Web Applications design and implementation

Dr. Abdelkarim Erradi

Dept. Computer Science & Engineering

QU

Outline

1. Web Applications
2. MVC-based Web applications
3. Web and HTTP

Web Applications

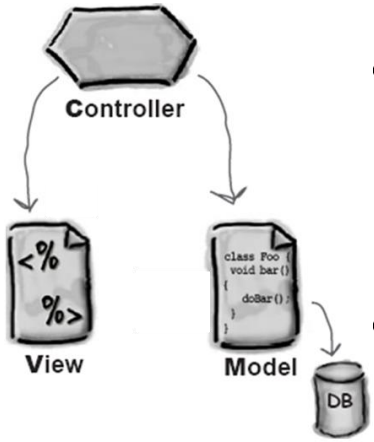
Web Applications

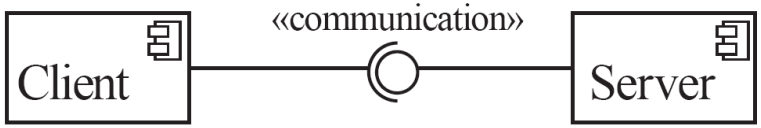
- **Web Applications are becoming critical applications** that **automate business processes** and **support the organization in achieving its goals**
- There are typically characterized by:
  - A large number of concurrent users. Hence they need to be scalable
  - Users often require acceptable response time
  - Mission critical hence they need to be secure, reliable and highly availabile

=> To achieve these quality attributes Web applications are typically **distributed** but appear to users as a single coherent system.

Design of Web Applications

- To achieve the required quality attributes, Web applications need to have a well defined software architecture that follows the best practices.
- **Software architecture = Components + Connectors**
- We will explore several architectures:
  - Client Server architecture





  - Model View Controller (MVC) architecture
    - Single Page Application (SPA)

Software Architecture Elements

- Software architecture (SA) describes: (1) how software is **decomposed and organized into components** (2) How these components **interact**
- *A* **component** is a *physical, replaceable* part of a system containing an implementation which conforms to a set of *interfaces*
- *A* **connector** is an mechanism that mediates communication among components.

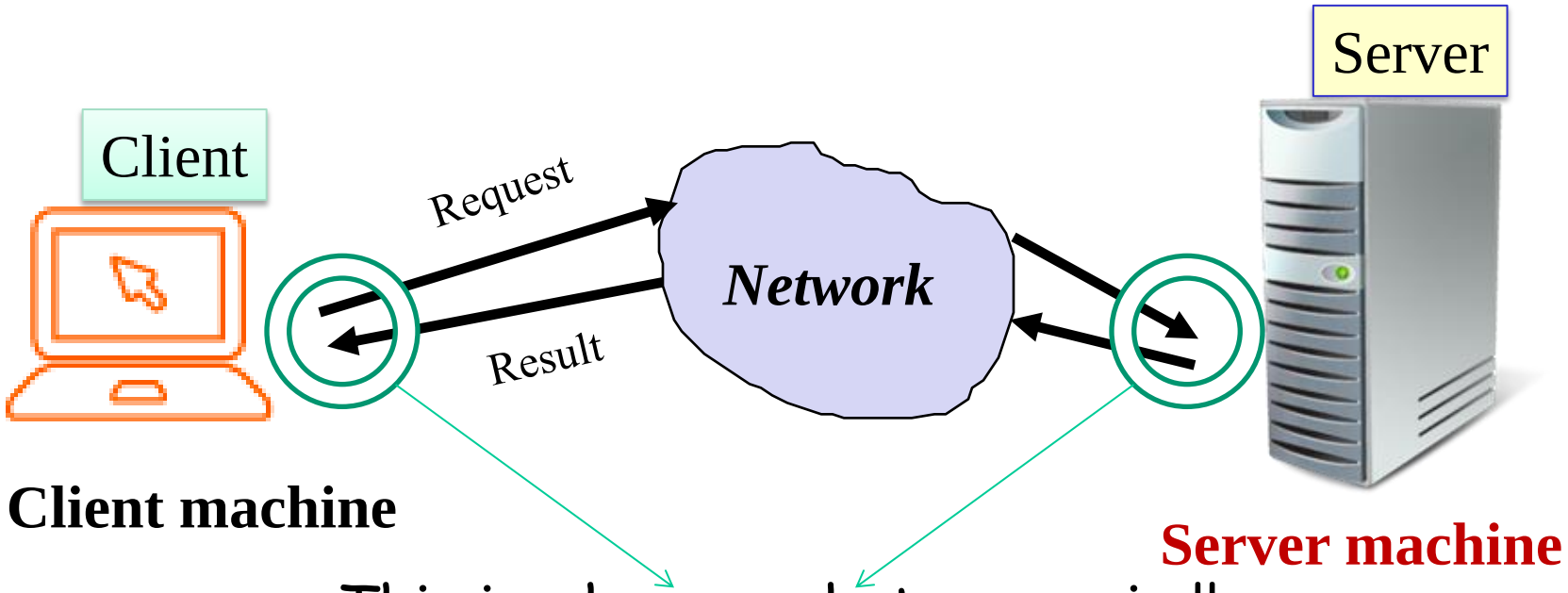
Connectors = how we can integrate components.

E.g.:

- Message Exchange (e.g., using Sockets)
- Remote Method calls (e.g., using Web services)

Client/Server Model

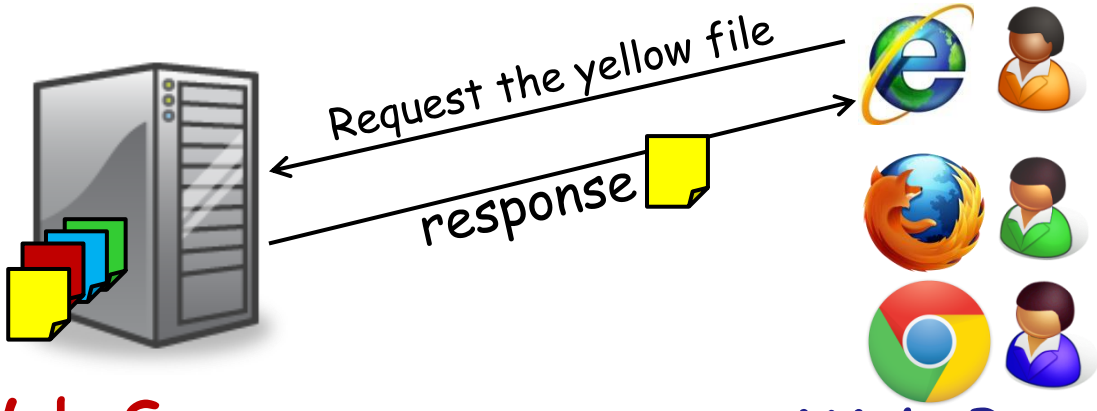
The Client/Server model decompose the application into a **Server** component(s) that **provides services** to a client. The **client** issues commands that the server executes



This is where sockets come in !!

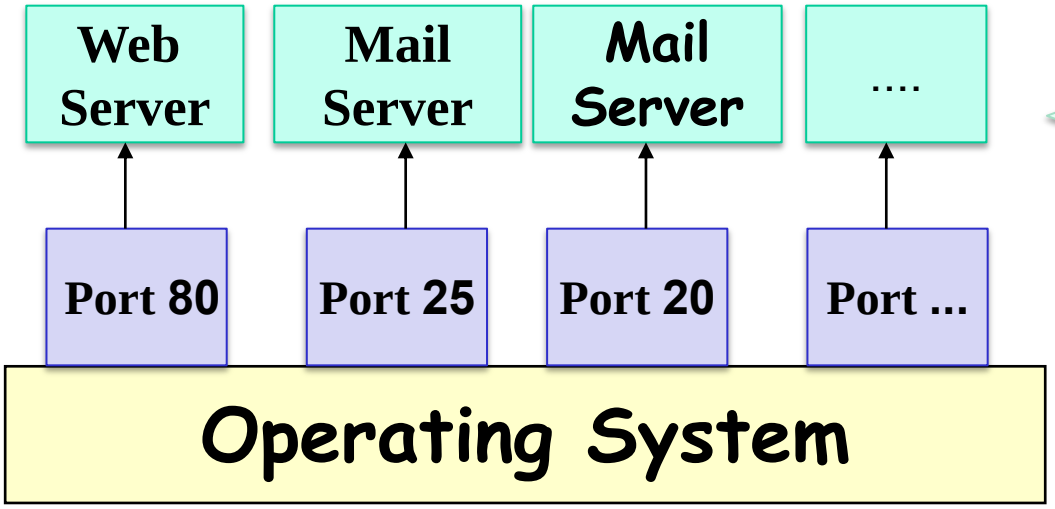
The **Client** communicate with the **Server** via the **network**. Programming the interactions between the client and the server uses **Sockets**

- **The server** is a program that offers services to **client** programs:
  - Waits for incoming requests
  - Reads the request
  - Performs the requested computation
  - Returns the results to the requester
- **Example Servers** include: Web server, Mail Server, File Server, Print Server, …etc.
- The interaction uses **request/response** pattern
  - e.g., via exchange of messages. The clients sends a request message, the server handles the request and returns a response message



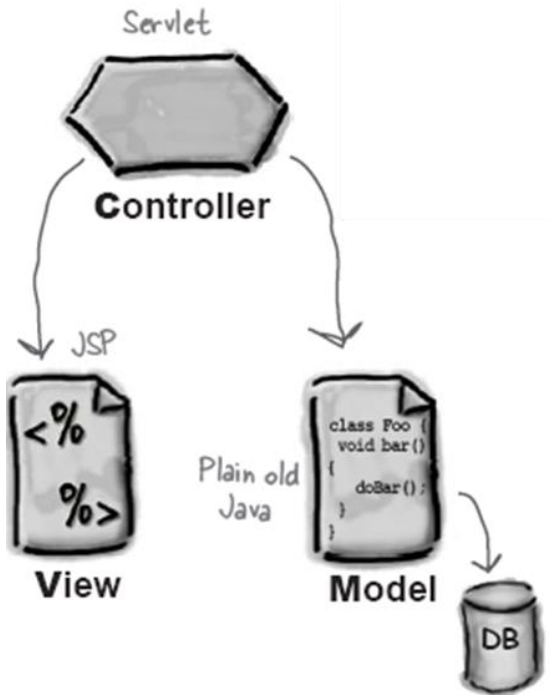
Role of Ports

- Server listens on a Port and waits for incoming requests
- A single machine can host many servers (e.g., Web server and Database server) each listening of a particular port
- The programmer selects a port between 0 and 65535



An example machine hosting different servers each listening on a particular port

MVC-based Web applications



MVC-based Web application

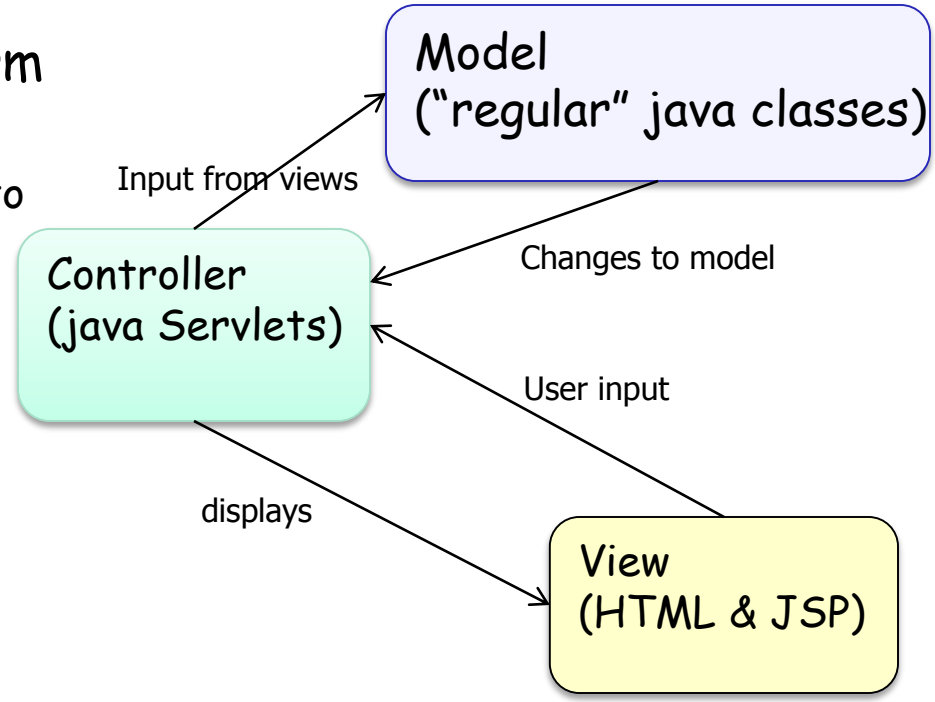
Controller –

- accepts incoming requests and user input
- instructs the model to perform actions based on that input
  - e.g. the controller adds an item to the user's shopping cart
- decides what view to display for output

Model – implements business logic and computation, and manages application's data

View – responsible for

- getting input from the user
- displaying output to the user



Advantages of MVC

- ***Separation of concerns***
  - Views, controller, and model are separate components. This allows modification and change in each component without significantly disturbing the other.
    - Computation is not intermixed with Presentation. Consequently, code is cleaner and easier to understand and change.
- **Flexibility**
  - The view component, which often needs changes and updates to keep the users continued interests, is separate
    - The UI can be completely changed without touching the model in any way
- **Reusability**
  - The same model can used by different views (e.g., Web view and mobile view)
- **Disadvantages:**
  - Heavily dependent on a framework and tools that support the MVC architecture (e.g. ASP.Net MVC, Spring MVC)

MVC is widely used and recommended particularly for interactive web-applications


WWW

Web and HTTP


GET /index.html
HTTP/1.1
HTTP/1.1 200 OK
"Welcome to our
Web site!"

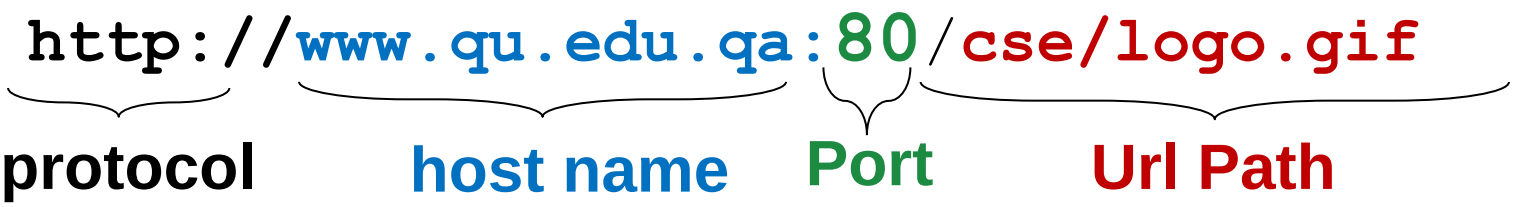
Key Concepts

- URLs
- HTTP Protocol
- Request-response model
- GET vs. POST methods
- HTTP Response Codes
- Internet Media Types

What is Web?

- Web = **global distributed system of interlinked hypertext documents accessed over the Internet using the HTTP protocol to serve billions of users worldwide**
  - Consists of set of resources located on different servers
    - HTML pages, images, videos and other resources
  - Resources have unique **URL** (Uniform Resource Locator) address
  - Accessed through standard protocols such as HTTP, HTTPS
- The Web has a Client/Server architecture:
  - **Web server** sends resources in response to requests (using HTTP protocol)
  - **Web browser** (client) requests, receives (using HTTP protocol) and displays Web resources

Uniform Resource Locator (URL)



- URL is a formatted string, consisting of:
  - **Protocol** for communicating with the server (e.g., `http`, `ftp`, `https`, …)
  - **Name of the server or IP** address plus optional port (e.g. `qu.edu.qa`, `localhost:8080`)
  - **Path of a resource** (e.g. **`/directory/index.php`**)
  - **Parameters** aka **Query String** (optional, e.g. `http://google.com/search`**`?q=qatar+university`**)

URL Encoding

- According RFC 1738, the characters allowed in URL are alphanumeric [0-9a-zA-Z] and the special characters $-_.+!*'()
- Unsafe characters should be encoded, commonly encoded values:

ASCII Character	URL-encoding
space	%20
!	%21
"	%22
#	%23
$	%24
%	%25
&	%26

e.g., http://google.com/search?q=qatar%20university

What's a protocol?

human protocols:

- السلام عليكم ورحمة الله وبركاته

=> و عليكم السلام ورحمة الله وبركاته

- "What's the time?"

=> 8am

… specific messages sent

… specific actions taken when a message received, and a response message sent

network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

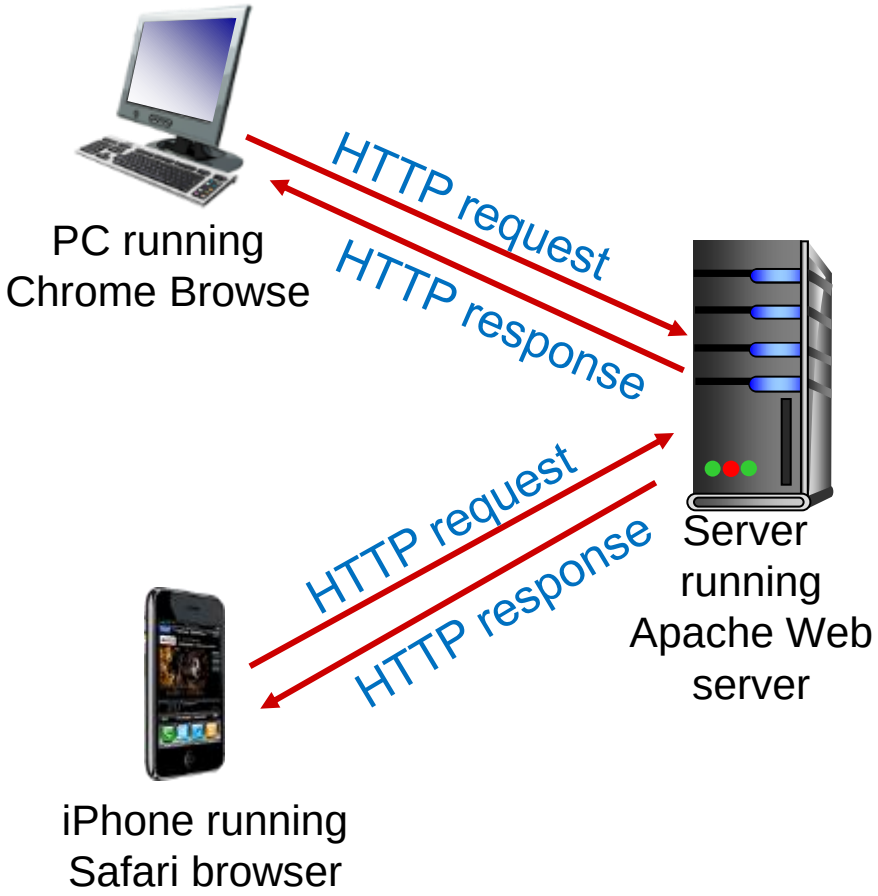
protocols define format, order of messages sent and received among network entities, and actions taken on message receipt

HTTP overview

Hypertext Transfer Protocol (HTTP)

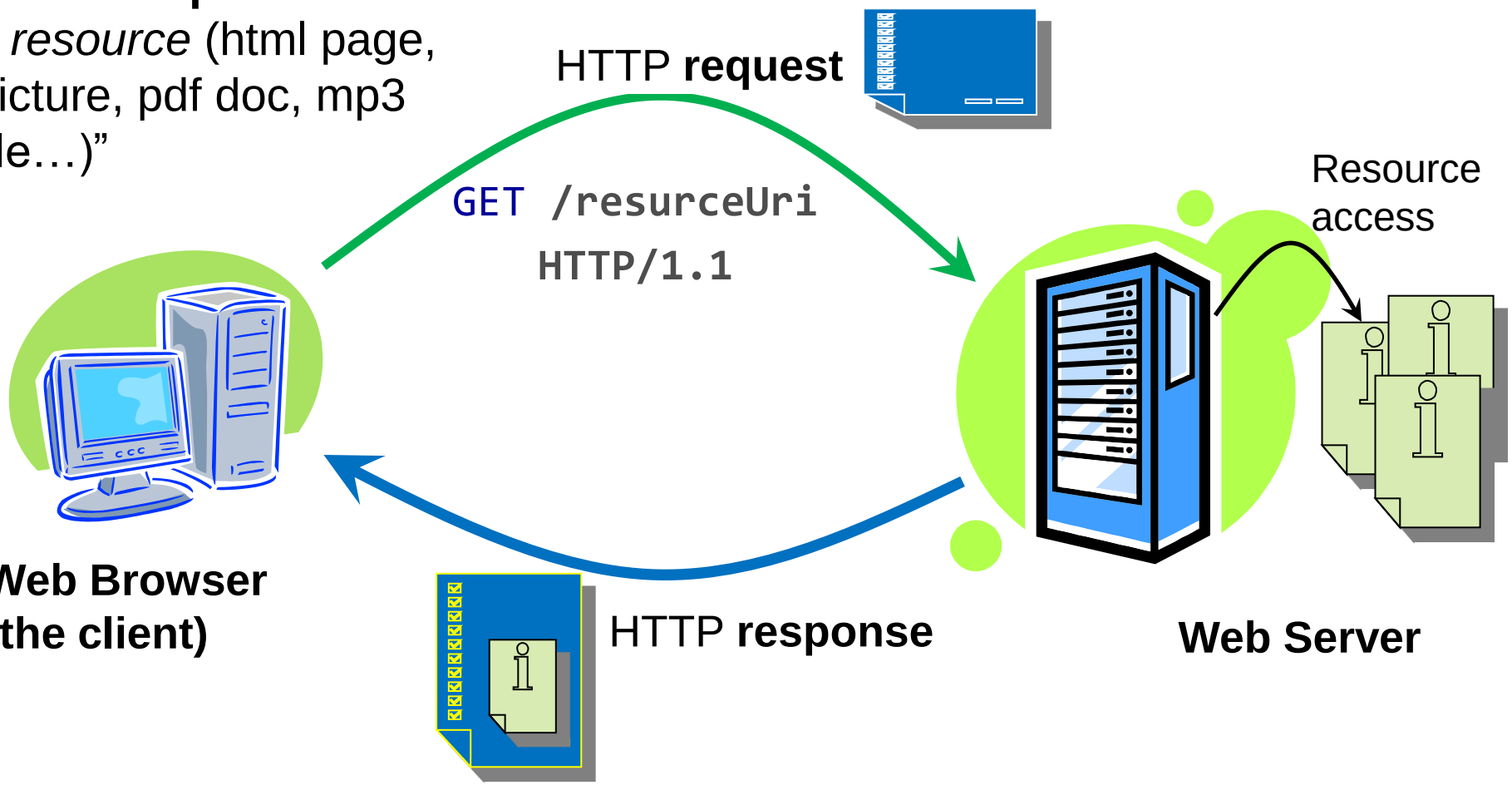
- Client-server protocol for transferring Web resources (HTML files, images, etc.)

- Important properties of HTTP
  - Request-response model
  - Text-based format
  - Relies on a unique resource URLs
  - Provides resource **metadata** (e.g. content-type, encoding)



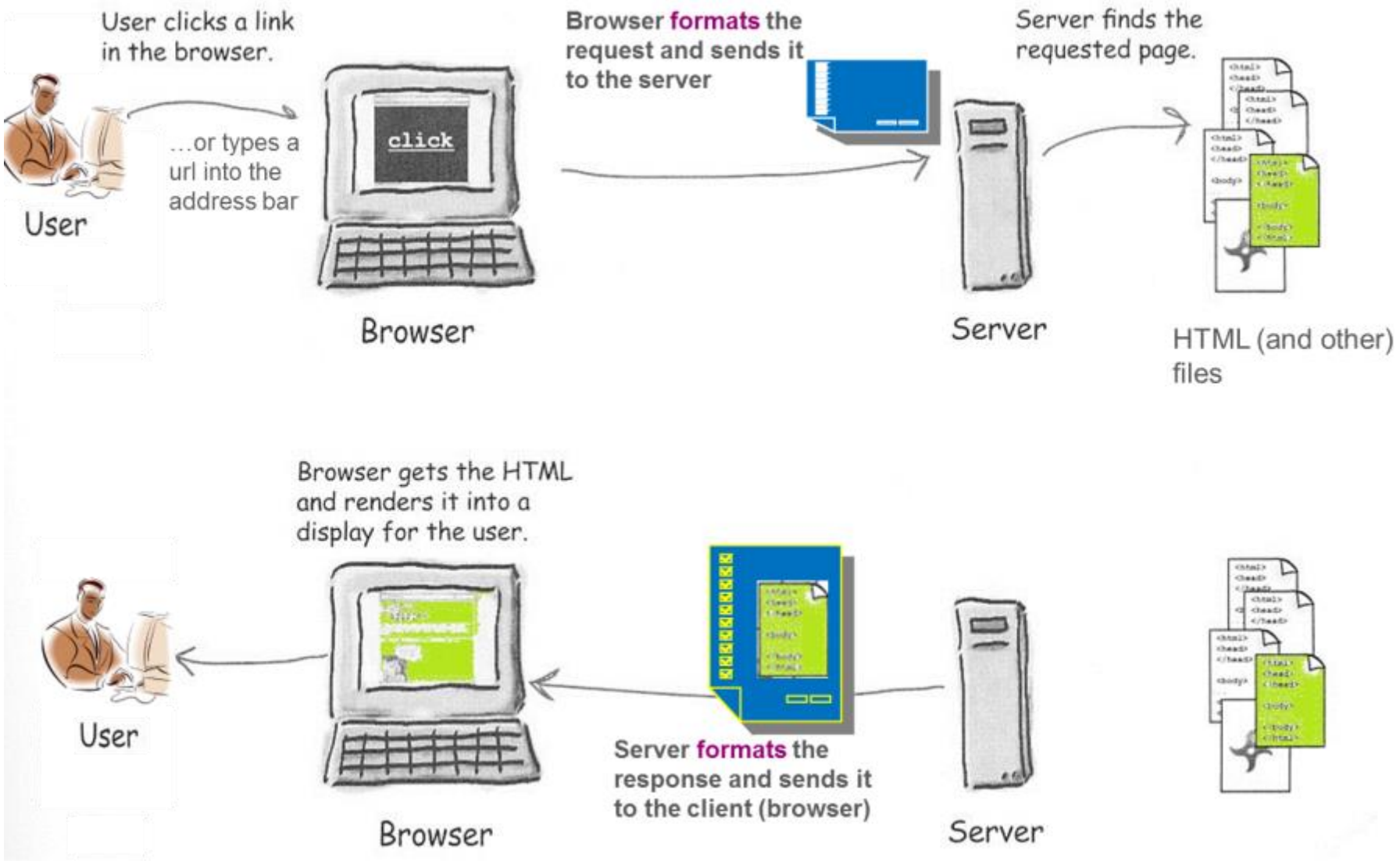
Web uses Request/Response interaction model
HTTP is the *message protocol* of the Web

Client Request: "I need a *resource* (html page, picture, pdf doc, mp3 file…)"



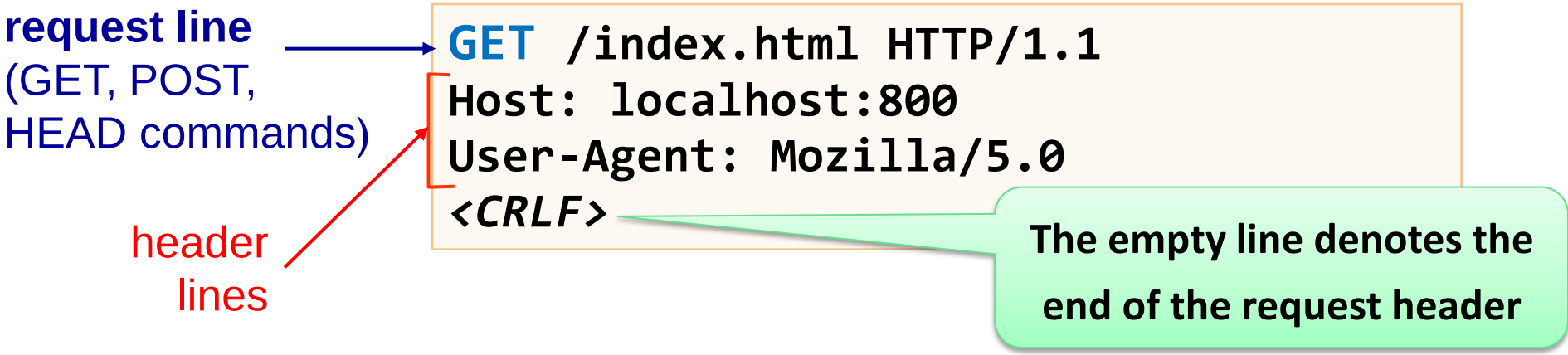
Server Response: "Here you go!"

The sequence for retrieving a static web page

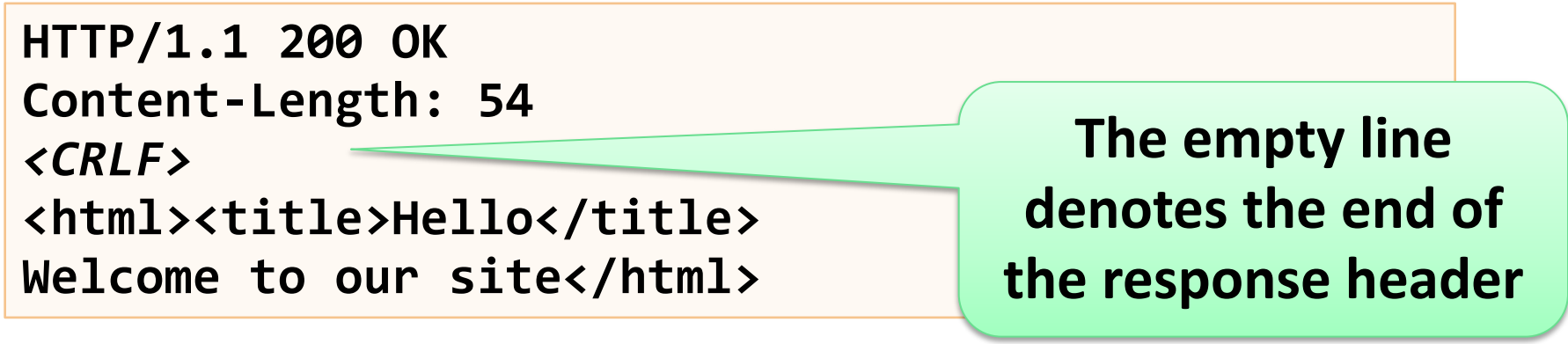


Request and Response Examples

- **HTTP request:**



- **HTTP response:**



HTTP Request Message

- Request message sent by a client consists of
  - **Request line** – request method (GET, POST, HEAD, ...), resource URI, and protocol version
  - **Request headers** – additional parameters
  - **Body** – optional data
    - e.g. posted form data, files, etc.

```
<request method> <URI> <HTTP version>
<headers>
<empty line>
<body>
```

HTTP Request Methods

- **GET**
  - **Retrieve a resource** (could be static resource such as an image or a dynamically generated resource)
  - Input is appended to the request URL E.g.,

    `http://Google.com/?q=Qatar`

- **POST**
  - **Update a resource**
  - Web pages often include form input. Input is submitted to server in the **message body.** E.g.,

POST /calc HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
num1=20&operation=*&num2=10

HTTP Response Message

- Response message sent by the server
  - **Status line** – protocol version, status code, status phrase
  - **Response headers** – provide metadata such as the Content-Type
  - **Body** – the contents of the response (i.e., the requested resource)

```
<HTTP version> <status code> <status text>
<headers>
<empty line>
<response body>
```

HTTP Response – Example

status line (protocol status code status text)

Try it out and see HTTP in action using **HttpFox**

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: QU Web Server
Content-Length: 131
<CRLF>
<html>
  <head><title>Calculator</title></head>
  <body>20 * 10 = 200
      <br><br>
      <a href='/calc'>Calculator</a>
  </body>
</html>
```

HTTP response headers

The empty line denotes the end of the response header

Response body. e.g., requested HTML file

HTTP Response Codes

- Status code appears in 1st line in response message
- HTTP response code classes
  - `2xx`: success (e.g., "`200 OK`")
  - `3xx`: redirection (e.g., "`302 Found`")
    - "`302 Found`" is used for redirecting the Web browser to another URL
  - `4xx:` client error (e.g., "`404 Not Found`")
  - `5xx`: server error (e.g., "`503 Service Unavailable`")

Popular Status Codes

Code	Reason	Description
200	OK	Success!
301	Moved Permanently	Resource moved, don't check here again
302	Moved Temporarily	Resource moved, but check here again
304	Not Modified	Resource hasn't changed since last retrieval
400	Bad Request	Bad syntax?
401	Unauthorized	Client might need to authenticate
403	Forbidden	Refused access
404	Not found	Resource doesn't exist
500	Internal Server Error	Something went wrong during processing
503	Service Unavailable	Server will not service the request

Browser Redirection

- HTTP browser redirection example
  - HTTP GET requesting a moved URL:

(Request-Line)	GET /qu HTTP/1.1
Host	localhost:800
User-Agent	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

  - The HTTP response says that the browser should request another URL:

(Status-Line)	HTTP/1.1 301 Moved Permanently
Location	http://qu.edu.qa

Common Internet Media Types

- The **Content-Type** header describes the media type contained in the body of HTTP message
- Commonly used media types (**type**/subtype):

Type/Subtype	Description
application/atom+xml	Atom feed
application/json	JSON data
image/gif	GIF image
image/png	PNG image
video/mp4	MP4 video
text/xml	XML
text/html	HTML
text/plain	Just text

Full list @ http://en.wikipedia.org/wiki/MIME_type

Typical server steps to process an HTTP Request

- Parse the HTTP request (i.e., convert a textual representation of the request into an object)
- Generate a response either static one by reading a file or a dynamic response
  - Dynamic response could be either generated programmatically from scratch or it could be generated by filling-up a page template read from a file
- Send the response to the client including:
  - Response headers
  - Response body

HTTP Developer Tools

- **Chrome / Firefox**

  Ctrl-Shift-I → Developer Tools

- **HTTP Fox** plug-in for Firefox

  – Intercepts HTTP traffic

  – https://addons.mozilla.org/en-US/firefox/addon/httpfox/

- **Firebug** plug-in for Firefox

  – The ultimate tool for monitoring, editing and debugging HTTP, HTML, CSS, JavaScript, etc.

  – http://getfirebug.com/

- **Fiddler** – HTTP proxy

  – Intercepts and analyzes HTTP traffic

  – http://www.telerik.com/fiddler

References

- **HTTP Made Really Easy**
  http://www.jmarshall.com/easy/http/

- **HTTP 1.1 Specification (RFC 2616)**
  http://www.w3.org/Protocols/rfc2616/rfc2616.html