Web-Technologies

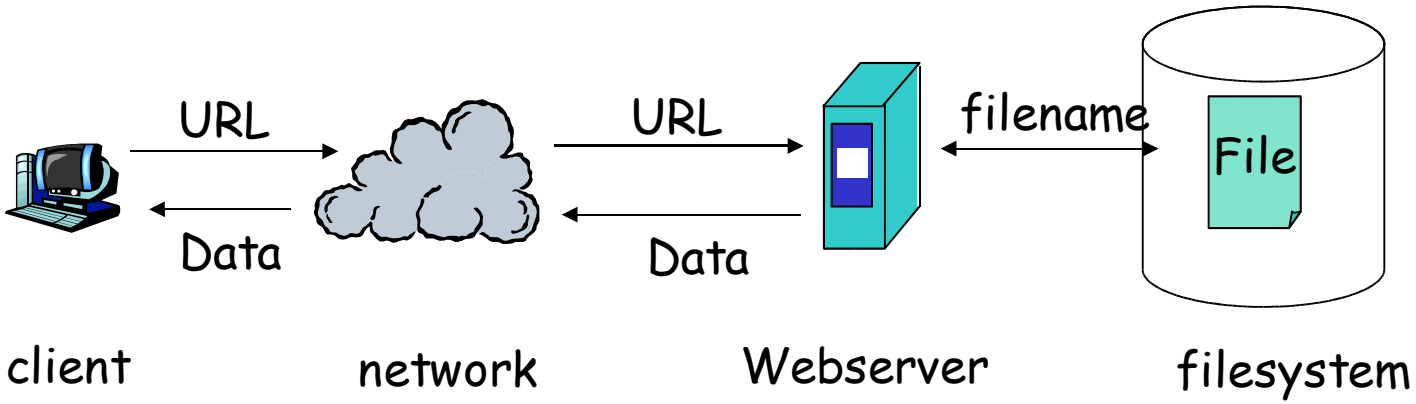
- Chapters
  - Server-Side Programming: Methods for creating dynamic content
  - Web-Content-Management
  - Excurse: Server Apache
  - Client-Side Programming (Next Lesson)
  - Web-Services (Next Lesson)
  - Search engines and Spiders (Next Lesson)

Server-Side Programming 1

- Introduction
  - Server-Side Programming:
    - User (Browser) requests a dynamic document
    - Additional information is send to the server using GET or POST
    - Server parses the user-request and creates the document by internal procedures
    - On success, the document is send back to the user
  - Several methods for servers to create a document:
    - CGI
    - SSI
    - PHP
    - ASP and others

Server-Side Programming 2

- To Recall: Accessing a static page



  - Typical access: URL = Protocol + Domainname or IP (+ Port) + Filename within the DocumentRoot
  - Examples:
    - http://www.uni-erlangen.de/index.html
    - http://www.uni-erlangen.de:181/index.html

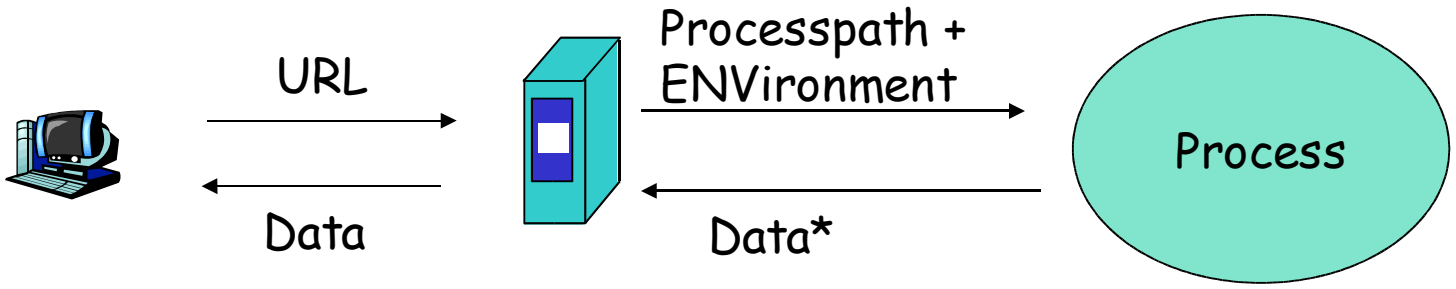
Server-Side Programming 3

- (cont.) Accessing a static page
  - DocumentRoot: „Starting point" (path) within the filesystem
  - Data of a webpage consists out of:
    - Header-Informations
      - Examples:

        „Content-type: text/html"
        „Server: Apache/1.3.19 (Unix) PHP/4.0.4pl1"
        „Title: Portal"
        „Status: 200"
        „Content_length: 6675"
    - Body (Plain Text, HTML, XML, ...)

Server-Side Programming 4

- CGI (Common Gateway Interface)



client Webserver

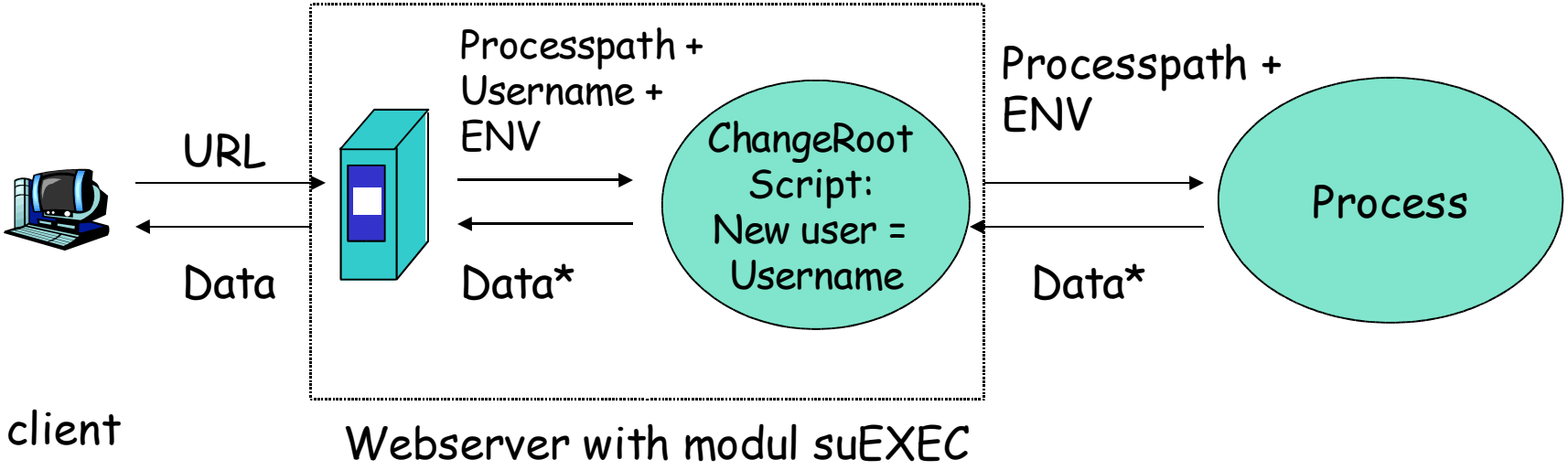
- Header-Info: Part of the header-information the webserver sends. At least „Content-type"
- Output-Data: Output as defined within Content-Type.
- Data* = Header-Info + Output-Data

Server-Side Programming 5

- CGI (cont.)
  - Process will be loaded and executed anew at every access
  - GET:
    - Data will be transmitted as addition to the URL
      - Example: http://www.uni-erlangen.de/cgi-bin/webenv.pl?data=value
    - Server will transform this into $ENV{'QUERY_STRING'}
      - Example: *QUERY_STRING = "data=value"*
  - POST:
    - Data will be transmitted to the script on <STDIN>
    - Length of transmitted data: $ENV{'CONTENT_LENGTH'}
  - Special addition: Sending data on $ENV{'PATH_INFO'}, e.g.:
    - http://www.uni-erlangen.de/cgi-bin/webenv.pl/pathinfo?data=value

Server-Side Programming 6

- CGI with User-Environment
  - Reason: Security problems at webserver running as special user (e.g. root !)
  - Several moduls to solve this: CGIWrap, suEXEC, sBox
  - Base idea: Script is executed by a user without admin-rights



Server-Side Programming 7

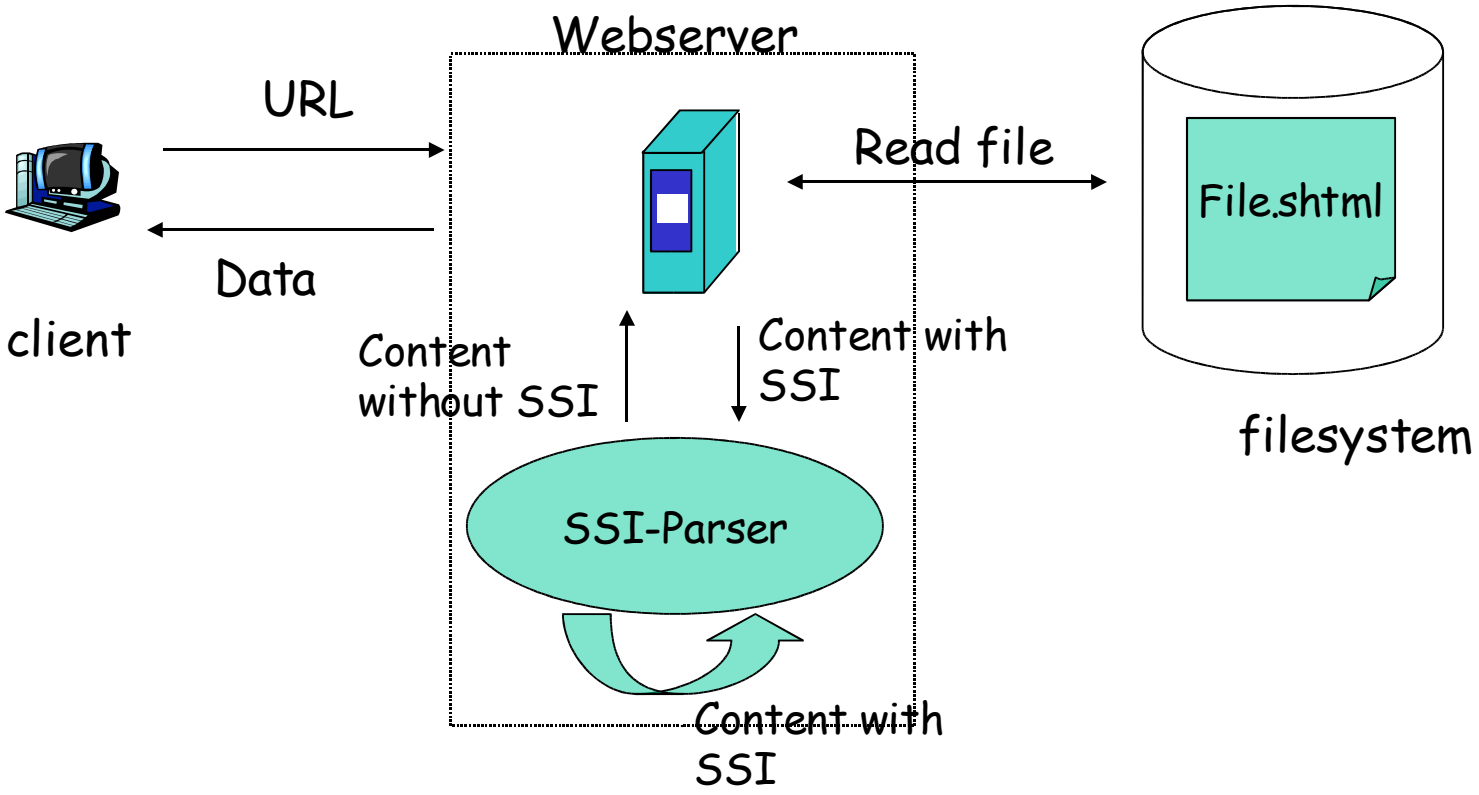
- CGI with User-Environment (cont.)
  - CGIWrap: User CGI Access (http://cgiwrap.unixtools.org)
    - Allowing the execution of cgi-scripts from local user-homes with http://www.DOMAIN.TLD/~login/cgi-bin/skript.cgi
    - /~login/cgi-bin/ forces a redirect to a wrapper-script, that executes the skript.cgi as user „login".
  - sBox: (Lincoln Stein, http://stein.cshl.org/software/sbox/)
    - CGIWrap + Configurable ceilings on script resource usage (CPU, disk, memory and process usage, sets priority and restrictions to ENV)

Server-Side Programming 8

- CGI with User-Environment (cont.)
  - suEXEC: Apache-modul (http://httpd.apache.org/docs/suexec.html)
    - Allows the execution of all CGI, SSI and PHP on a different user ID
    - Unlike Wrappers it is not bound to a special syntax in cgi-directories
    - Supports the use for virtual hosts

Server-Side Programming 9

- SSI (Server Side Includes)

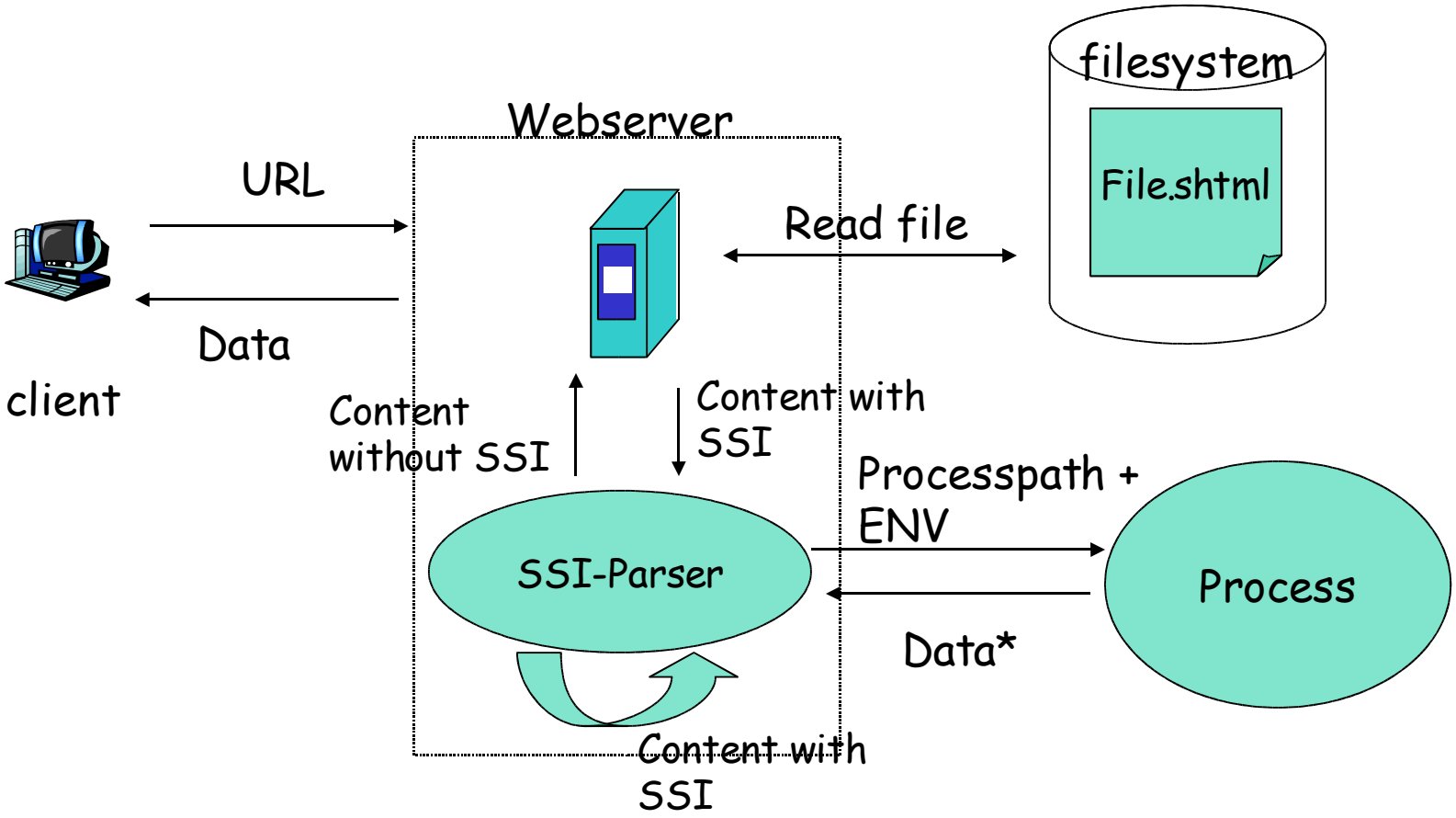


Server-Side Programming 10

- SSI (cont.)
  - SSI-Tags are parsed by the server
  - SSI-Tags are parsed as long as there are no tags anymore
  - Examples:
    - <!--#echo var=„DATE_LOCAL“--> will be replaced with the string for the local time of the server
    - <!--#include virtual=„filename.shtml“ --> will insert the content of filename.shtml. filename.shtml can use SSI-Tags too!
      (Recursive includes of files will be detected.)
    - <!--#include virtual=„/cgi-bin/skript.cgi?values“--> can be used to execute scripts
  - SSI-files mostly use the suffix „.shtml“
  - SSI works together with suEXEC, but not with CGIWrap or sBox

Server-Side Programming 11

- SSI + CGI (without suEXEC)



Server-Side Programming 12

- SSI + CGI (cont.)
  - Example SSI-file: index.shtml

```
<body>
        <!--#include virtual=„navigation.shtml“-->
        Hallo, <br>
        willkommen auf meiner Seite.
</body>
```

  - navigation.shtml

```
<hr><a href=„http://www.fau.de“>FAU</a>
<a href=„http://www.web.de“>Web.de</a>  Zeit:
<!--#config timefmt=„%d.%m.%Y, %H.%M“-->
<!--#echo var=„DATE_LOCAL“--><hr>
```

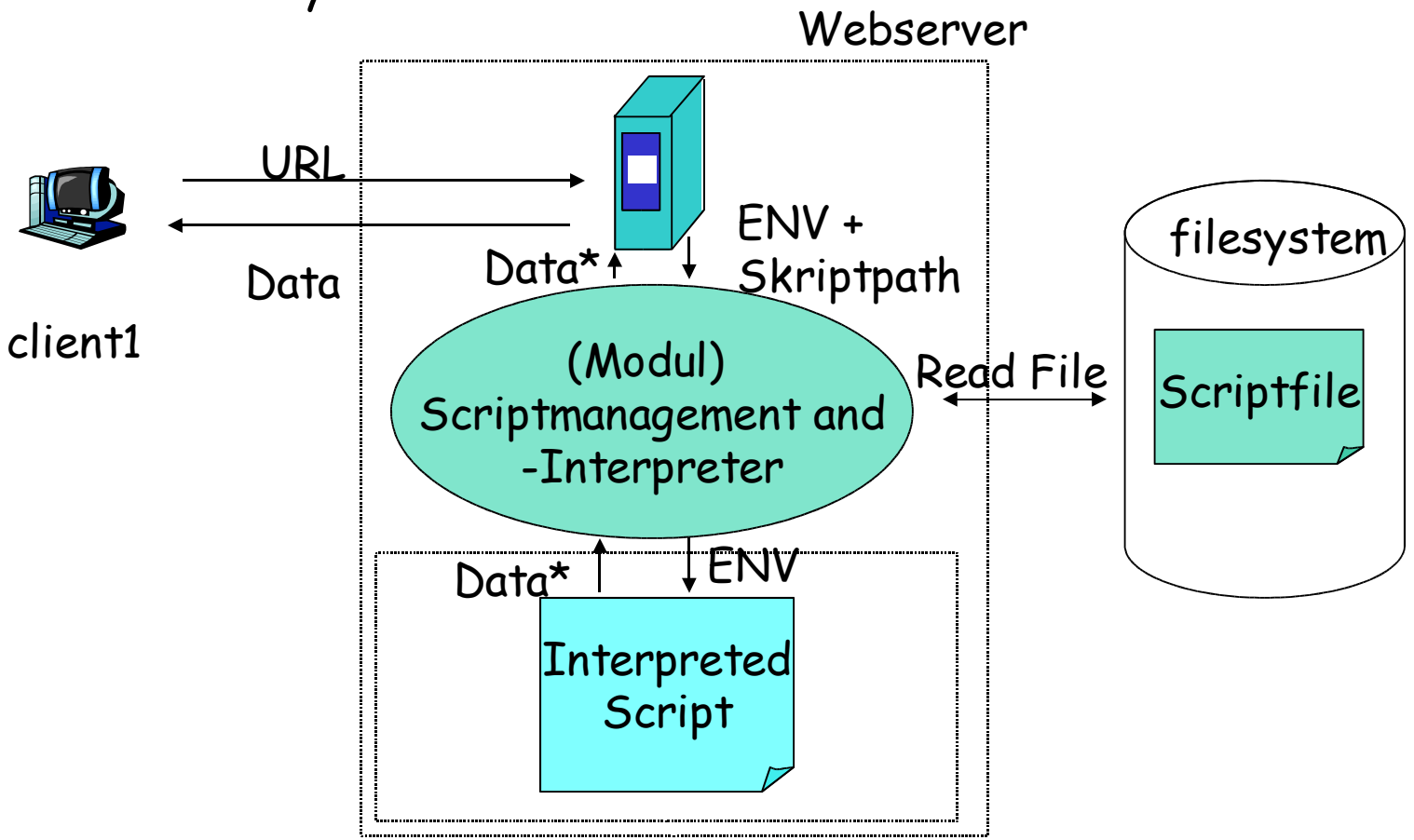
  - German samples: http://cgi.xwolf.com/faq/ssi-sample1.shtml

Server-Side Programming 13

- Embedded Scripts
  - Recall: Normal CGI-processes will be loaded and executed anew at every request.
  - Embedded scripts keep already loaded scripts in memory.
  - Script-Interpreter is part of the webserver or implemented as modul (like in Apache later Version 1.3.12)
  - Popular in use with PHP
  - Also in use for Perl-CGI-scripts and Databases

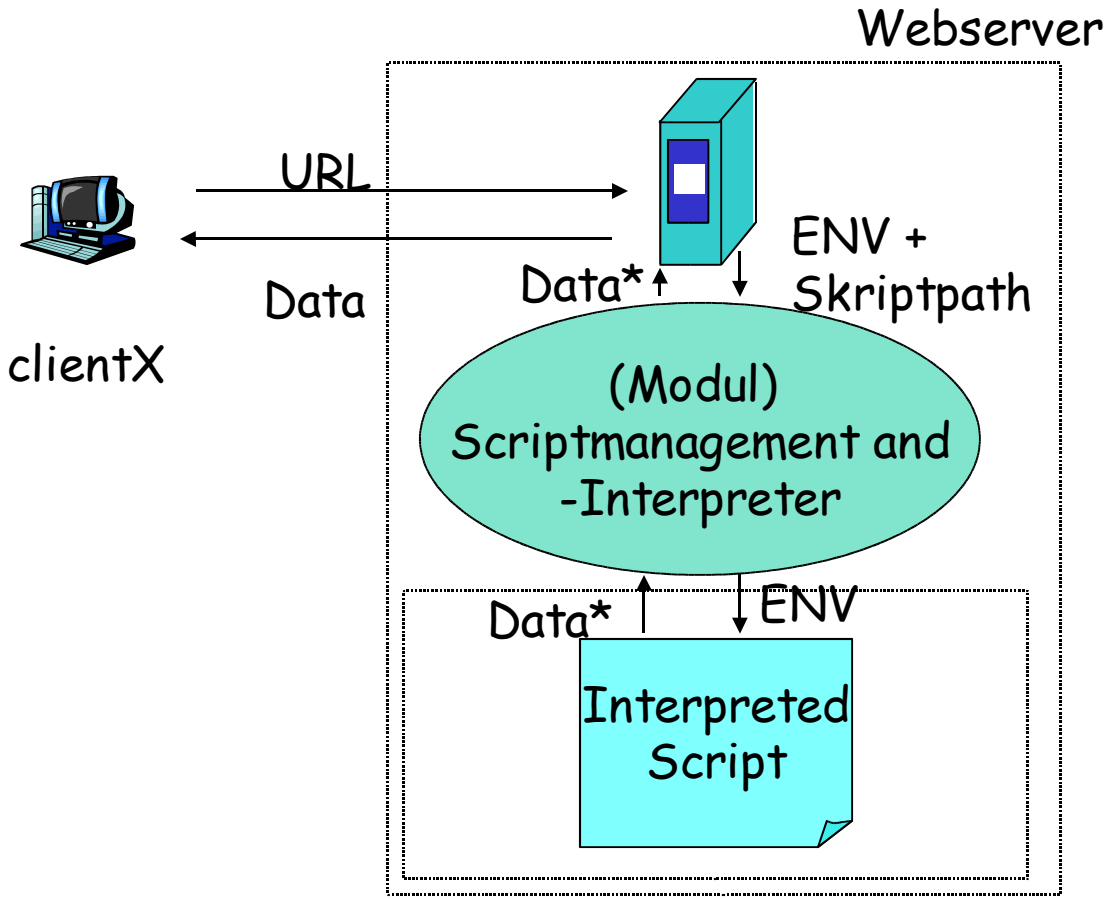
Server-Side Programming 14

- Embedded Scripts (cont.)
  - First access by client1:



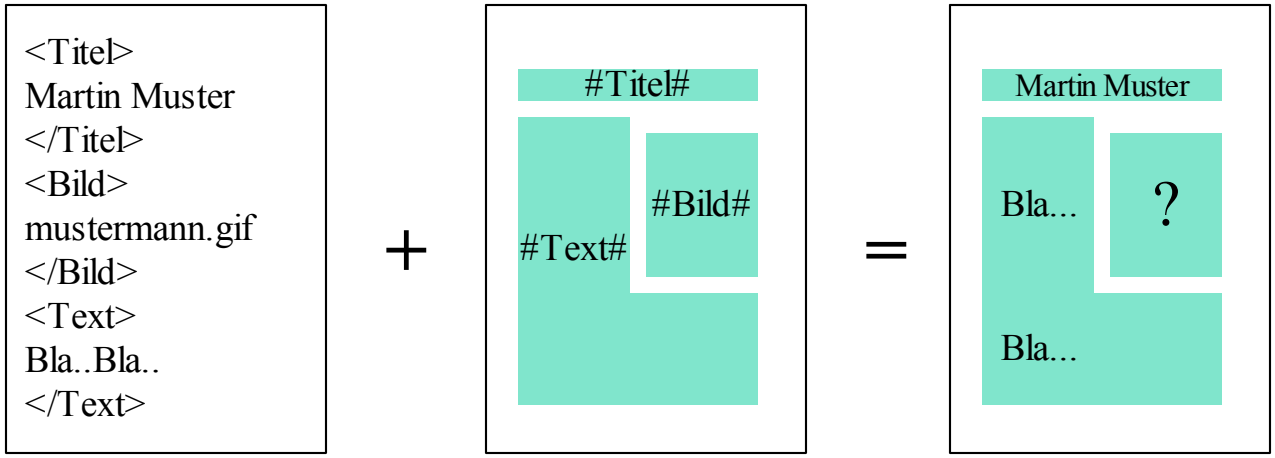
Server-Side Programming 15

- Embedded Scripts (cont.)
  - Later access for clientX



Web-Content-Management 1

- Base Principle:
  - Parting Content and Layout



Web-Content-Management 2

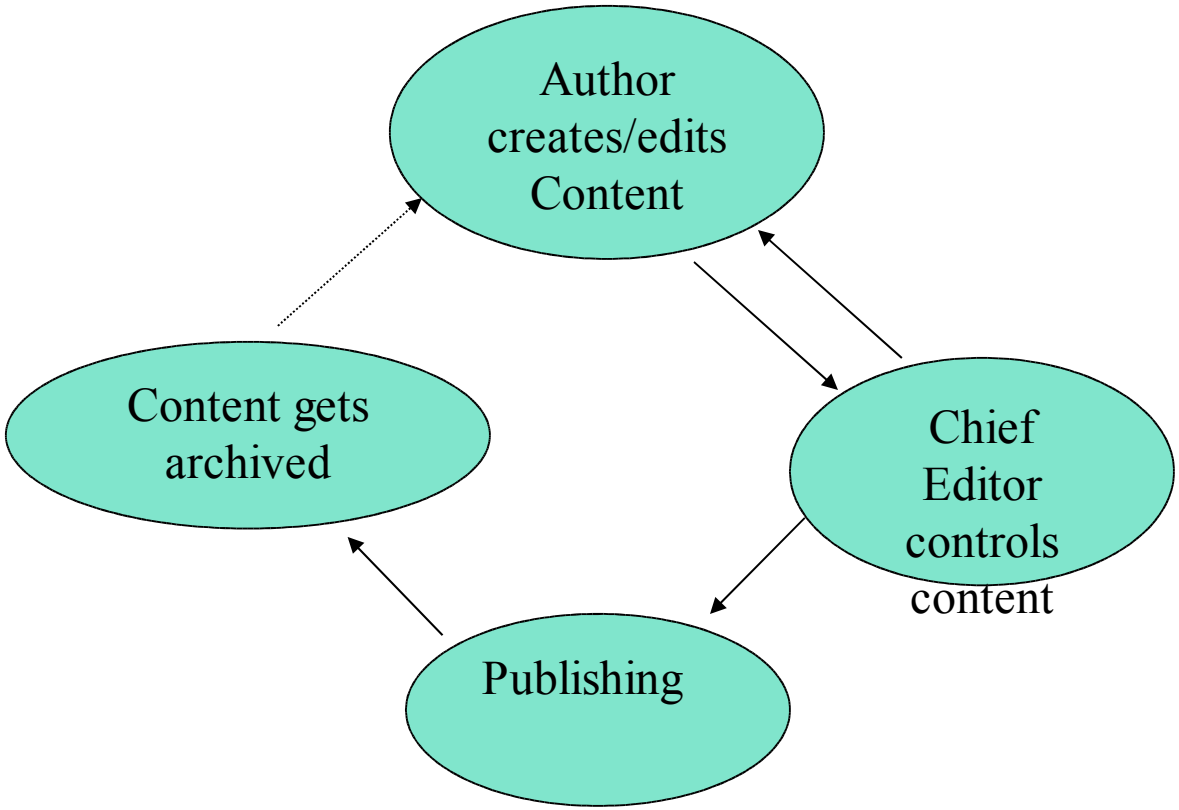
- Content-Management is need at:
  - Huge amount of information, gathered and created by many people
  - Information with references to many other information, that might refer back: complex link-trees
  - Information with a limited lifetime: Content-lifecycle
- Web-Content-Management
  - Information = Content is presented within a given layout to the public
  - Clients are requesting all information from a webserver
  - All techniques a webserver offers can be used by a web-content-management

Web-Content-Management 3

- Web-Content-Management-Systems (WCMS) are using several technics of server-side programming:
  - CGI
  - SSI
  - Embedded Scripts
- Basic aspects of WCMS are
  - Management of content and layout
  - Interaction with databases and/or special file formats
  - Concepts for data management ins respect of Web-Requests
  - User-Management
  - Workflow for content-lifecycle

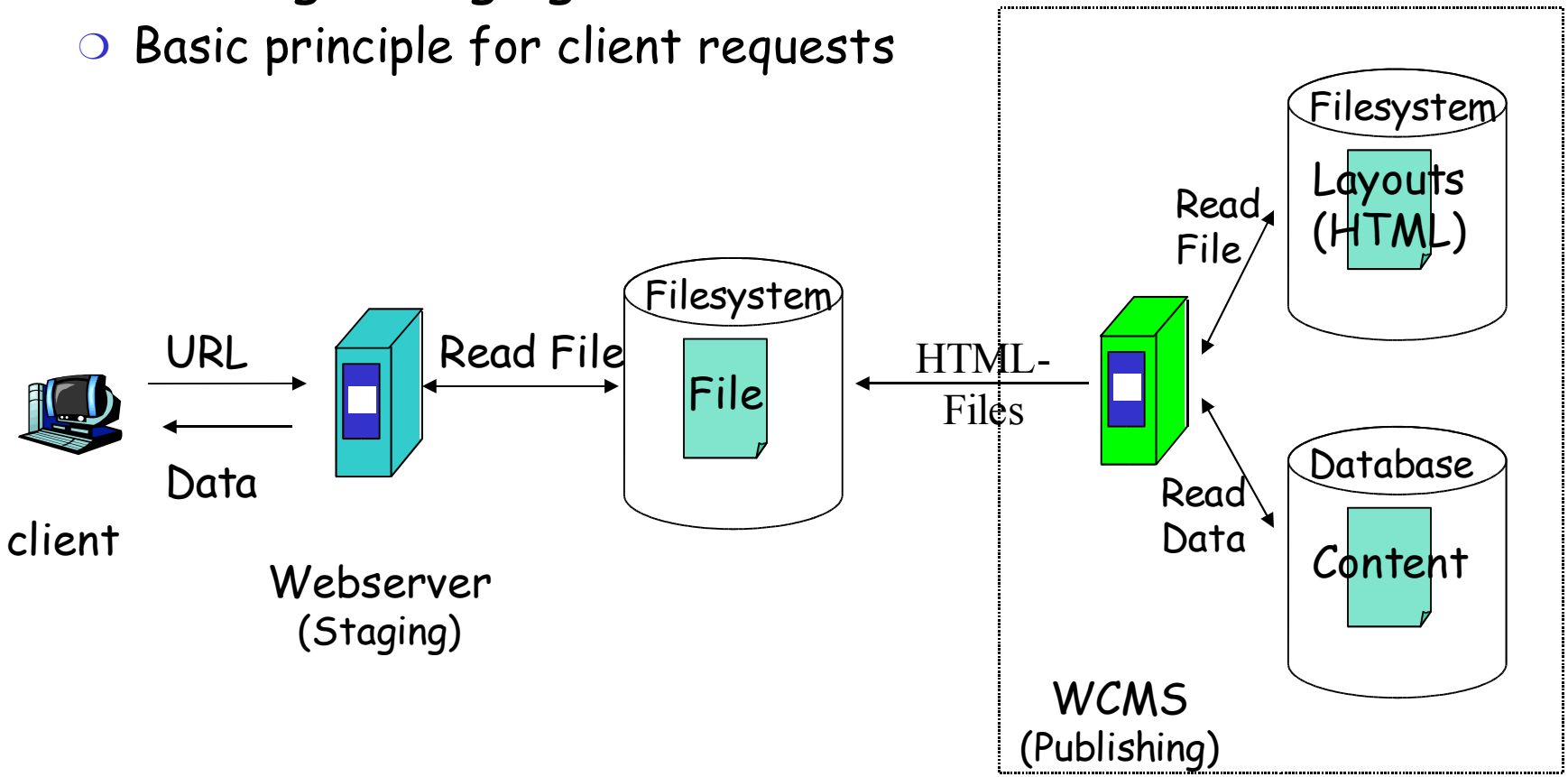
Web-Content-Management 4

- Content lifecycle



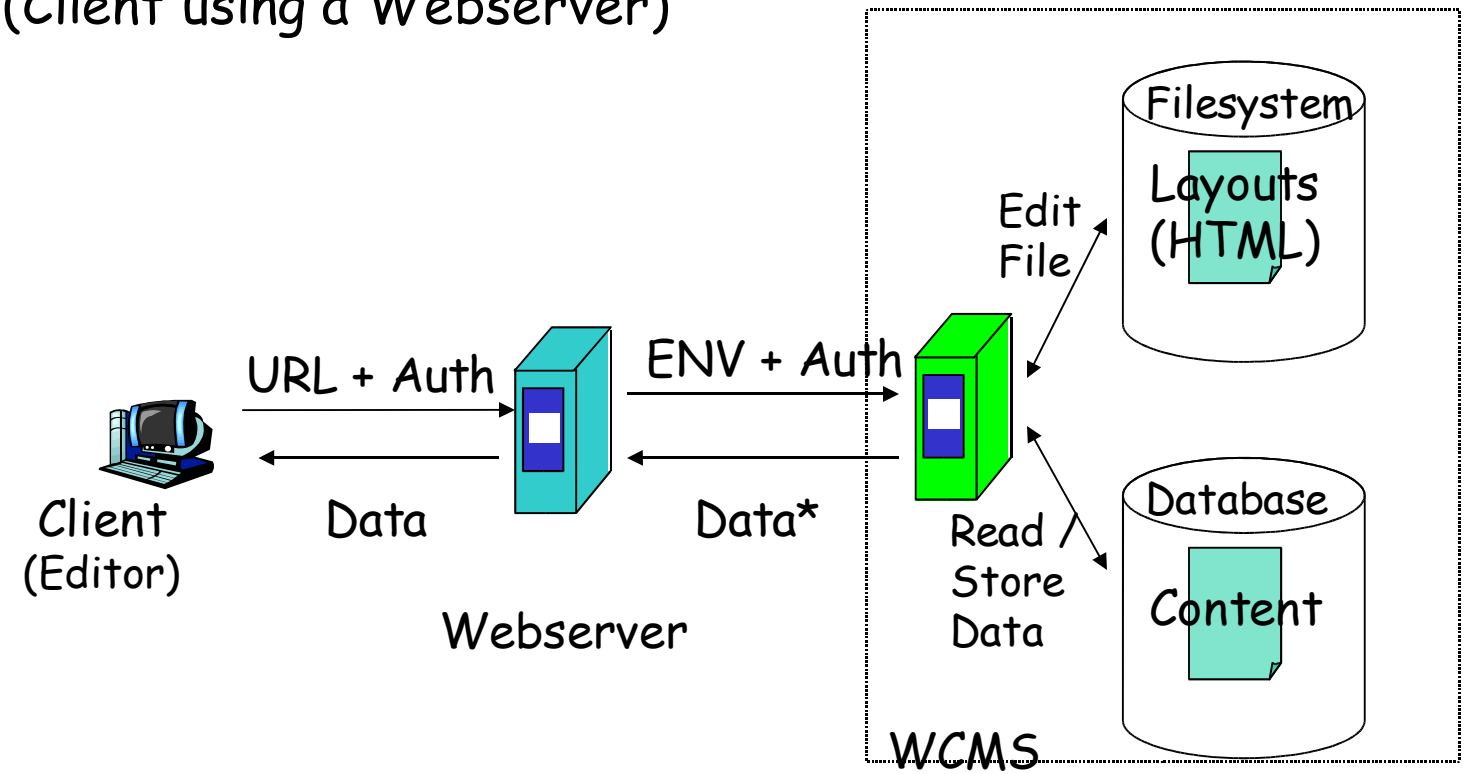
Web-Content-Management 5

- Publishing-/Staging-Server
  - Basic principle for client requests



Web-Content-Management 6

- Publishing-/Staging-Server (cont.)
  - Editors view
    (Client using a Webserver)

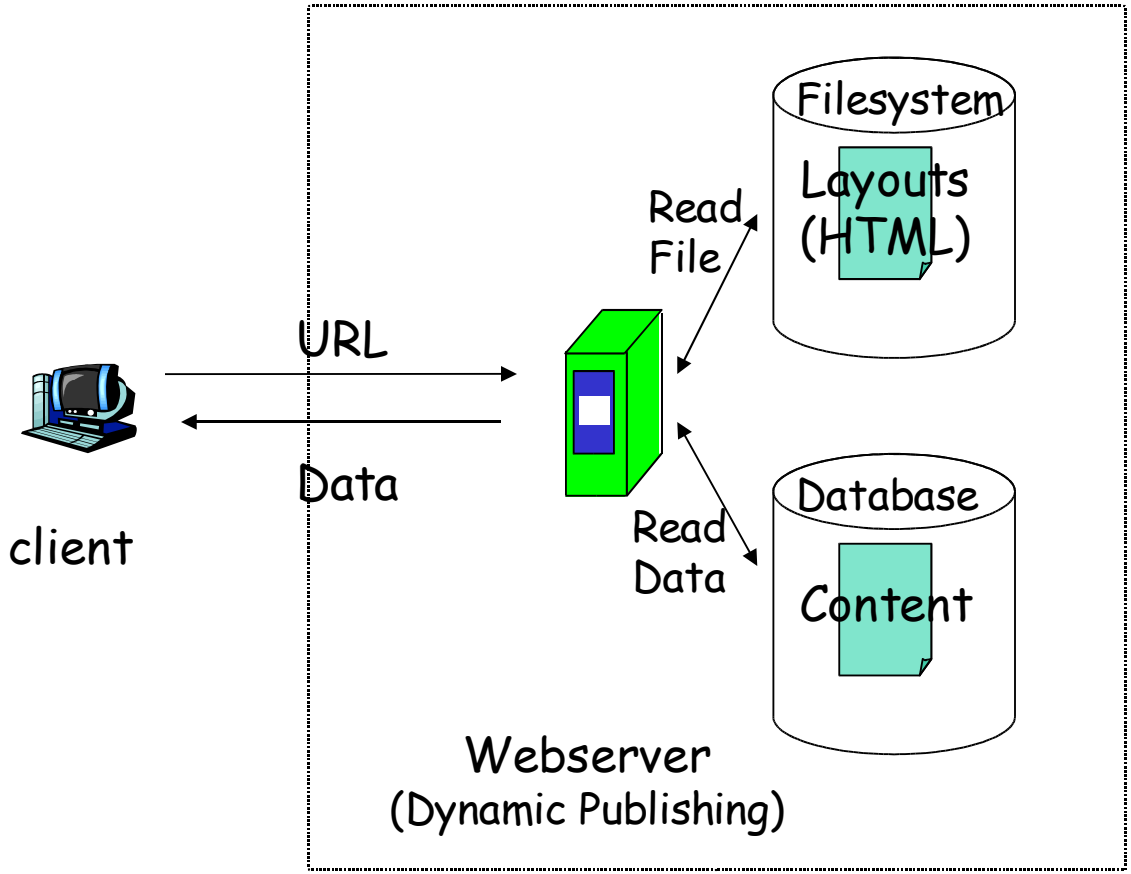


Web-Content-Management 7

- Publishing-/Staging-Server (cont.)
  - On editor command or time interval, WCMS will dump new HTML-files on Webserver's file system
  - The use of WCMS with this principle is unseen by users which are requesting web pages
  - Files are secure against modifications on the webserver: Dump of the WCMS will overwrite it
  - Good performance due to static HTML-files on webserver
  - Supports backup (database of WCMS)
  - Consistency-problems during file-dumping. Bad for pages with many changes in short time
  - Static pages are registered by internet search engines

Web-Content-Management 8

- Dynamic Publishing

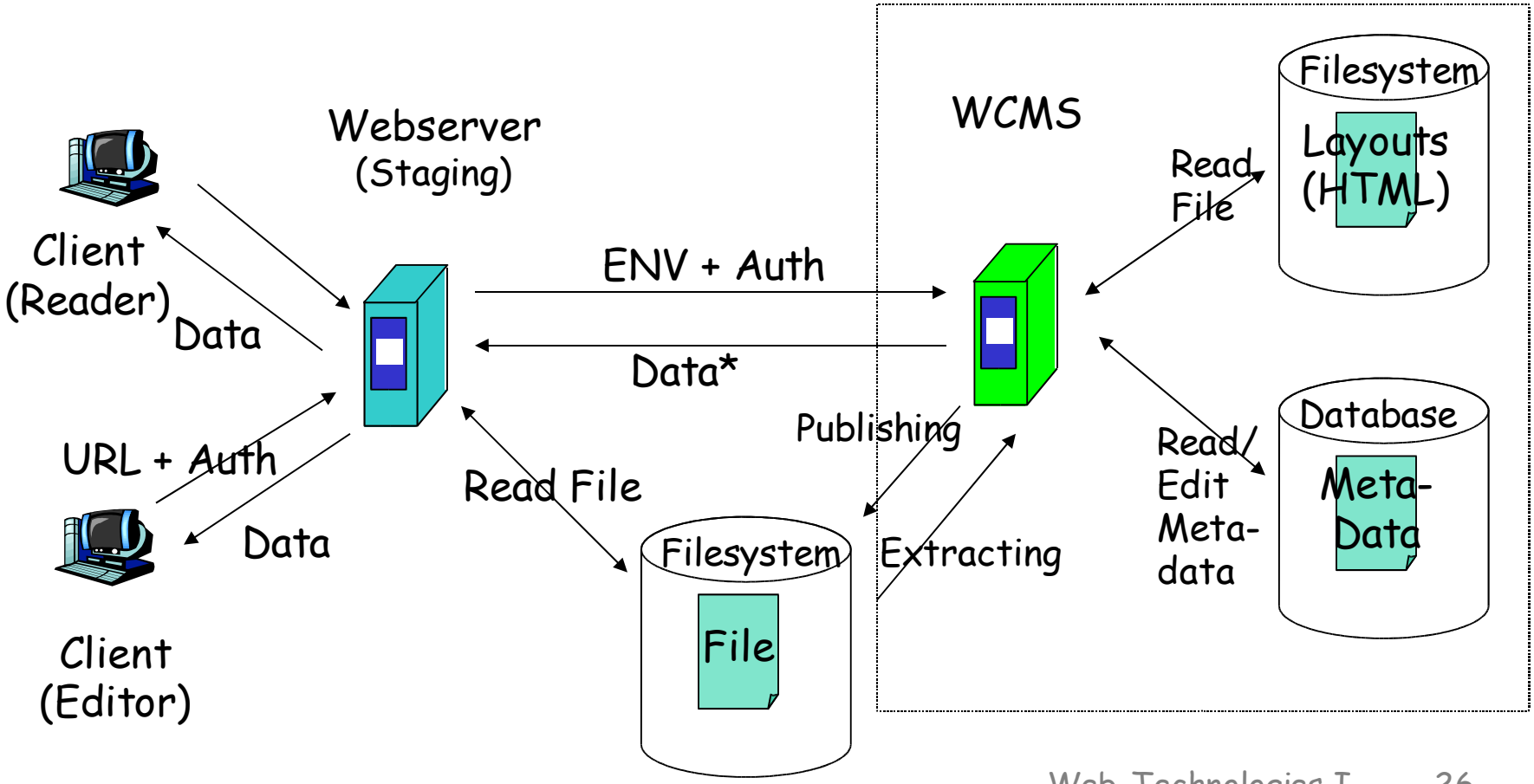


Web-Content-Management 9

- Dynamic Publishing (cont.)
  - All data is created on-the-fly: No Static pages anymore!
  - Changes in content or layout are published as soon as they are accepted
  - Local Search engines (database search) can be used to get new data-output
  - Output can get personalized for clients and/or authentificated users
  - Needs huge resources for server-hardware (CPU, disk, memory and process usage)
  - Problems with internet search engines: Mostly dynamic pages arn't registered.

Web-Content-Management 10

- Publishing- /Staging and Extract-Concept



Web-Content-Management 11

- Publishing- /Staging and Extract-Concept (cont.)
  - Good performance due to static HTML-Files
  - Supports files with many content-refreshes
  - Allows import of existing files
  - Allows the use of other WCMS and Webeditors (!)
  - Problems at change for Layout of many files

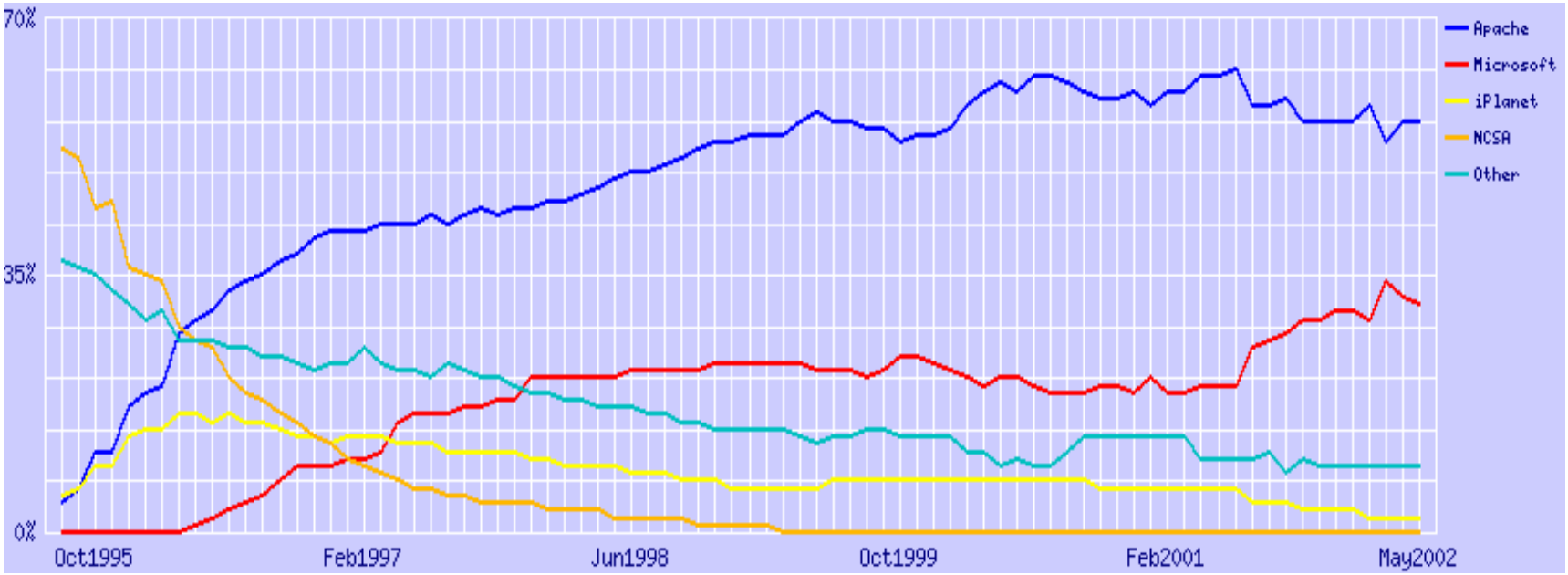
- Other concepts
  - Combinations of the methods above
  - Dynamic publishing with caching: Dumpout of few HTML-files that are requested often

Excurse Apache 1

- Apache („a patchy server")
  - Free HTTP server, supports HTTP/1.1 (RFC2616)
  - Useable on nearly all OS (but not Mac)
  - Build upon NCSA httpd (V1.3) since 1994. First release of Apache: April 1995, V 0.6.2 as beta
  - First public version in December 1, 1995
  - Developer-Team consists out of volunteers – open source project
  - Today the #1 webserver on the internet
  - Current version (Jul 2001): 1.3.20 as final and 2.0.18 as beta
  - http://www.apache.org

Excurse Apache 2

- Apache (cont.)
  - Currently used by appr. 56% of all servers in use. (MS-IIS: 31%, Netscape-Enterprise/iPlanet: 2%)
  - **37,574,105** sites tested



http://www.netcraft.com/survey

Excurse Apache 3

- Principle:
  - After start Apache will listen to requests onto port 80 (or any other defined port)
  - Configuration is stored within a textfile „httpd.conf", which is read by the httpd-process
  - On a request it will fork itself;
  - The child-process will answer the request, close the connection and then die
    - Before sending an answer, the process will parse the requesting URL and look it up for errors.
    - If the request aims a special filetype (like a server-parsed SSI-document), needed moduls are dynamically loaded or called

Excurse Apache 4

- Sample configuration file (extract)

```
Listen 131.188.3.67:80
ServerName www.rrze.uni-erlangen.de
User www
Group www
PidFile logs/httpd.pid
ServerRoot /usr/local/apache
MaxClients 220
...
LoadModule vhost_alias_module libexec/mod_vhost_alias.so
...
AddModule mod_vhost_alias.c
...
```

Excurse Apache 5

- Sample configuration file (cont.)

```
...
NameVirtualHost 131.188.3.67

<Virtualhost 131.188.3.67>
ServerName www.techfak.uni-erlangen.de
User www
Group www
DocumentRoot /proj/websource/tf/www.techfak.uni-erlangen.de
ScriptAlias /cgi-bin/ /proj/webbin/www.techfak.uni-erlangen.de/
</VirtualHost>
...
```