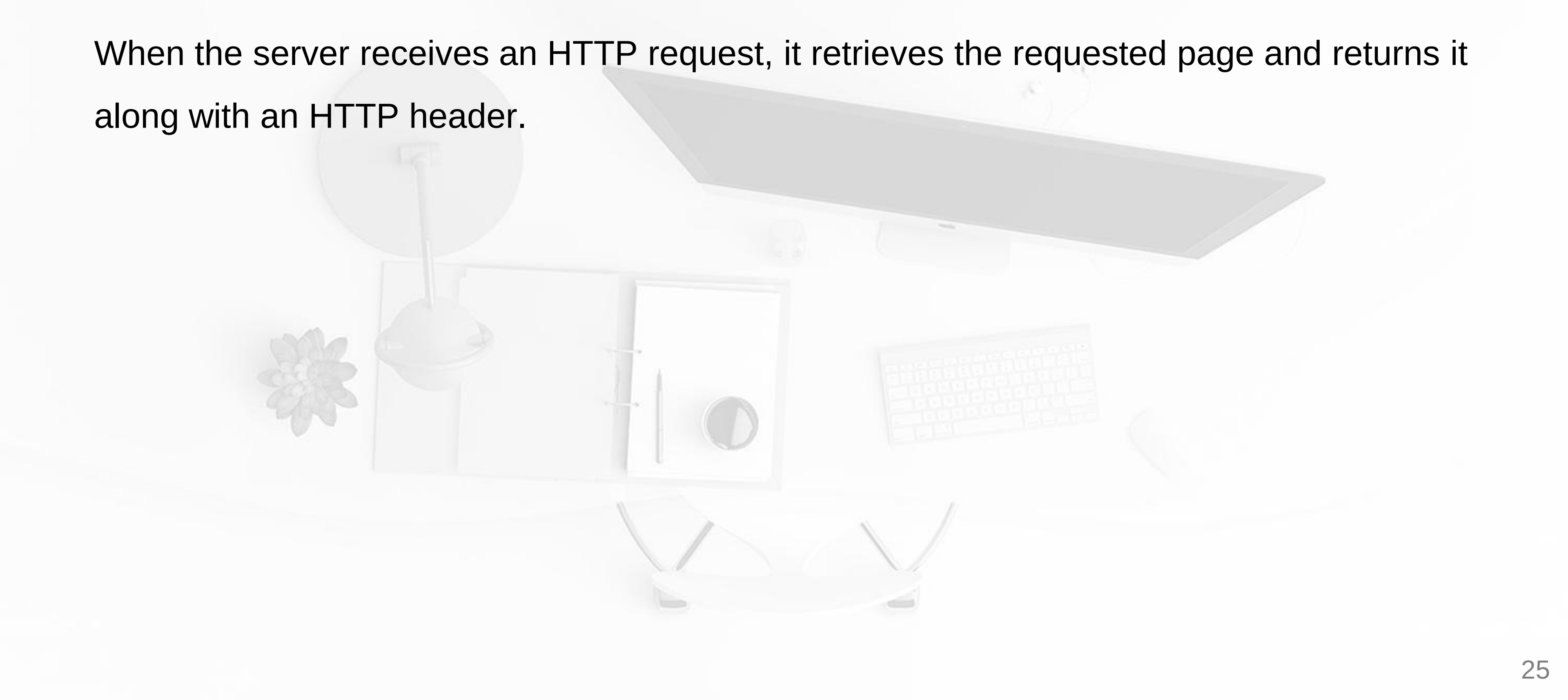
2.1. HTTP request

- The POST request:

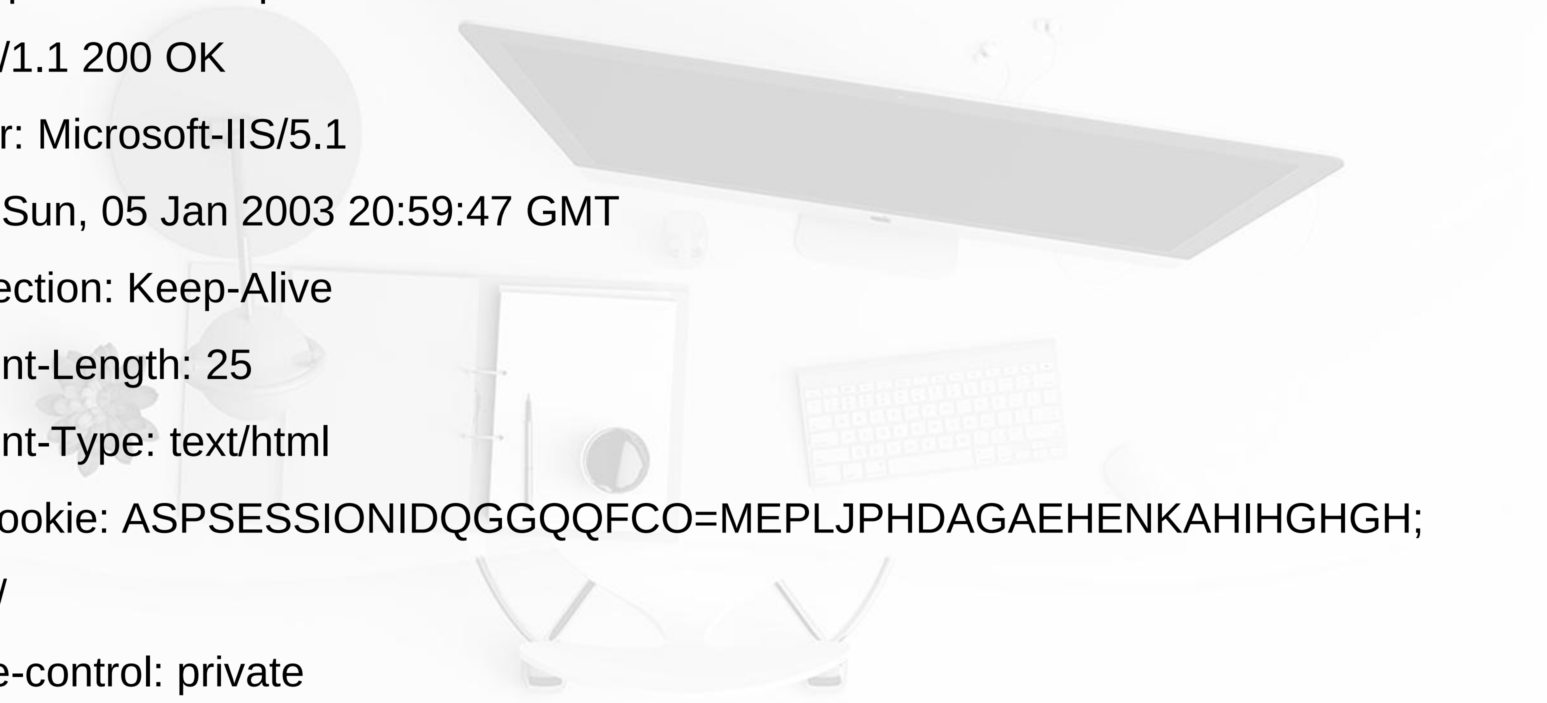
```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 17
myField=some+text
```

2.2. HTTP response

When the server receives an HTTP request, it retrieves the requested page and returns it along with an HTTP header.



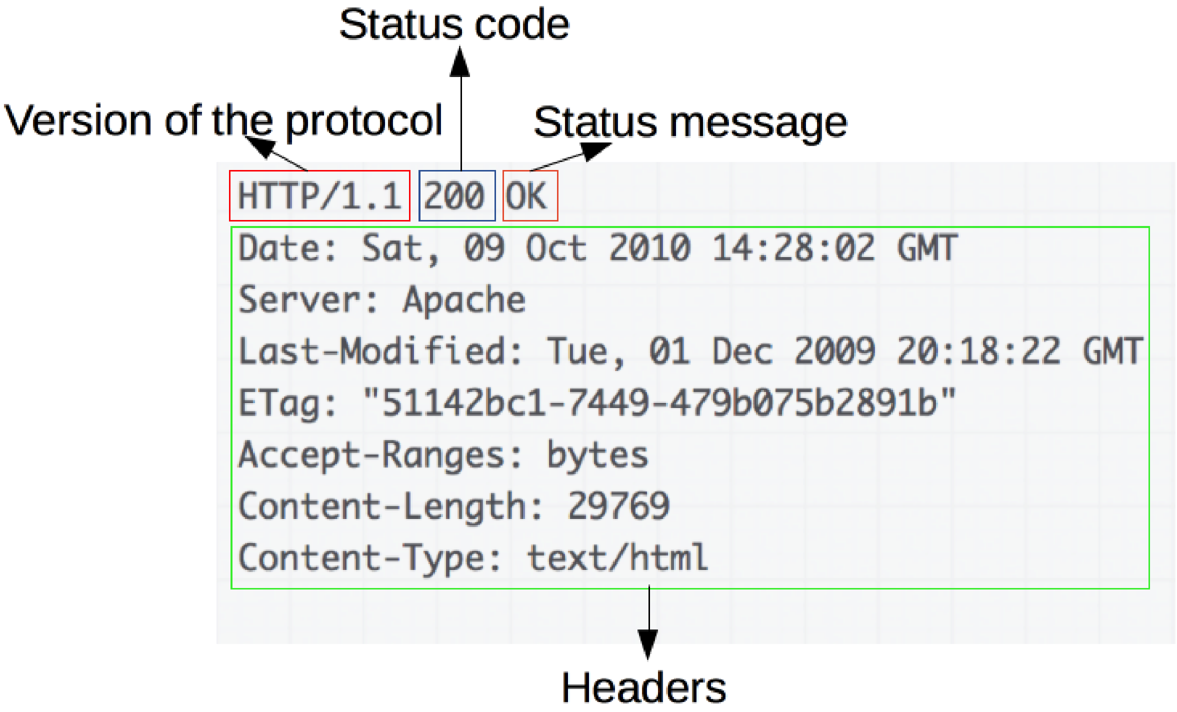
2.2. HTTP response



- A sample HTTP response:

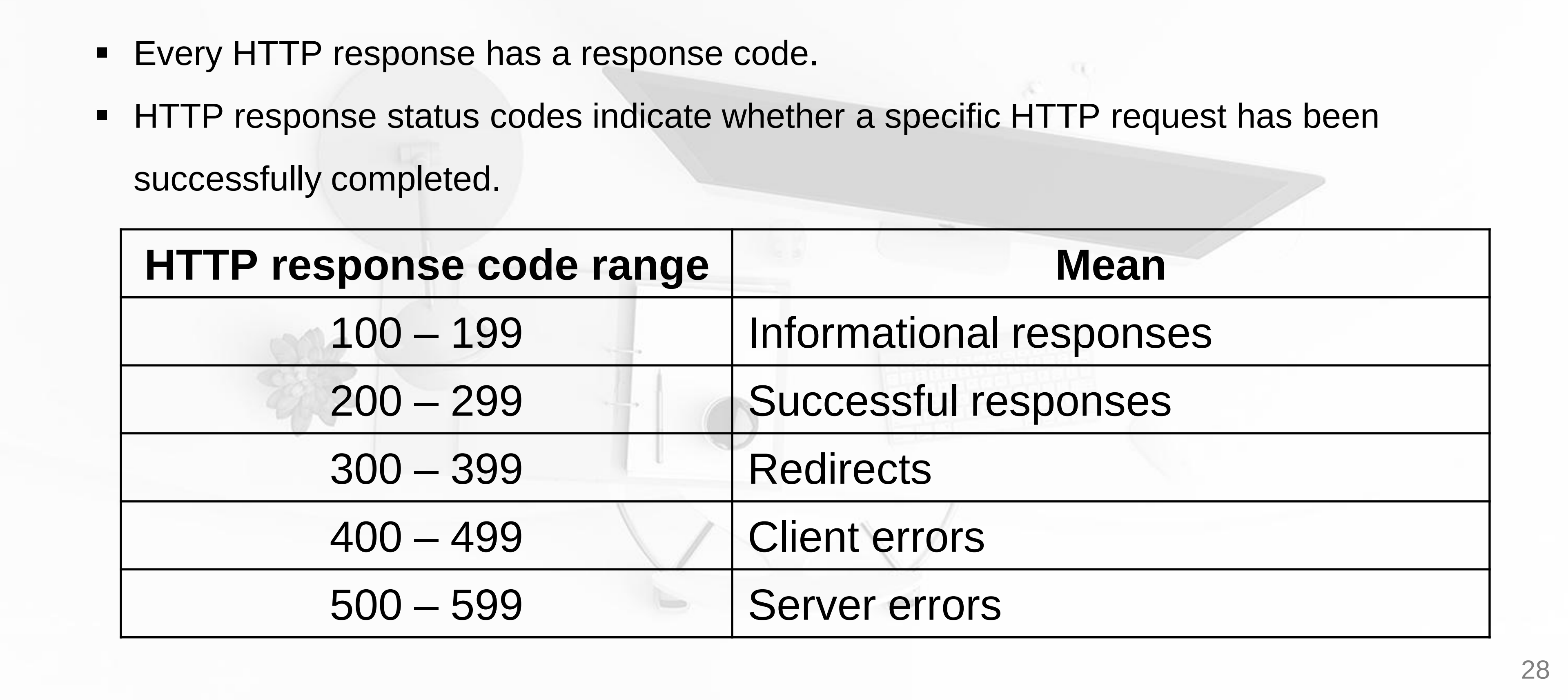
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Sun, 05 Jan 2003 20:59:47 GMT
Connection: Keep-Alive
Content-Length: 25
Content-Type: text/html
Set-Cookie: ASPSESSIONIDQGGQQFCO=MEPLJPHDAGAEHENKAHIHGHGH;
path=/
Cache-control: private
This is a test html page!
```

2.2. HTTP response



```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

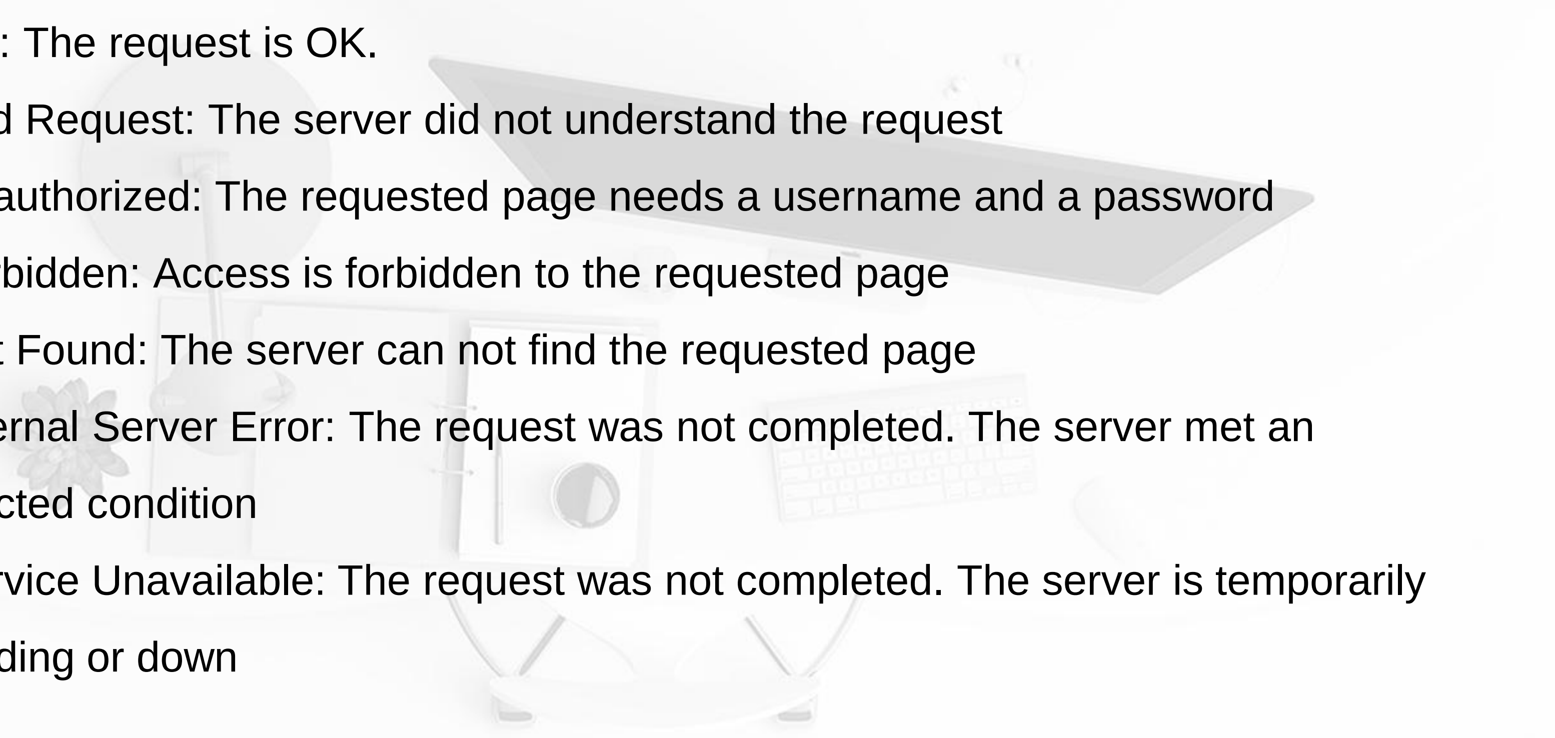
2.2. HTTP response



- Every HTTP response has a response code.
- HTTP response status codes indicate whether a specific HTTP request has been successfully completed.

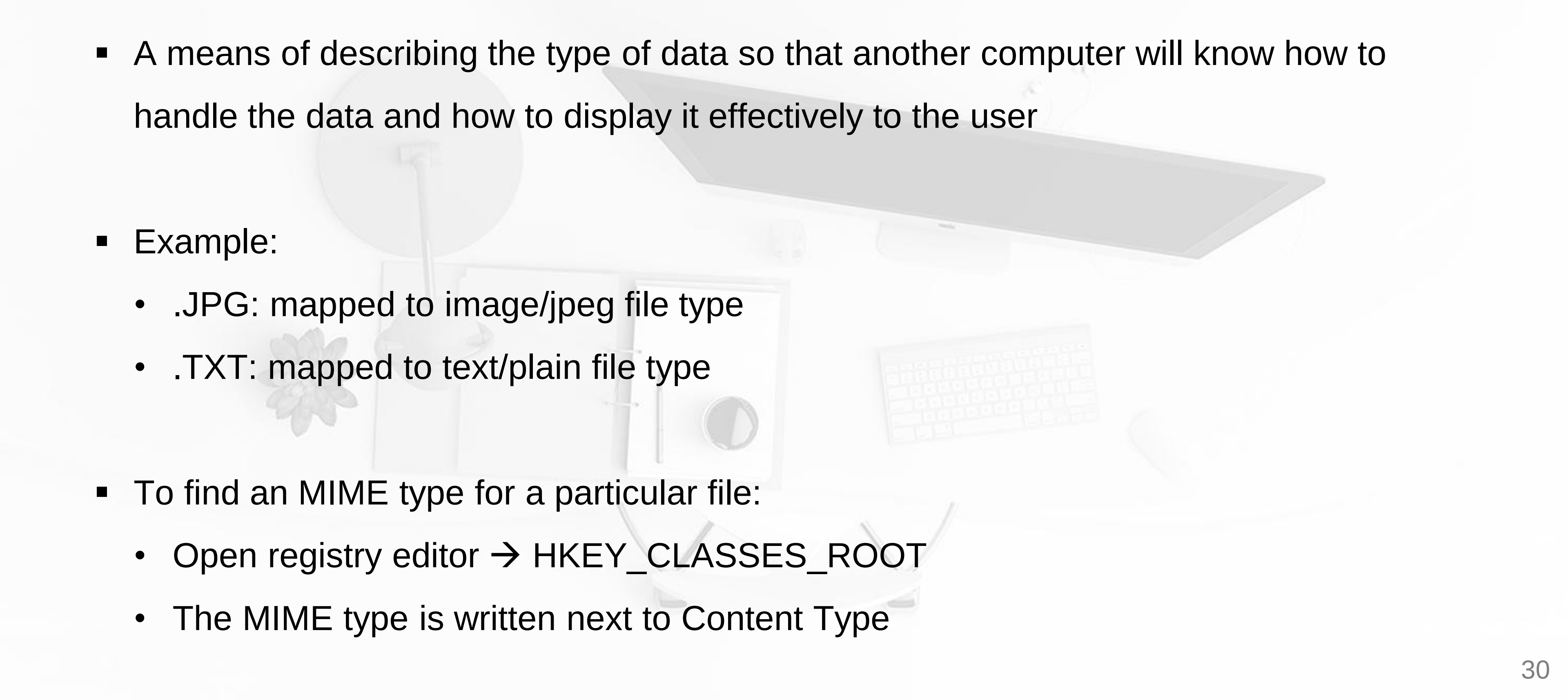
HTTP response code range	Mean
100 – 199	Informational responses
200 – 299	Successful responses
300 – 399	Redirects
400 – 499	Client errors
500 – 599	Server errors

2.2. HTTP response



- 200 OK: The request is OK.
- 400 Bad Request: The server did not understand the request
- 401 Unauthorized: The requested page needs a username and a password
- 403 Forbidden: Access is forbidden to the requested page
- 404 Not Found: The server can not find the requested page
- 500 Internal Server Error: The request was not completed. The server met an unexpected condition
- 503 Service Unavailable: The request was not completed. The server is temporarily overloading or down

2.3. MIME – Multipart Internet mail extensions



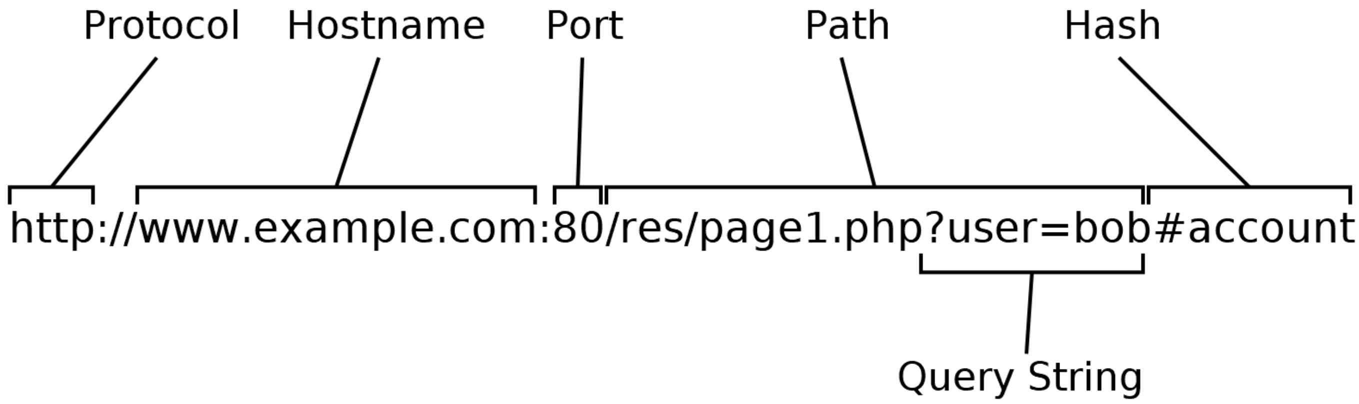
- A means of describing the type of data so that another computer will know how to handle the data and how to display it effectively to the user

- Example:
  - .JPG: mapped to image/jpeg file type
  - .TXT: mapped to text/plain file type

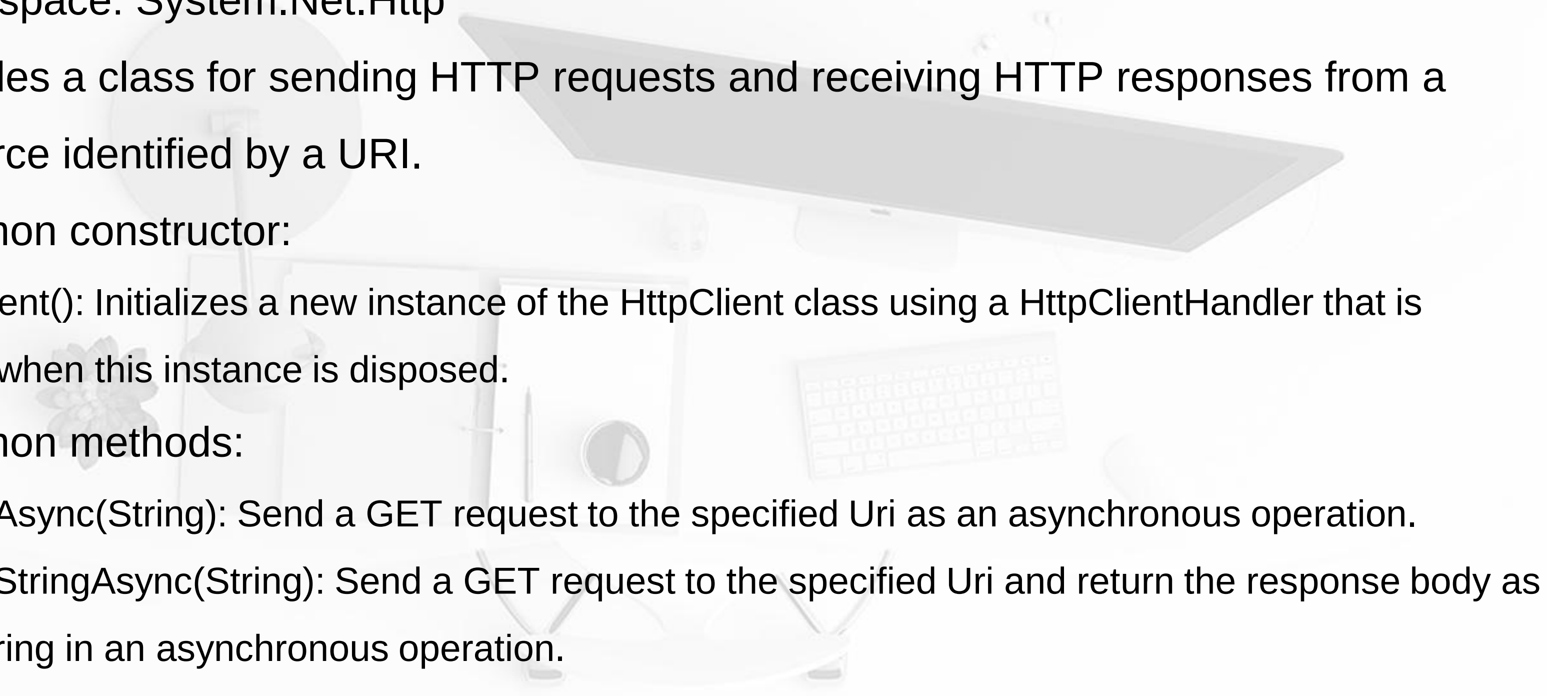
- To find an MIME type for a particular file:
  - Open registry editor → HKEY_CLASSES_ROOT
  - The MIME type is written next to Content Type

2.4. What’s in a URL

- Uniform Resource Locators (URL), also known as web addresses, provide a convenient way to specify particular web resources.
- You can navigate to a URL by typing it into your web browser’s address bar.
- Alternately, if you’re browsing a web page and click on a link, that link is indicated with a URL.



HttpClient Class

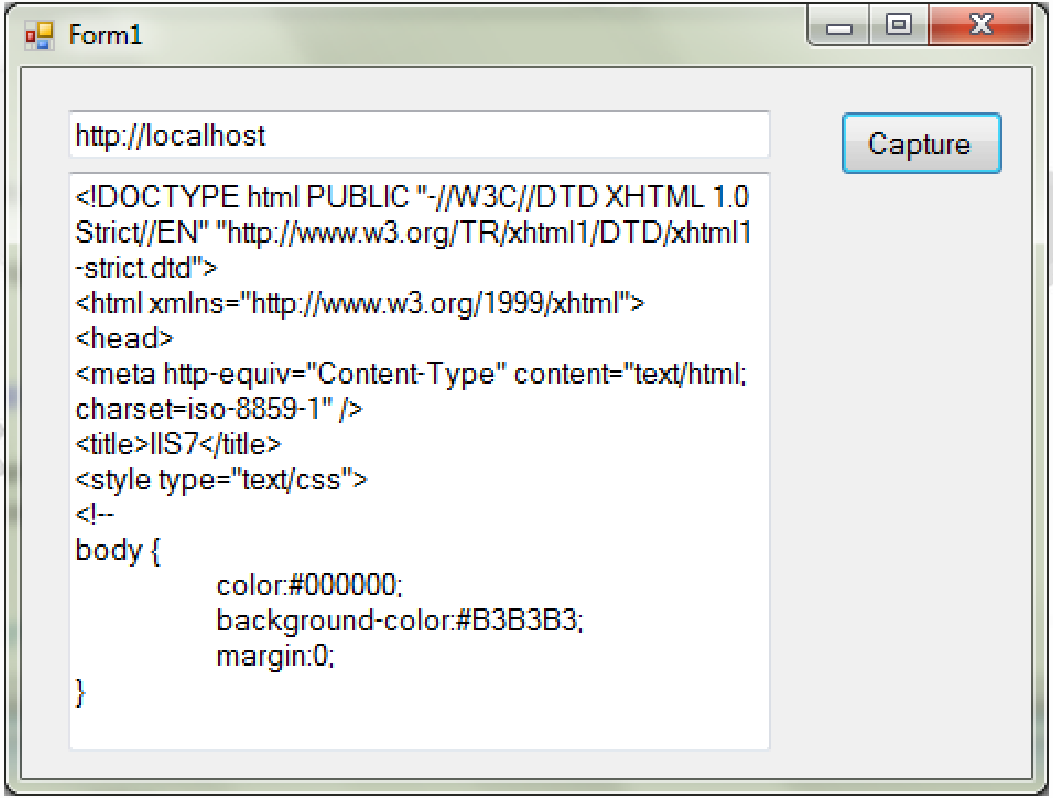


- Namespace: System.Net.Http
- Provides a class for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.
- Common constructor:

  HttpClient(): Initializes a new instance of the HttpClient class using a HttpClientHandler that is disposed when this instance is disposed.

- Common methods:
  - GetAsync(String): Send a GET request to the specified Uri as an asynchronous operation.
  - GetStringAsync(String): Send a GET request to the specified Uri and return the response body as a string in an asynchronous operation.

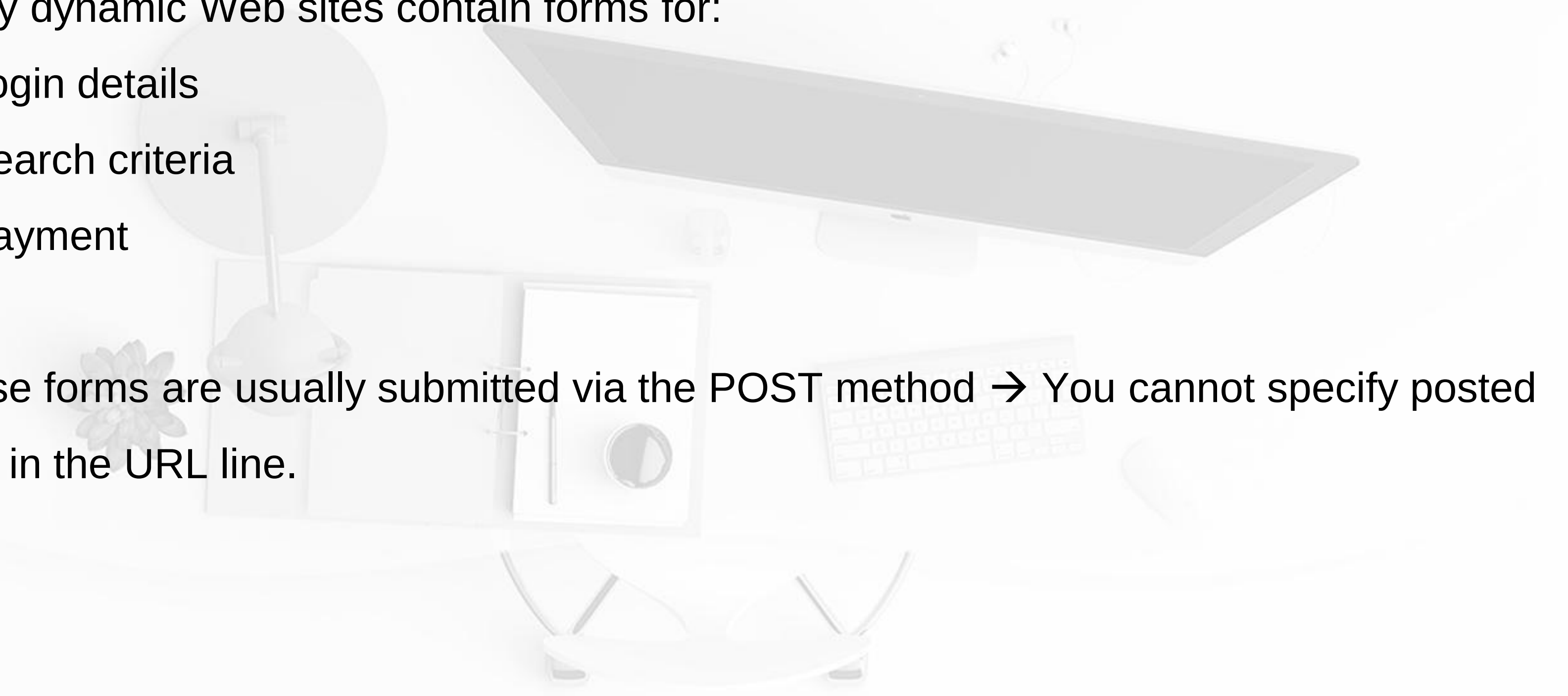
Exercise



http://localhost

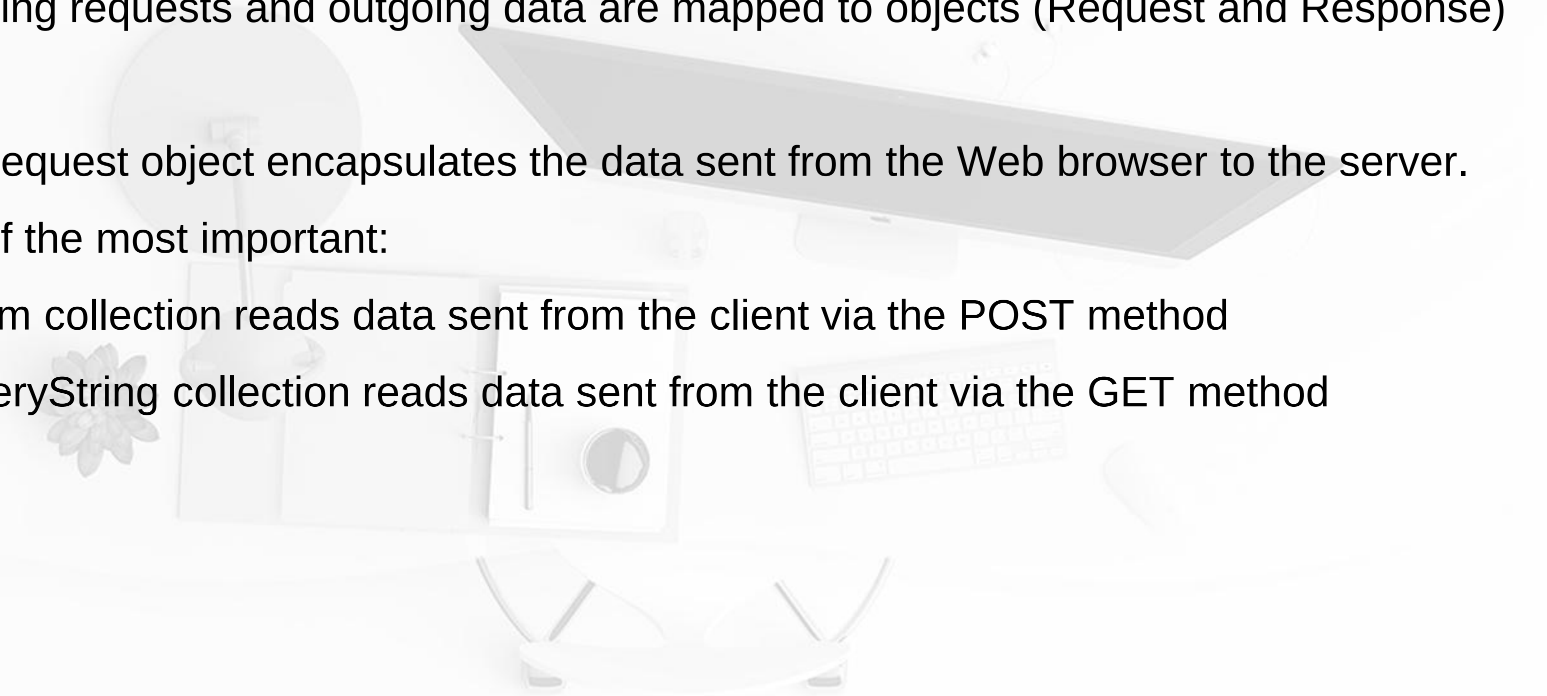
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1
-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
<title>IIS7</title>
<style type="text/css">
<!--
body {
        color:#000000;
        background-color:#B3B3B3;
        margin:0;
}
```

2.4. Posting data



- Many dynamic Web sites contain forms for:
  - Login details
  - Search criteria
  - Payment
- These forms are usually submitted via the POST method → You cannot specify posted data in the URL line.

2.4. Posting data

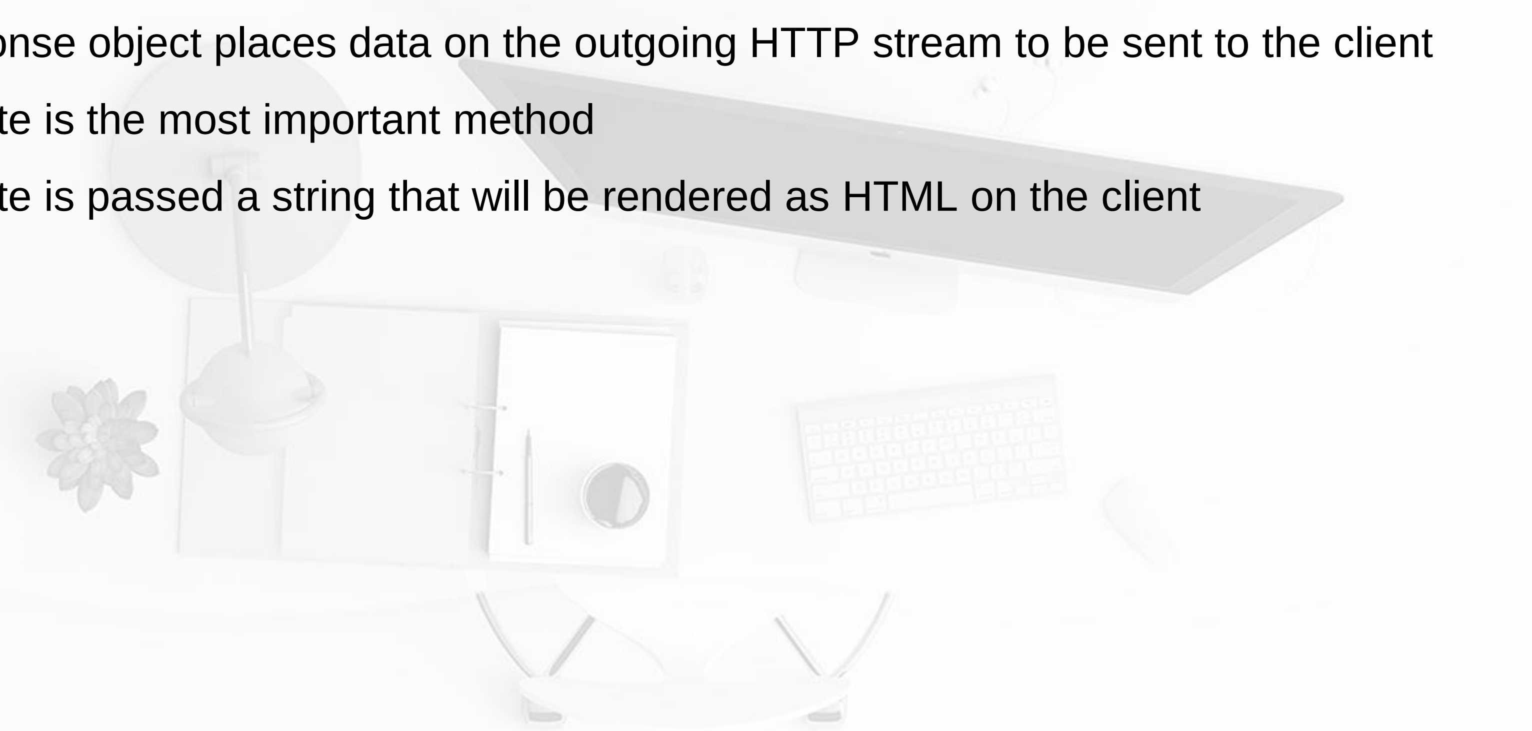


- Incoming requests and outgoing data are mapped to objects (Request and Response)

- The Request object encapsulates the data sent from the Web browser to the server.
- Two of the most important:
  - Form collection reads data sent from the client via the POST method
  - QueryString collection reads data sent from the client via the GET method

2.4. Posting data

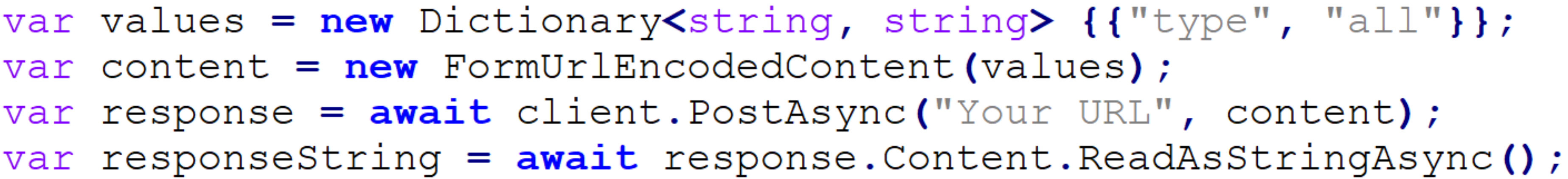
- Response object places data on the outgoing HTTP stream to be sent to the client
  - Write is the most important method
  - Write is passed a string that will be rendered as HTML on the client



HttpClient Class

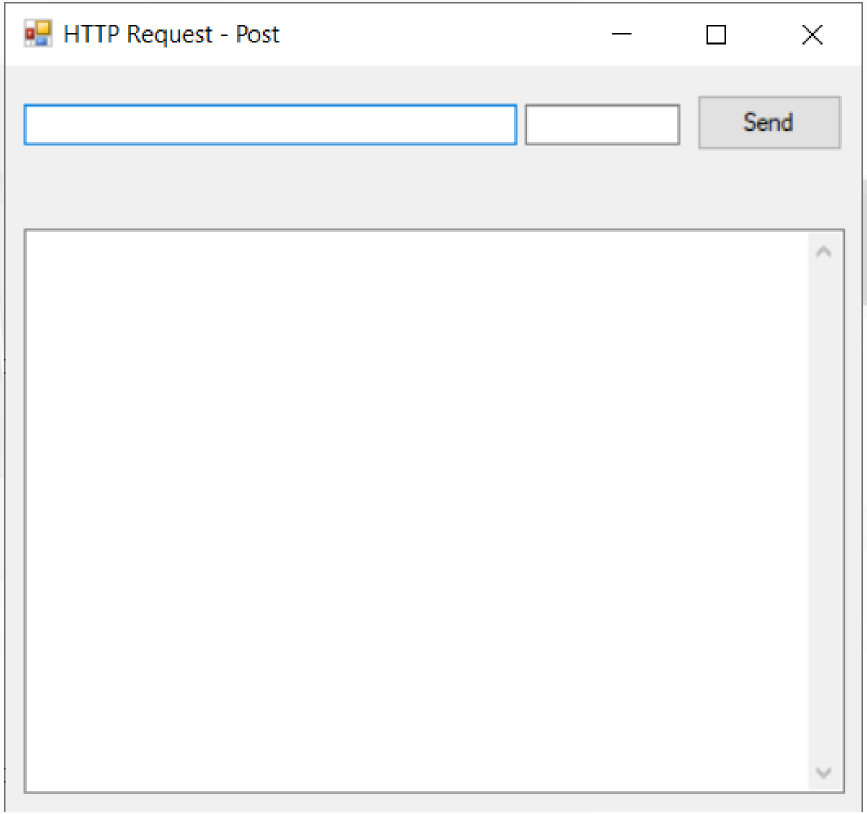
- Namespace: System.Net.Http
- Common methods:

  PostAsync(String, HttpContent): Send a POST request to the specified Uri as an asynchronous operation.



```
var values = new Dictionary<string, string> {{"type", "all"}};
var content = new FormUrlEncodedContent(values);
var response = await client.PostAsync("Your URL", content);
var responseString = await response.Content.ReadAsStringAsync();
```

Exercise

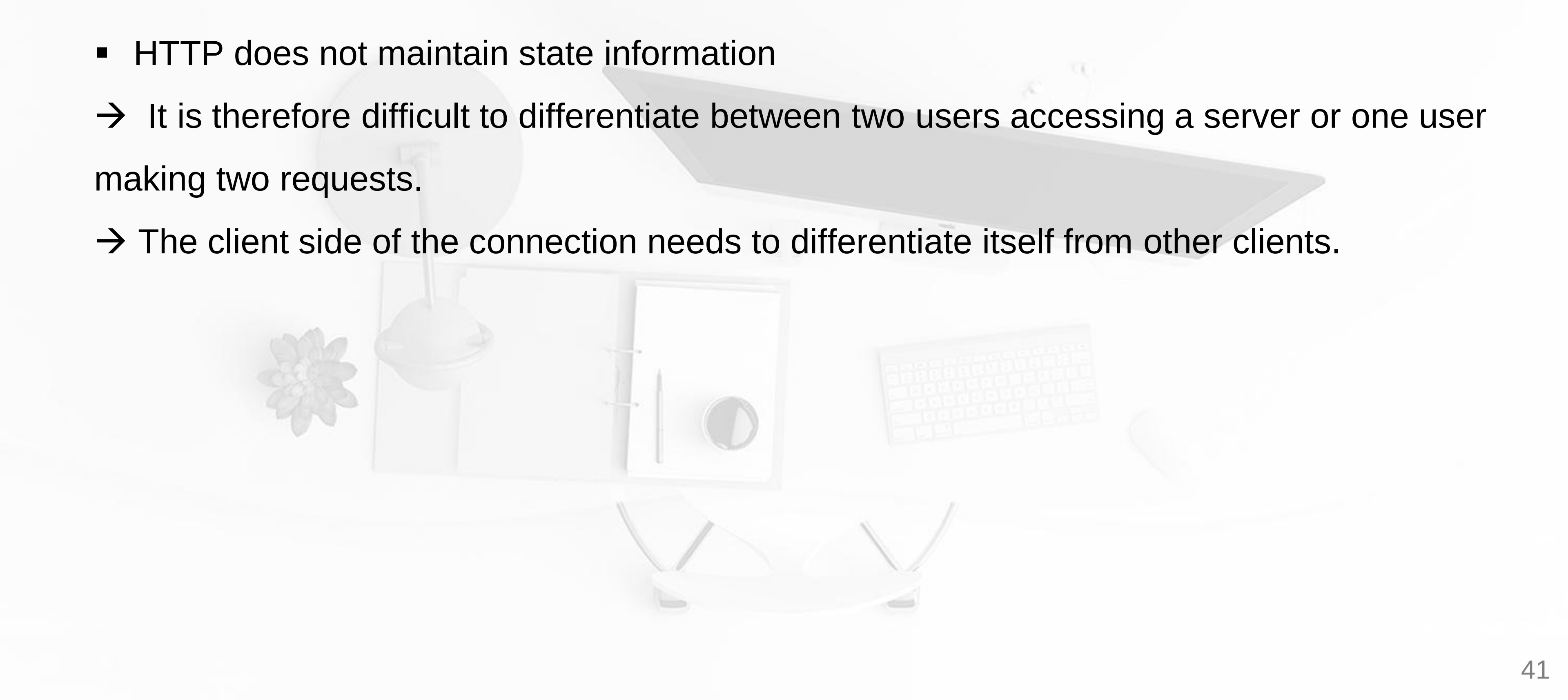


2.5. Cookie

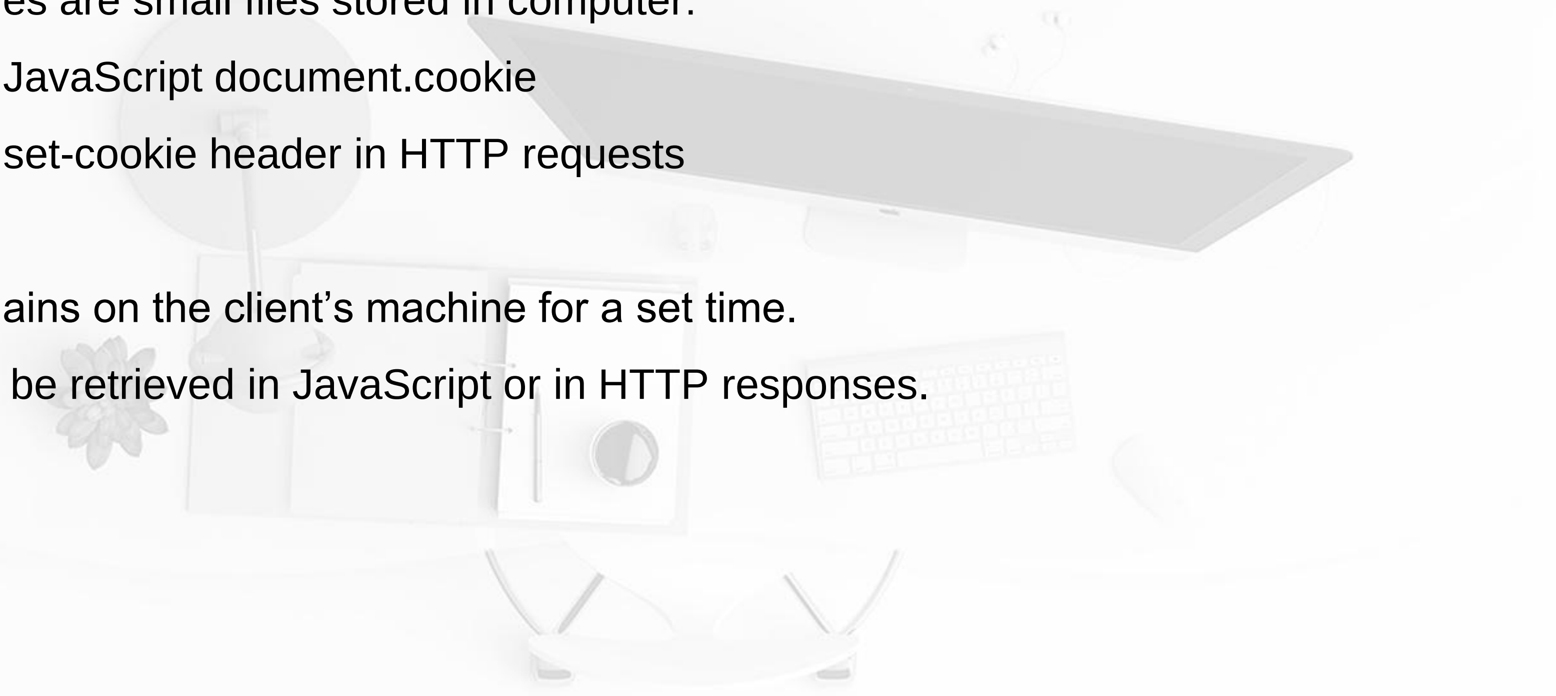
- HTTP does not maintain state information

→ It is therefore difficult to differentiate between two users accessing a server or one user making two requests.

→ The client side of the connection needs to differentiate itself from other clients.



2.5. Cookie



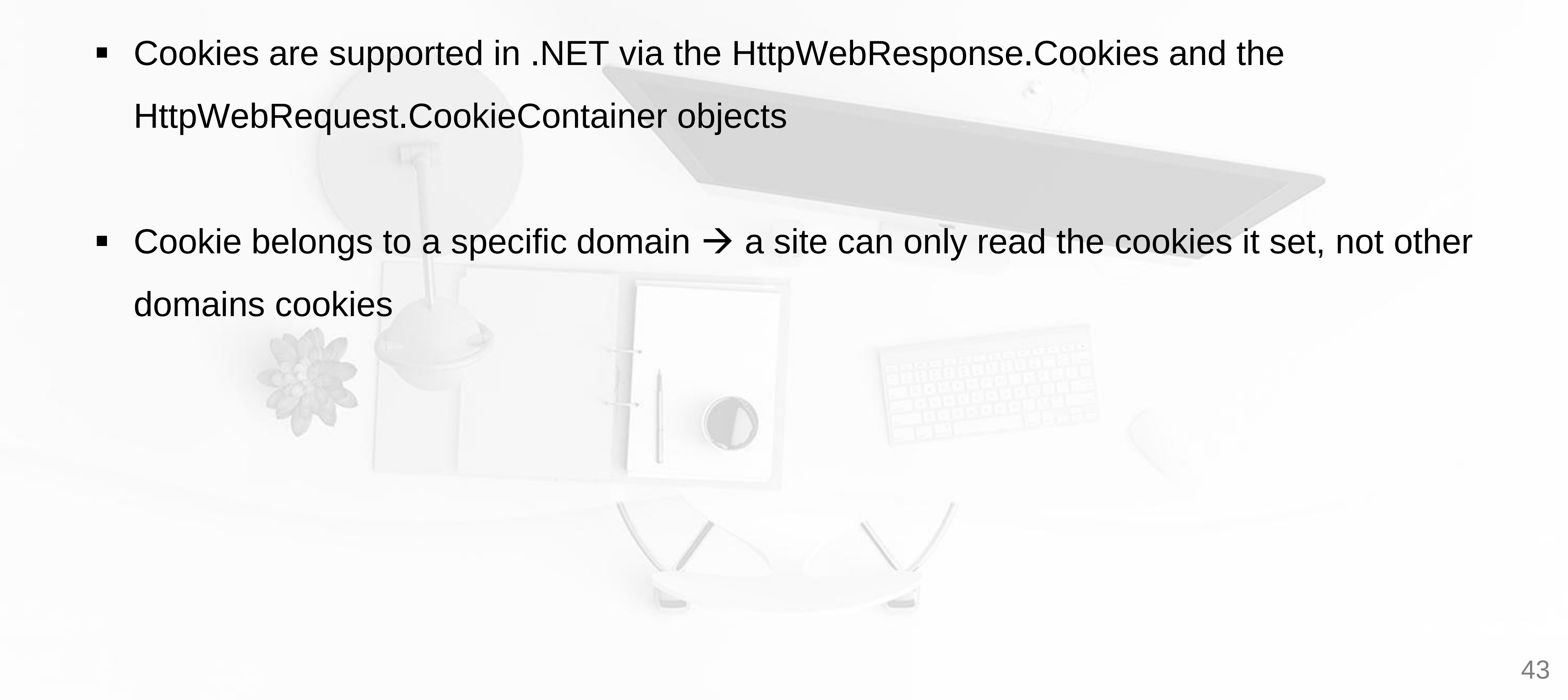
- Cookies are small files stored in computer:
  - By JavaScript document.cookie
  - By set-cookie header in HTTP requests

- It remains on the client’s machine for a set time.
- It can be retrieved in JavaScript or in HTTP responses.

2.5. Cookie

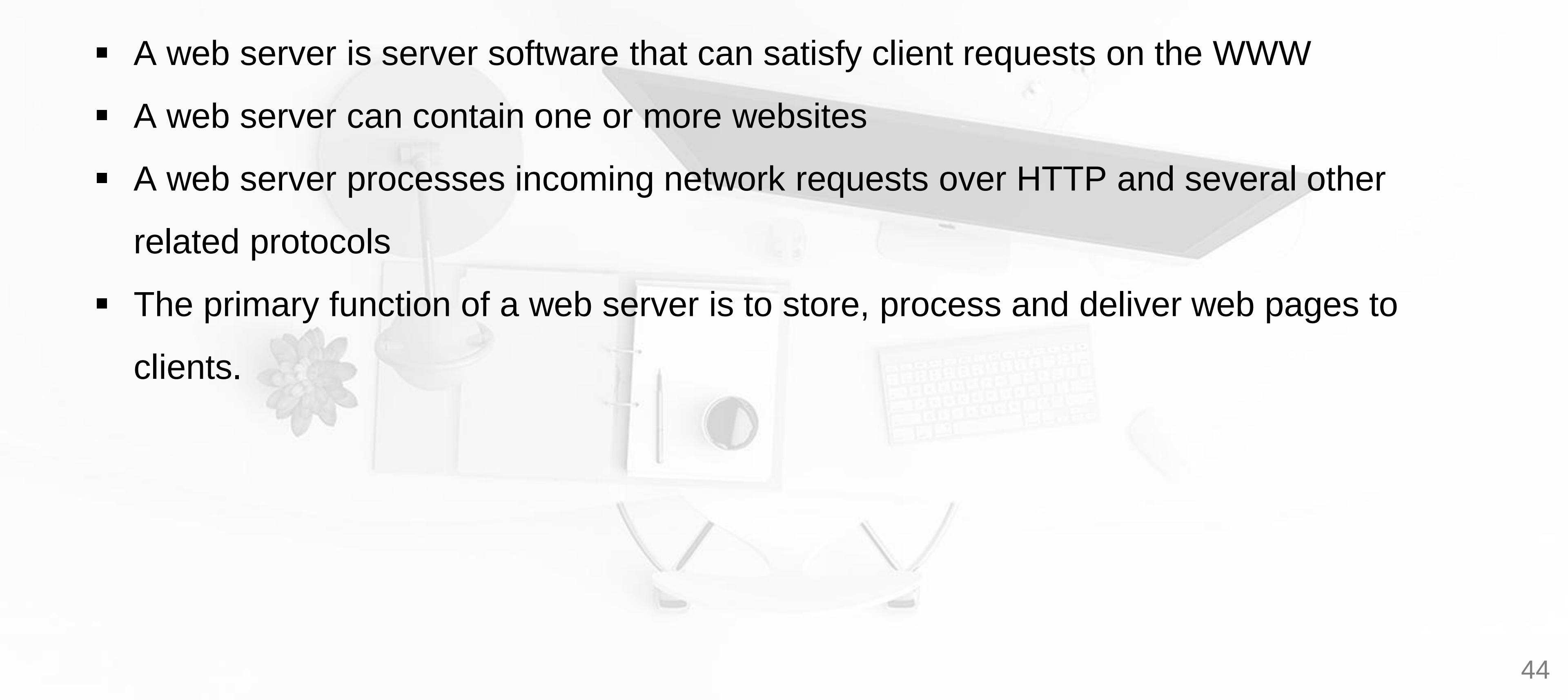
- Cookies are supported in .NET via the HttpWebResponse.Cookies and the HttpWebRequest.CookieContainer objects

- Cookie belongs to a specific domain → a site can only read the cookies it set, not other domains cookies



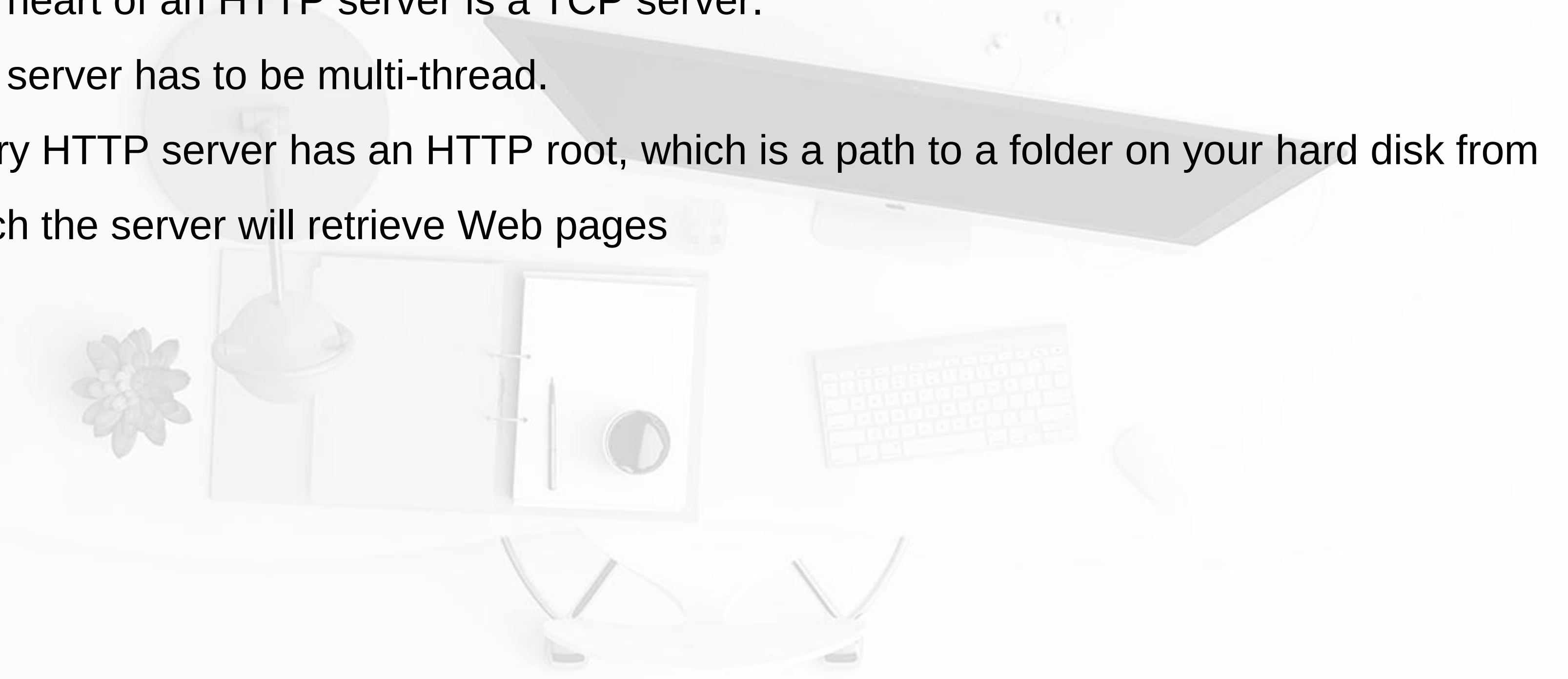
3. Web server

- A web server is server software that can satisfy client requests on the WWW
- A web server can contain one or more websites
- A web server processes incoming network requests over HTTP and several other related protocols
- The primary function of a web server is to store, process and deliver web pages to clients.



3. Web server

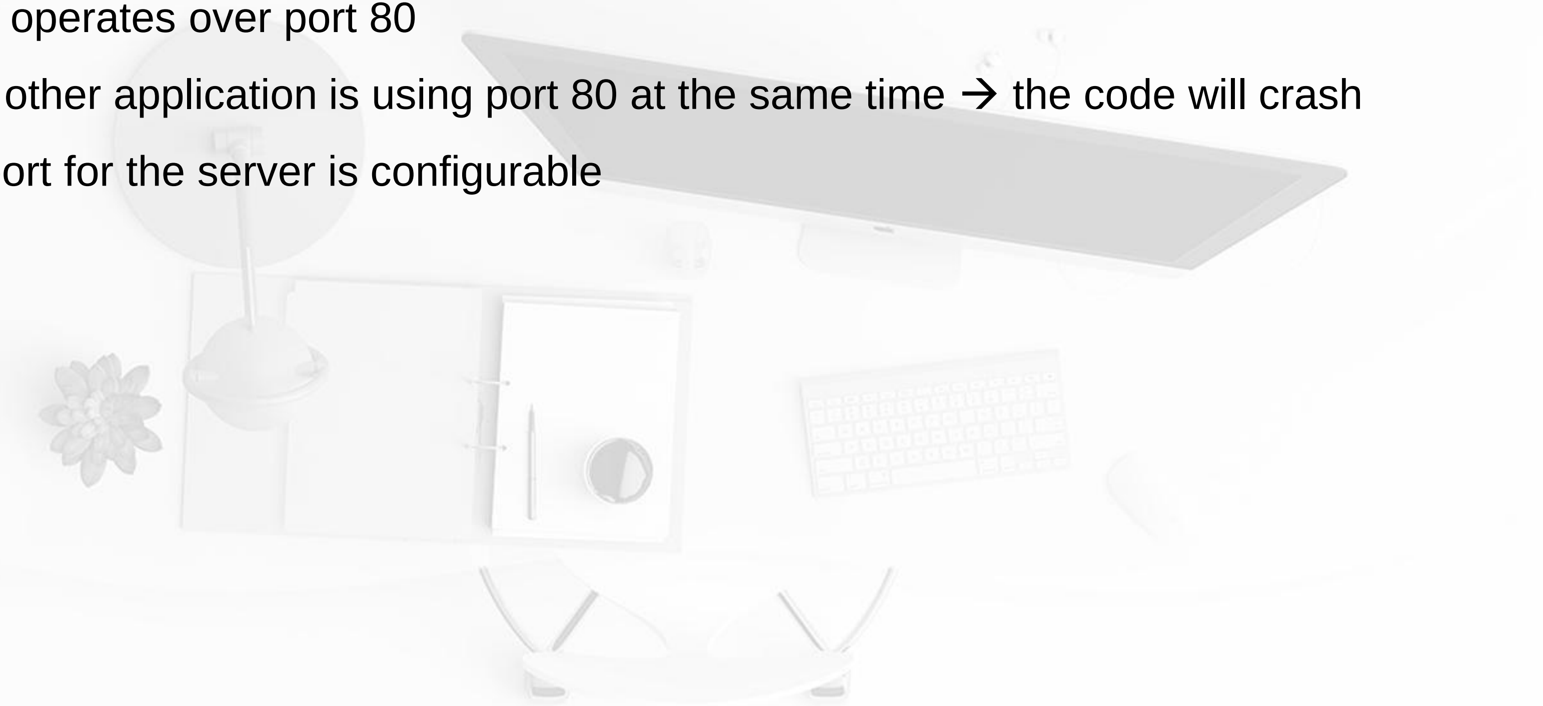
- The heart of an HTTP server is a TCP server.
- The server has to be multi-thread.
- Every HTTP server has an HTTP root, which is a path to a folder on your hard disk from which the server will retrieve Web pages



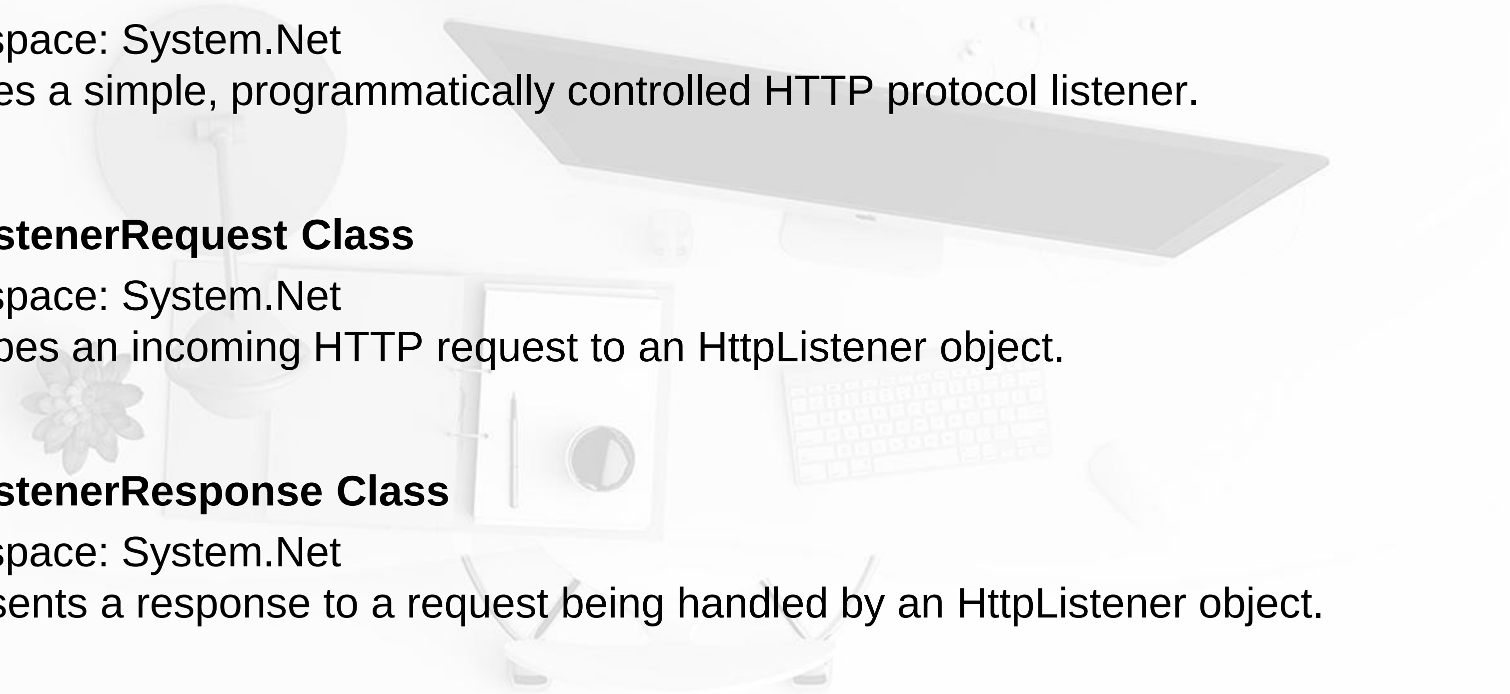
3. Web server

- HTTP operates over port 80
- If any other application is using port 80 at the same time → the code will crash

→ The port for the server is configurable



Library



- **HttpListener Class**

  Namespace: System.Net
  Provides a simple, programmatically controlled HTTP protocol listener.

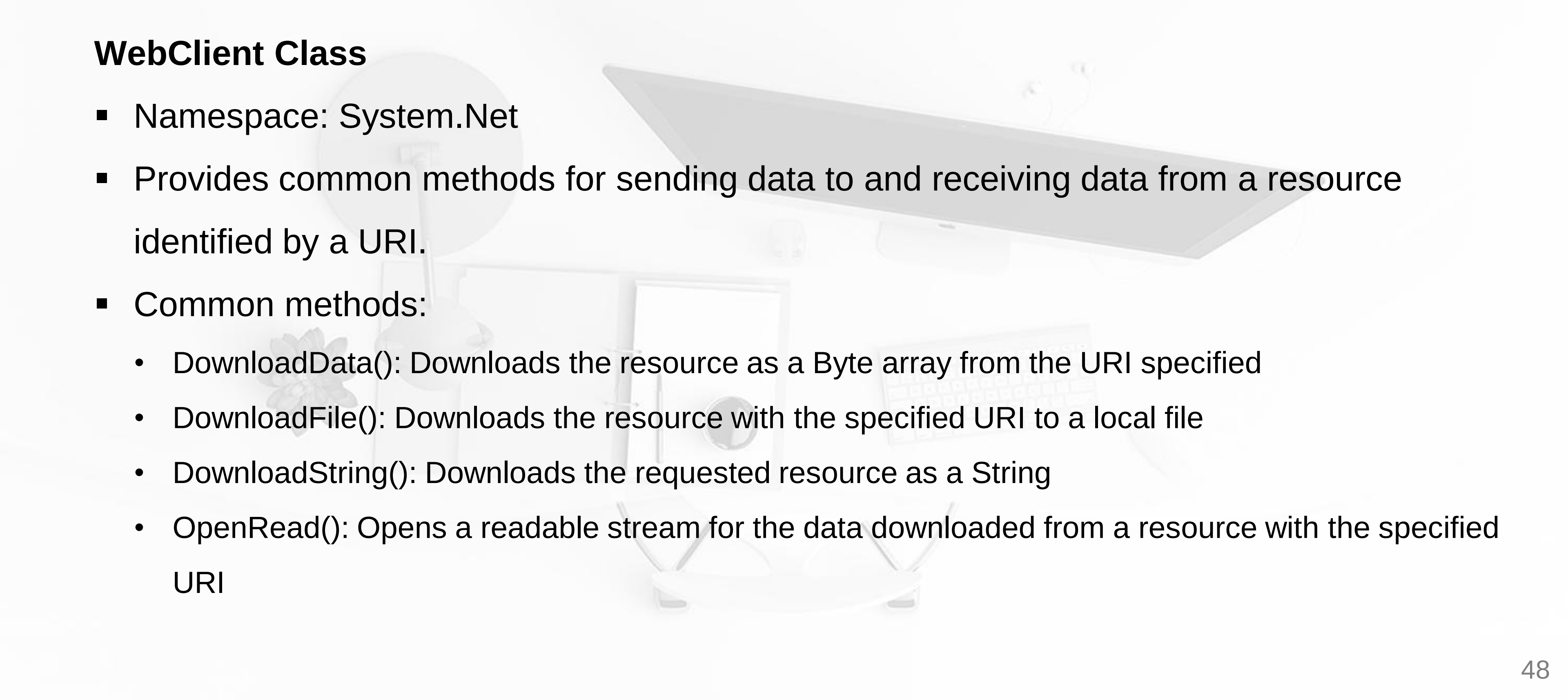
- **HttpListenerRequest Class**

  Namespace: System.Net
  Describes an incoming HTTP request to an HttpListener object.

- **HttpListenerResponse Class**

  Namespace: System.Net
  Represents a response to a request being handled by an HttpListener object.

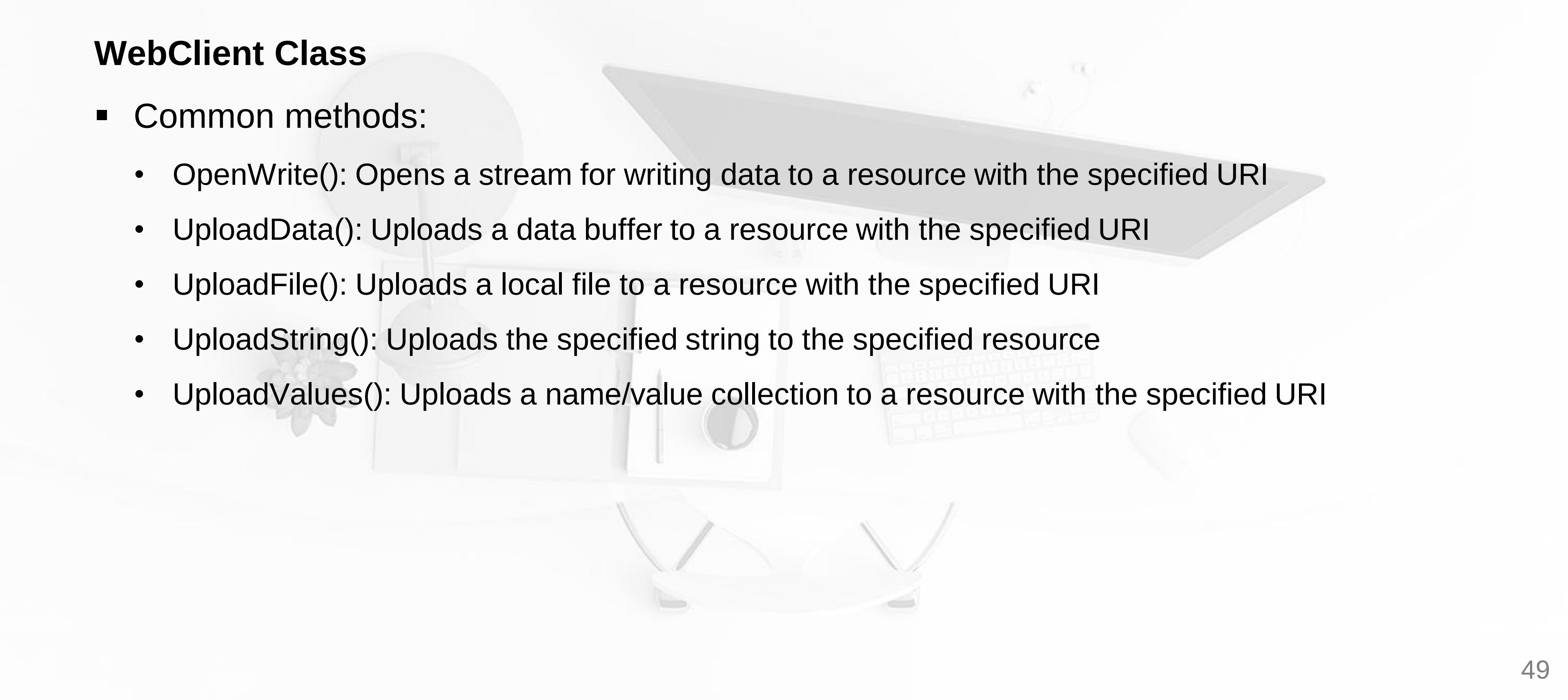
Library



WebClient Class

- Namespace: System.Net
- Provides common methods for sending data to and receiving data from a resource identified by a URI.
- Common methods:
  - DownloadData(): Downloads the resource as a Byte array from the URI specified
  - DownloadFile(): Downloads the resource with the specified URI to a local file
  - DownloadString(): Downloads the requested resource as a String
  - OpenRead(): Opens a readable stream for the data downloaded from a resource with the specified URI

Library



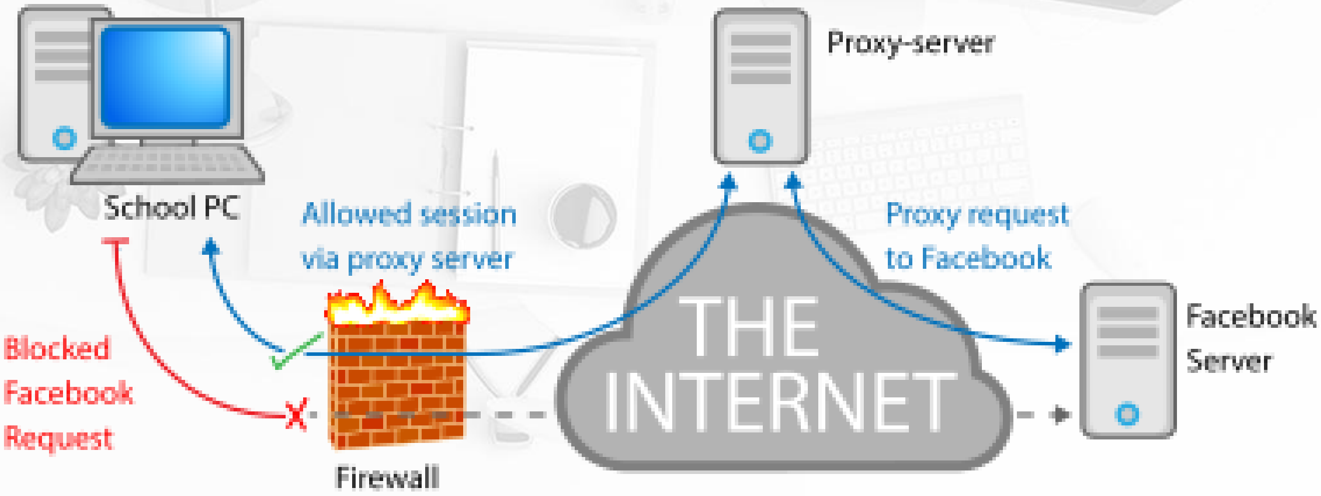
WebClient Class

- Common methods:
  - OpenWrite(): Opens a stream for writing data to a resource with the specified URI
  - UploadData(): Uploads a data buffer to a resource with the specified URI
  - UploadFile(): Uploads a local file to a resource with the specified URI
  - UploadString(): Uploads the specified string to the specified resource
  - UploadValues(): Uploads a name/value collection to a resource with the specified URI

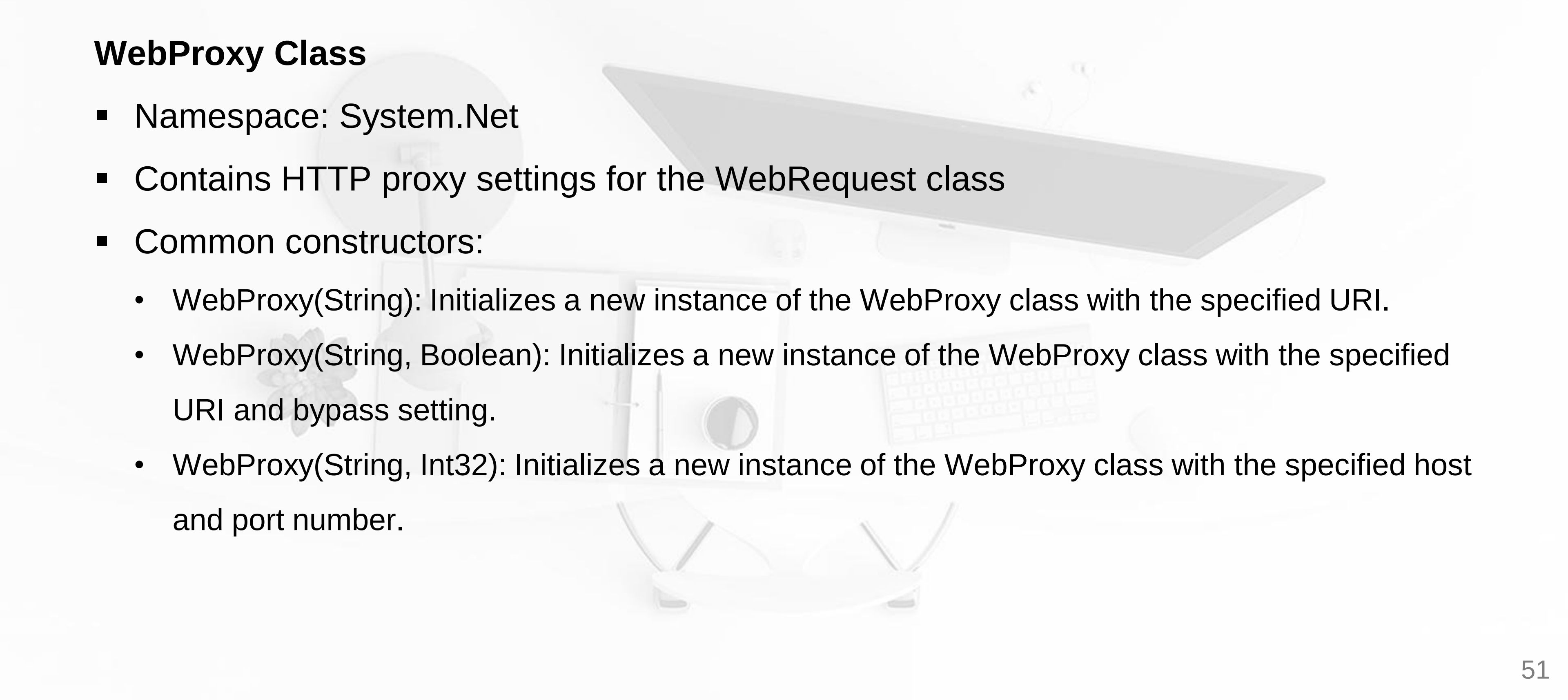
4. Proxy Server

A proxy server is a server application that acts as an intermediary between a client requesting a resource and the server providing that resource.

It improves privacy, security, and performance in the process.



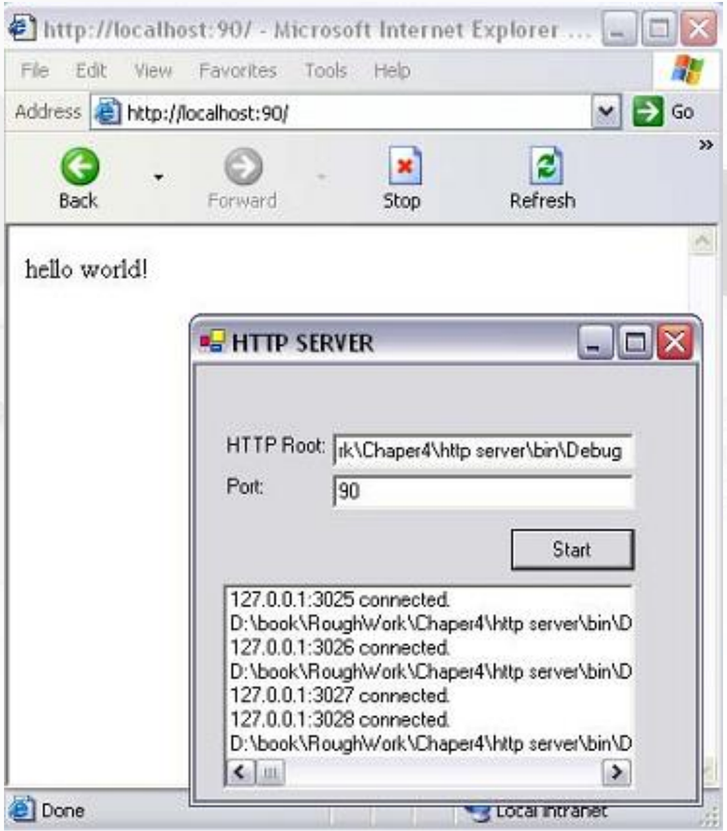
Library



WebProxy Class

- Namespace: System.Net
- Contains HTTP proxy settings for the WebRequest class
- Common constructors:
  - WebProxy(String): Initializes a new instance of the WebProxy class with the specified URI.
  - WebProxy(String, Boolean): Initializes a new instance of the WebProxy class with the specified URI and bypass setting.
  - WebProxy(String, Int32): Initializes a new instance of the WebProxy class with the specified host and port number.

Exercise


http://localhost:90/ - Microsoft Internet Explorer ...
File Edit View Favorites Tools Help
Address
Go
Back
Forward
Stop
Refresh
HTTP SERVER
Start
Done
Local intranet

http://localhost:90/

hello world!

HTTP Root: rk\Chaper4\http server\bin\Debug

Port: 90

```
127.0.0.1:3025 connected.
D:\book\RoughWork\Chaper4\http server\bin\D
127.0.0.1:3026 connected.
D:\book\RoughWork\Chaper4\http server\bin\D
127.0.0.1:3027 connected.
127.0.0.1:3028 connected.
D:\book\RoughWork\Chaper4\http server\bin\D
```

Q&A