

Web & Web applications

DBW 2017

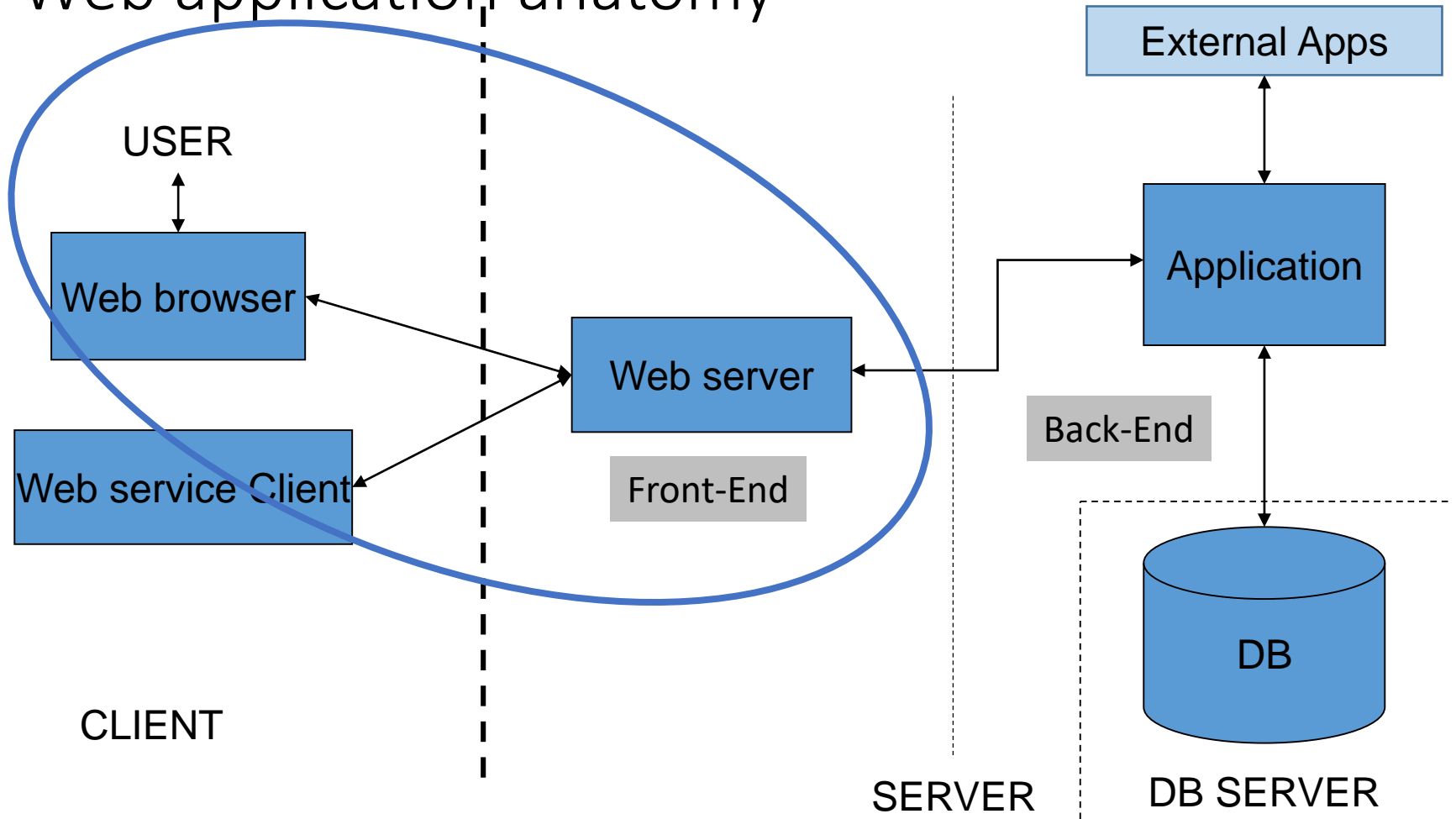
Outline

- Web basics
 - HTTP servers and browsers
 - Languages
 - Software
- Concept and types of Web applications. Web services
- CGI Protocol.
- Time issues
- Personalization. Cookies & session persistence
- Languages involved

Reminder: Why Web-based?

- User interface is provided by standard Web servers and browsers (no need to develop friendly interfaces)
- No need to install client software (almost)
- Communications use standard protocols (HTTP, SMTP, SOAP) and languages (HTML, XML)
- Several programming languages available

Web application anatomy



Web (HTTP) Servers

- Computer applications that listens to a TCP port (80 typically), and understands HTTP requests.
- Information served are text or binary files (*resources*) stored locally in the server.
- HTTP servers that implements the appropriate protocol, can run **server-side applications** according to the request.
- Example [SimpleHTTPserver](#)

Web (HTTP) clients (browsers, ...)

- Applications making requests to server at a given TCP port (typically 80) using HTTP protocol
- Simple browsers requests for files (using HTTP) in a similar way to FTP. Resources are identified by URL (wget for example)
- Normal browsers “understand” the contents of the obtained files and combine in graphical output information from one or more servers according to a given language (usually HTML)
- Most **browsers can execute applications** (client-side) obtained from the information server (scripts, Active-X, applets, ...)

Languages involved

- **HTML: Contents management language**
 - Defines contents and structure of the page, includes the necessary links to all elements
 - Tag formatted language (...<p>Some text</p>...)
- **CSS: Formatting language**
 - Defines how the contents is represented in the user browsers
 - `P {font-family:Times; font-family: 10pt; display:block; background-color:black}`
- **Data interchange formats**
 - **XML: Most traditionally used by web applications**
 - Same structure as HTML, but with no fixed tags
 - Requires XML-schema to specify tags and check coherence

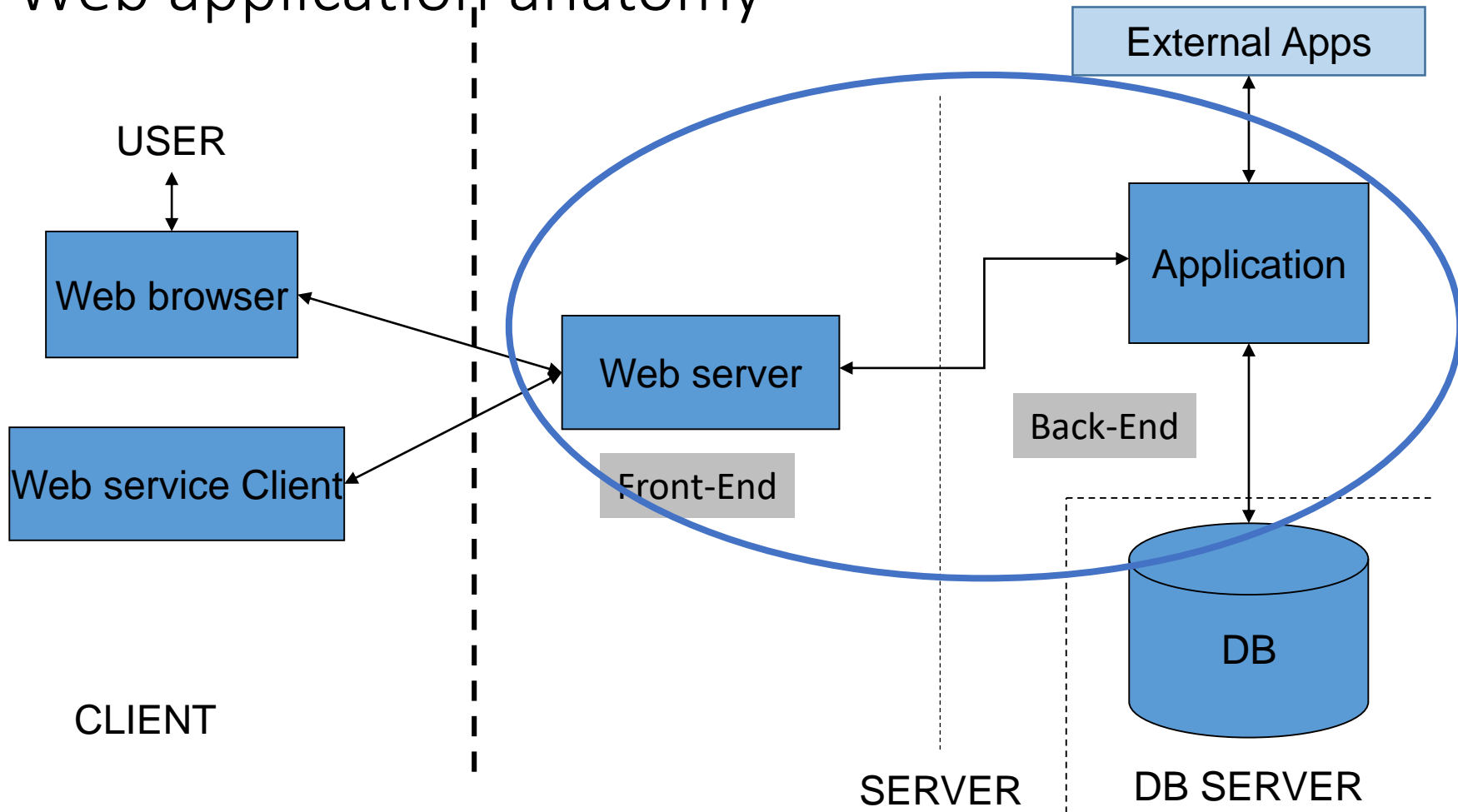
```
<Course id="DBW">  
    <Acronym>DBW</Acronym>  
    <Title>Databases and Web applications</Title>  
</Course>
```
 - **JSON: Data interchange format replacing XML**
 - Natively understood by Javascript, and of increasing popularity

```
Course: {Acronym: 'DBW', Title: 'Databases and Web applications'}
```

Software to build Web pages

- A simple text editor is enough (Notepad, vi, ...)
- Syntax checking editors are more useful (nedit, NetBeans, ...)
- WYSWYG editors are common (Dreamweaver, Openoffice...). However, they MUST allow to check HTML manually!
- Content Manager Systems (CMSs)
 - Integrated environments to build web sites, general or specialized
 - Can include some useful functionality (user management, email, ...)
 - Very useful to build static sites, but difficult to include applications
 - However, web structure and layout made by the CMS can be used
- Drupal, Joomla, Wordpress, Bookstrap, ...

Web application anatomy



Definition & types

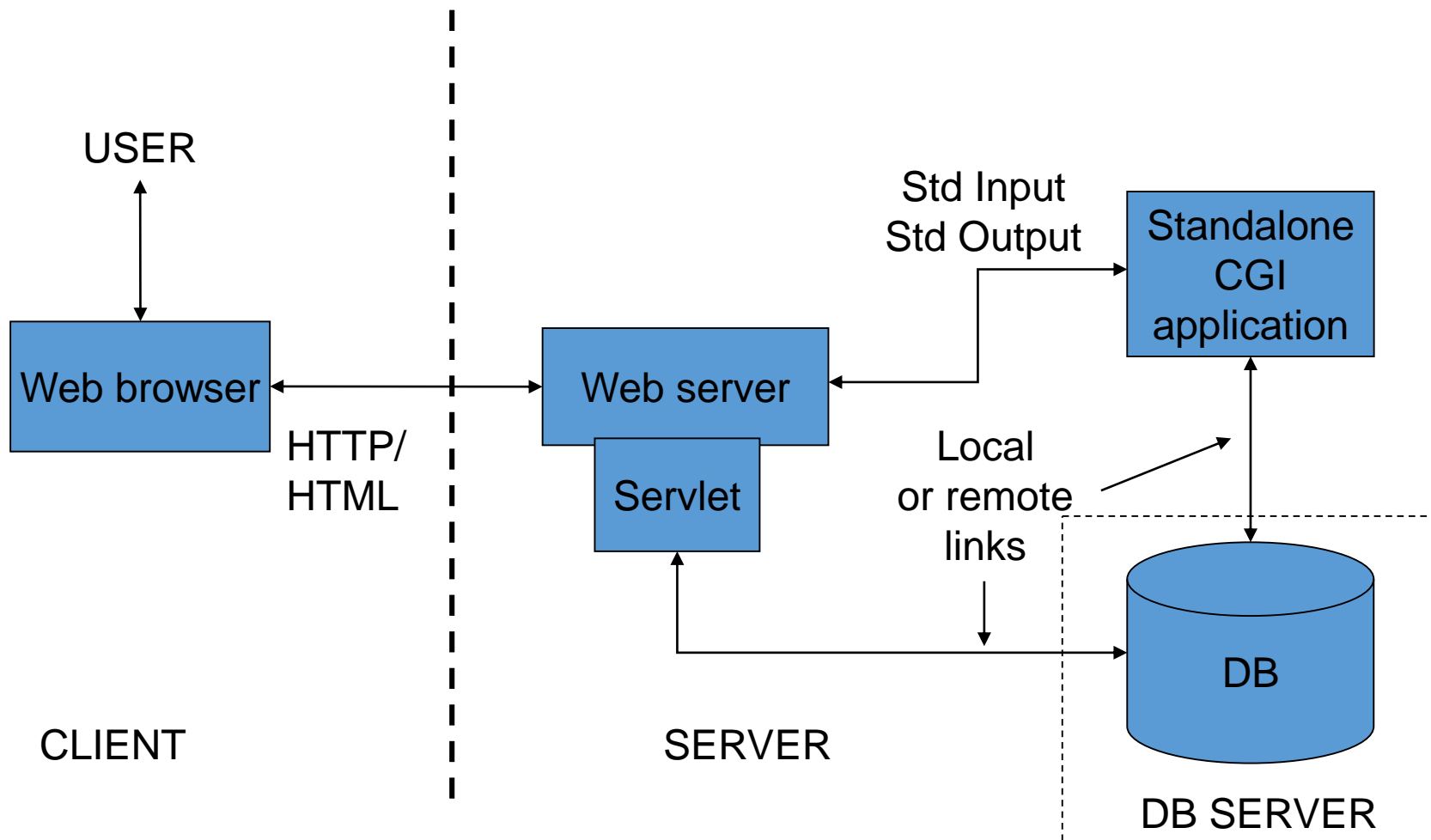
- A Web application is a dynamic extension of a Web server.
 - Adapts to user input
 - Can serve non-static information (generated in real-time)
 - Uses standard protocols (HTTP, SMTP)
 - Users can interact with the application using Web browsers
- Presentation-oriented
 - Generates dynamic Web pages (HTML/HTTP) responding to user queries
 - Usual way to provide bioinformatics results
- Service-oriented
 - Interacts with other applications (XML/SOAP)
 - Allows to build automatic workflows

Client side

- Application must be compatible with standard web browsers
 - HTTP protocol: GET, POST
- Input comes from URL's or HTML forms
- Output must be standard HTML (XML, JSON), CSS, Javascript
- Output may invoke other programs (plug-ins) though MIME
 - Almost obsolete, except Flash
- HTML v5 include a variety of native functionalities
 - Audio/video, SVG graphics, MathML, Geolocalization, parallel process, ...
- Modern browsers are able to run Java applets, and Javascript
 - Java applets/Javascript are fully qualified applications, served as static files, and run in the browser
 - Javascript is behind most dynamic behaviour of modern web sites (responsiveness).
 - Asynchronous interaction with server (new request do not require reload)
 - JsMol, JQuery, Angular.js,...

Server side

- Application is invoked by web server
 - External application (CGI)
 - Executable running in the server machine.
Can be written in **any** language.
 - Get input from standard input and writes in the standard output. Web server redirects both.
 - Server embedded (Servlet).
 - Web server is able to execute the application as a child process
 - Special languages: Scripting: PHP, Python, ASP, JSP, .NET, Servlets (Java, Javascript)
 - Java applications require special servers



CGI Protocol

- Common Gateway Interface (CGI)
 - Formal interface between Web server and external applications
 - CGI interface provides
 - Environment variables including all relevant information from the browser-server conversation
 - Includes GET queries
 - POST input data, as standard input
 - Redirection of application standard output & error to Web stream.
- External applications
 - Read Input information from Environment variables, and standard input
 - Provides results and error as standard output
 - Are executed by the operative system as usual command lines executables
 - For security reasons they must be in a special directory (cgi-bin) unless the server is configured otherwise (MIME type cgi).
- Servlets
 - Are executed as web server's subprocesses
 - Still use input and output standard and CGI variables, but data is processed

Some CGI variables

- SERVER_SOFTWARE
- SERVER_NAME
- GATEWAY_INTERFACE
- SERVER_PROTOCOL
- SERVER_PORT
- REQUEST_METHOD ("GET", "HEAD", "POST")
- PATH_INFO
- PATH_TRANSLATED
- SCRIPT_NAME
- QUERY_STRING
- REMOTE_HOST
- REMOTE_ADDR
- CONTENT_TYPE
- CONTENT_LENGTH (with POST)
- HTTP_ACCEPT
- HTTP_USER_AGENT
- HTTP_ HTTP headers

Basic Input and Output

- Input from HTTP “GET” appears as QUERY_STRING
- Input from POST appears on standard input

- Output is a document with a MIME Type header

```
Content-type: text/html
```

```
<blank>
```

```
<html><body>
```

```
...
```

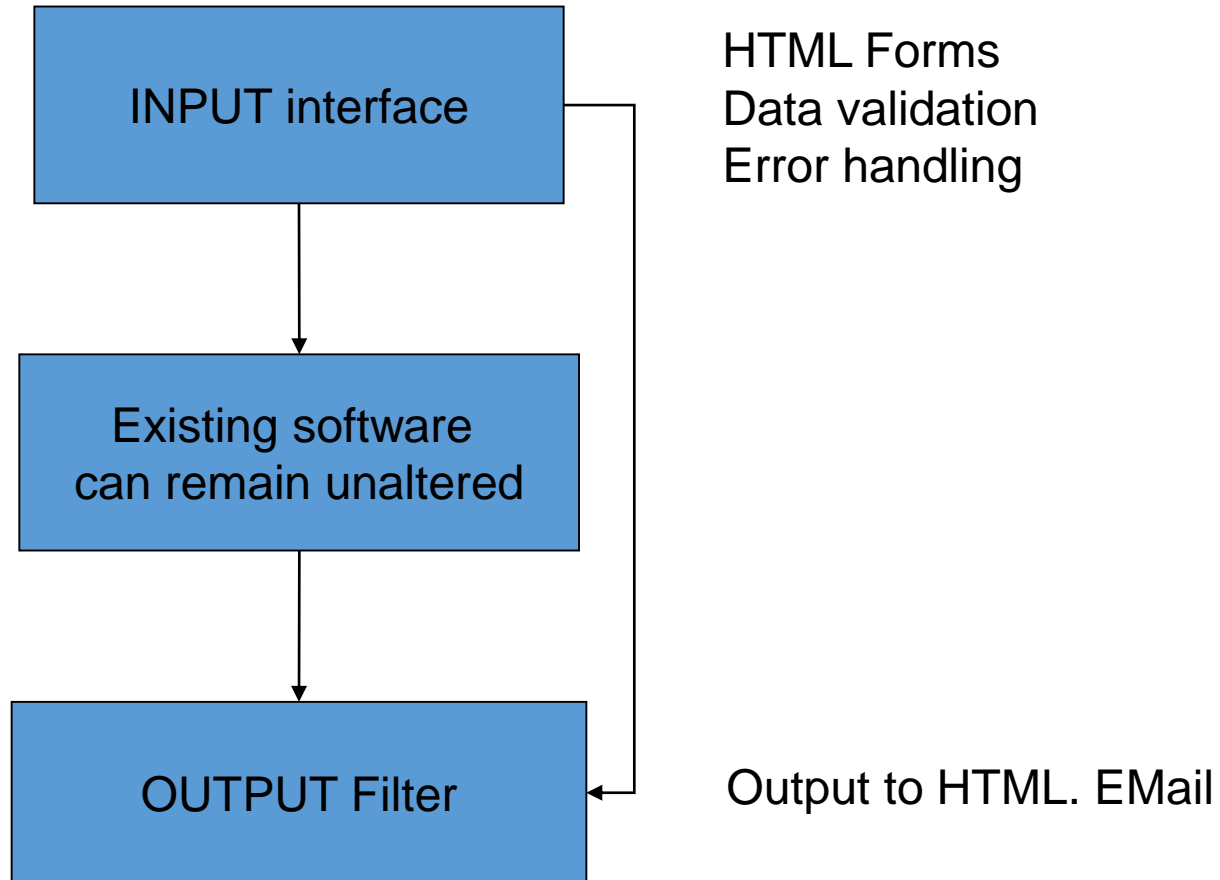
```
</body></html>
```

- Or a redirection

```
Location: new URL
```

```
<blank>
```


The simplest application



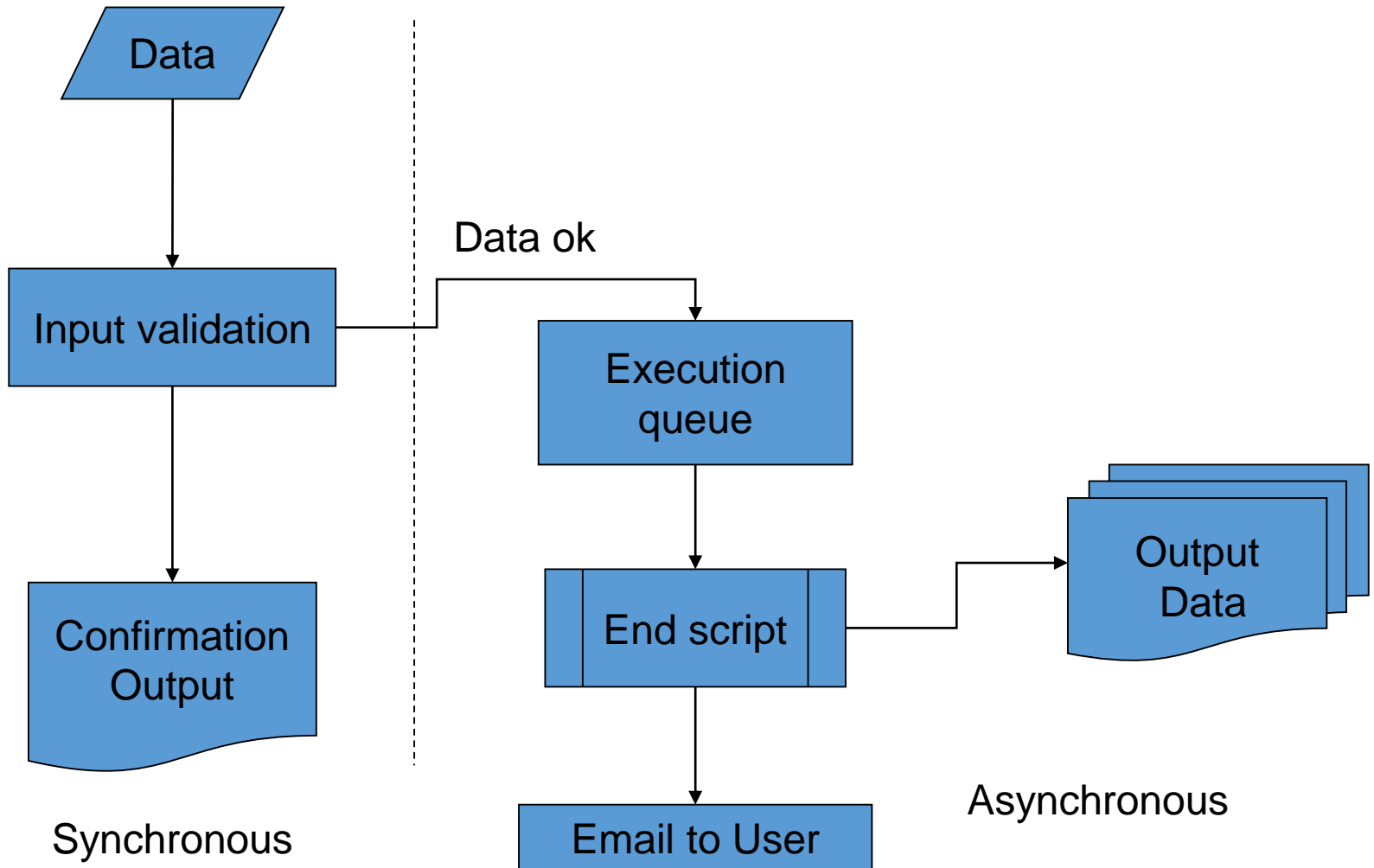
Servlet languages

- Depends on the server software used
 - Java application servers (Tomcat, Jboss, ...): JSP, Java
 - Microsoft servers: ASP (old), Aspx, .NET
 - Standard servers (Apache): PHP, node.js
 - With the appropriate extensions: Perl, Python
- Most popular:
 - PHP/Apache, Java/Tomcat
- Growing popularity
 - Python, Javascript (Node.js)

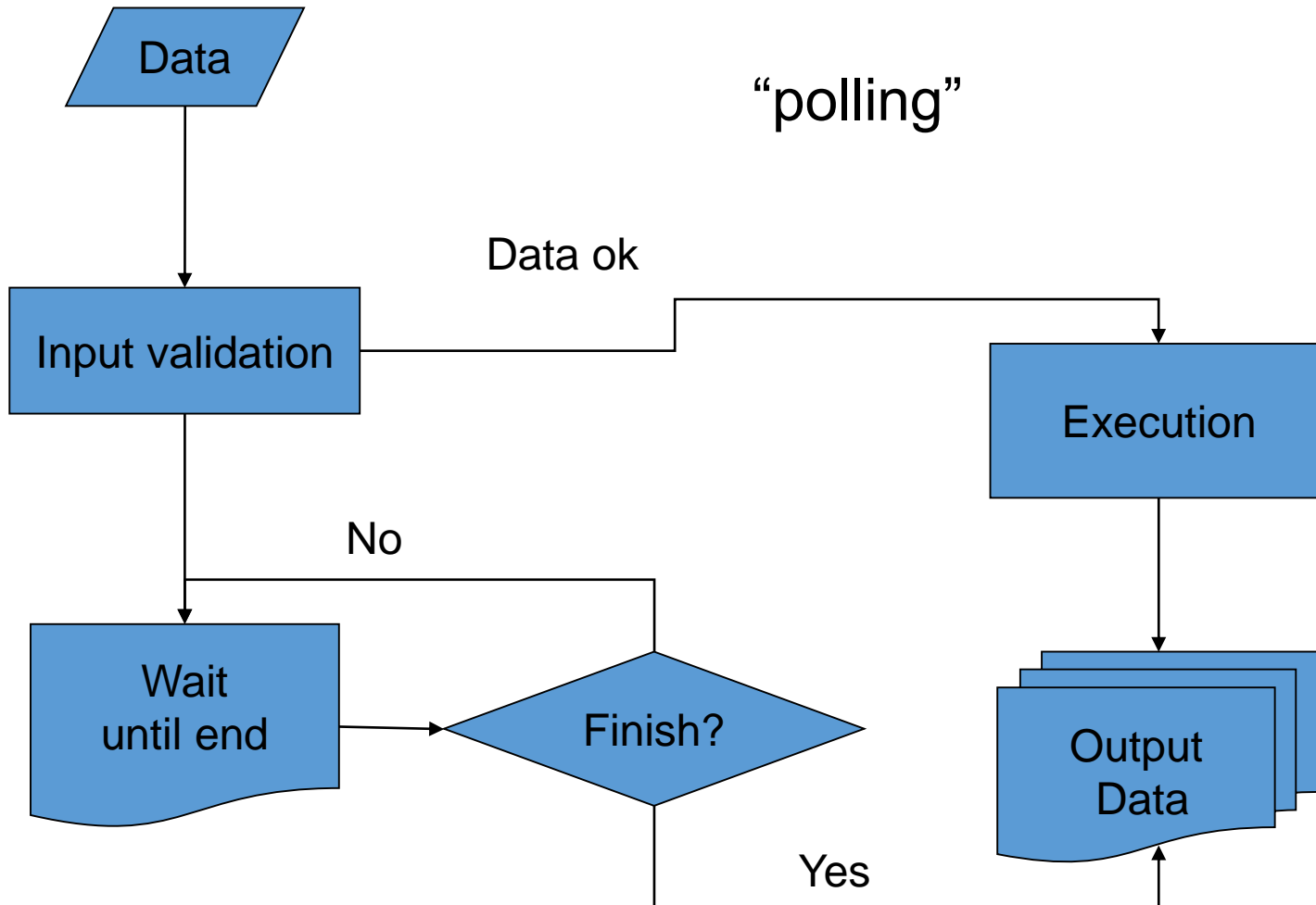
Known issues

- Time issues
 - Web users require “instant” responses
 - Most web browsers have short “timeouts”
 - Application that last more that 1-2 mins must be asynchronous
- Persistence, and User recognition
 - HTTP protocol is not persistent: Connection closes short time after the servers answers
 - Application need to recognize returning users
 - Authentication (user only must write the login/password once)
 - Keep personal preferences, and private data
 - Grant access to given resources according to previous requests
 - Avoid to request known data more than once
 - Avoid “reloads”

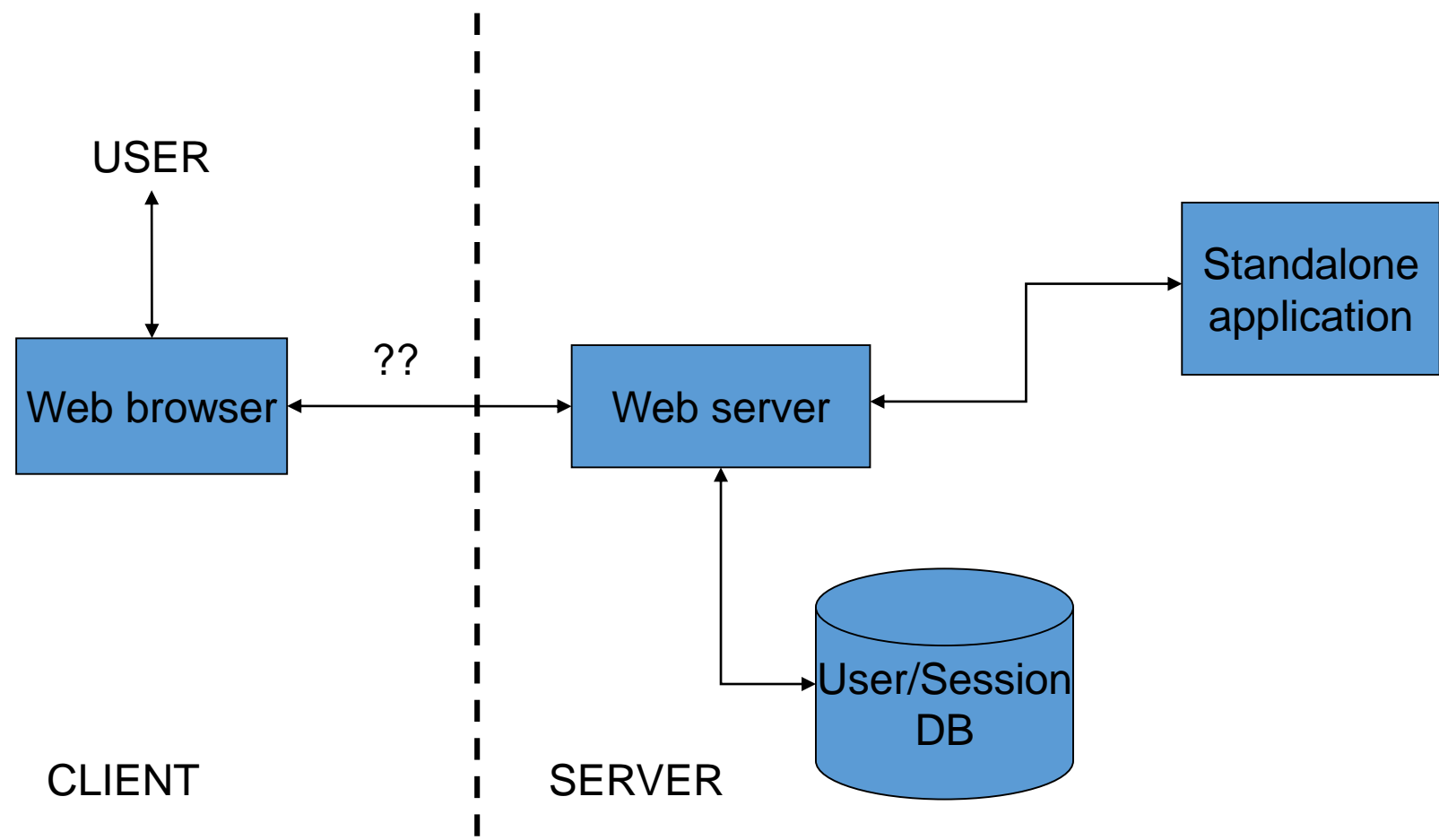
Time issues. Usual strategies



Time issues. Usual strategies



Personalization



A DB stores history of user connections and activities

Authentication schemes

- Server based
 - Based on unix-like passwd files (login / passwd)
 - Protects folders and sub-folders (.htaccess files)
 - Identity via CGI variable (REMOTE_USER)
 - May require access to server configuration
 - Persistence via HTTP
 - Environment managed by local DB
- Application based
 - Do not require access to server configuration
 - Authentication and environment managed by a local DB
 - Full control from the application (login / passwd, SSL Keys, ...)
 - Persistence via Cookies or language specific constructs (PHPSessionID / Session)
- Third party authentication
 - Authentication is done by public servers (Google, openID, ...), or other apps (eGroupWare, Drupal, ...) / Session, LDAP

User identification: cookies

- Small amount of text information stored by the server in the users' web browser.
- Do not require user/password (user do not need to be aware of)
- Limited to 4Kb
- Cookies can last for a single session or till a specified date
- Cookies can be used to avoid password request
- Cookies do not identify persons but browsers!!

Cookies: details

- Cookies are key / value pairs
- The normal identification cookie is a unique ID generated by the server
- Origin: server URL. Browsers send back cookies to the servers that created them (no other servers can get the data)
- Expiration date

Web application layout hints

- Static contents (text, images, etc.) stored as normal web resources
- Dynamic pages managed by servlet scripts
 - No general rule, depends on language, and programming style
 - The easy way: Each different screen is managed by a specific script.
- Global variables
 - Each script acts in a separated HTTP transaction!
 - All scripts should load the same global environment, usually included from a single file
- Protected/public data
 - Protected data should be stored outside of the web directory tree, and be accessed only programmatically

Web application layout hints

- Temporary data
 - Can be stored anywhere
 - Most language provide automatic temporary directories and file names.
 - Should be deleted after use!!
- Beware of multiple concurrent users
 - Use request specific file names for temporary data and results
 - Use user based directories
 - Think in a queueing system for lengthy operations
- Collect statistics of use