Web-Technologies

- Chapters
  - Client-Side Programming
  - Server-Side Programming
  - Web-Content-Management
  - Web-Services
  - Apache Webserver
  - Robots, Spiders and Search engines
    - Robots and Spiders
    - Search engines in general
    - Google

Client-Side Programming 1

- HTML
  - HTML = HyperText Markup Language
    - Developed since 1989 as platform independent markup language
    - International standardized by the W3C (http://www.w3.org)
    - Last release: Version 4.0
    - New extended version: XHTML 2.0
    - Often extended with non-standardized tags by developer of browser and web-authoring-programs

Client-Side Programming 2

- Example base structure of a HTML-document

```
<HTML>
      <HEAD>
         <TITLE>My HTML-Document</TITLE>
      </HEAD>
      <BODY>
         <P>Hallo World!</P>
      </BODY>
</HTML>
```

Client-Side Programming 3

- XML
  - Extensible Markup Language
  - With help of XML its possible to define content and the structured layout of a page in several parts => automatic analysis is possible.
    In other words:
  - *„XML is a set of rules for designing text formats, in a way that produces files that are easy to generate and read (by a computer), that are unambiguous, and that avoid common pitfalls, such as lack of extensibility, lack of support for internationalization, and platform-dependency.“*

Client-Side Programming 4

- Simple example of XML Usage:

```
<?xml version=„1.0“ ?>
<!DOCTYPE greeting [
      <!ELEMENT greeting (#PCDATA)>
      <!ELEMENT content (#PCDATA)>
]>
<greeting>Hallo XML! </greeting>
<content>
Here, we write a nice text that says nothing, but is out content...
</content>
```

See also:

http://www.w3.org/XML/
http://www.w3.org/TR/2000/REC-xml-20001006

Client-Side Programming 5

- JavaScript
  - JavaScript is a cross-platform, object-oriented scripting language.
  - Used mostly within HTML-pages.
  - JavaScript contains a core set of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements.
  - Created originally by Netscape and Sun Microsystems. (Within MSIE „extended“ by the JScript-Library).
  - Allows also usage for server-side programming

Client-Side Programming 6

- Sample JavaScript

```
<html>
<head>
      <title>Beispiel</title>
      <script language="JavaScript">    <!--
       function Quadrat(Zahl) {
         Ergebnis = Zahl * Zahl;
         alert("Das Quadrat von " + Zahl + " = " + Ergebnis);
        }
      //-->   </script>
</head>
<body><form>
<input type=button value="Quadrat von 6 errechnen" onClick="Quadrat(6)">
</form></body></html>
```

Client-Side Programming 7

- Sample JavaScript (cont.)

Client-Side Programming 8

- JavaScript (cont.)
  - JavaScript is mostly used as enhancement for webdesign; Due to its possibility to access and chance objects (like HTML-Tags), it allows effects fo improve the usability of websites.
    - Often used: onMouseOver, onClick
    - Professional effects in combination with CSS (but CSS also replaces some JS-functions)

Client-Side Programming 9

- Cascading Style Sheets (CSS)
  - HTML specification lists guidlines on how browsers should display HTML-Tags. Example:

```
<style type=„text/css“>
      h1,h2,h3,h4 {
            color: navy;
            font-family: Garamond, Helvetica, serif;
      }
      h1.dark {
            color: black;
      }
</style>
```

Client-Side Programming 10

- Cascading Style Sheets (cont.)
  - CSS is, like HTML, standardized by the W3C http://www.w3.org/Style/CSS
  - In combination with new HTML-Versions, it will replace old HTML-tags, like <font>, <hr>, , ...
  - CSS requires browsers that supports this format (IE / NS >= V4.0). Older browsers will ignore all settings made by CSS.
  - CSS definitions can be placed within a file; Therefore it‘s possible to chance the layout of all WebPages by changing one single CSS-file.

Client-Side Programming 11

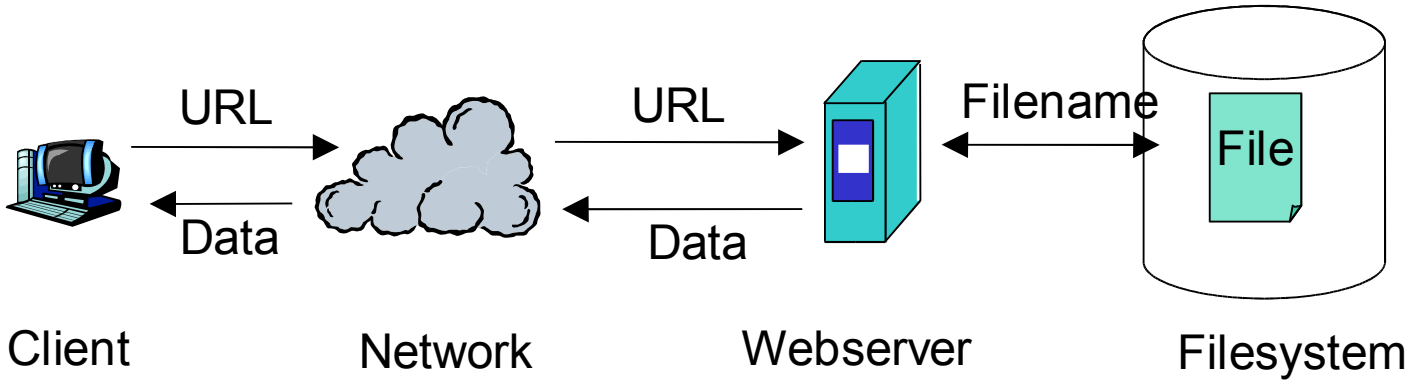
- Other client-side techniques
  - Flash
    - Browser-Plugin by Macromedia (http://www.macromedia.com)
    - Allows interactive vector-graphics and animations
    - Mostly used for special effects, small movieclips and 3D-graphics
  - RSS-Feeds
    - RSS = ***R**eally **S**imple **S**yndication* (Web content syndication format.)
    - XML-File for special news. Often also used for Blogs (Web-Logs)
    - Technical infos: http://backend.userland.com/rss
  - Experimental or old techniques:
    - cURL, VRML (Virtual Reality Modeling Language)

Server-Side Programming 1

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

Server-Side Programming 2

- Recall: Accessing a static page



Typical access: *URL* = *Protocol* + *Domain name* or *IP* (+ *Port*) + *Filename* *within the Document Root*

Examples:

- *http://www.uni-erlangen.de/index.html*
- *http://www.uni-erlangen.de:181/index.html*
- *https://131.188.3.67/internal/documents/*

Server-Side Programming 3

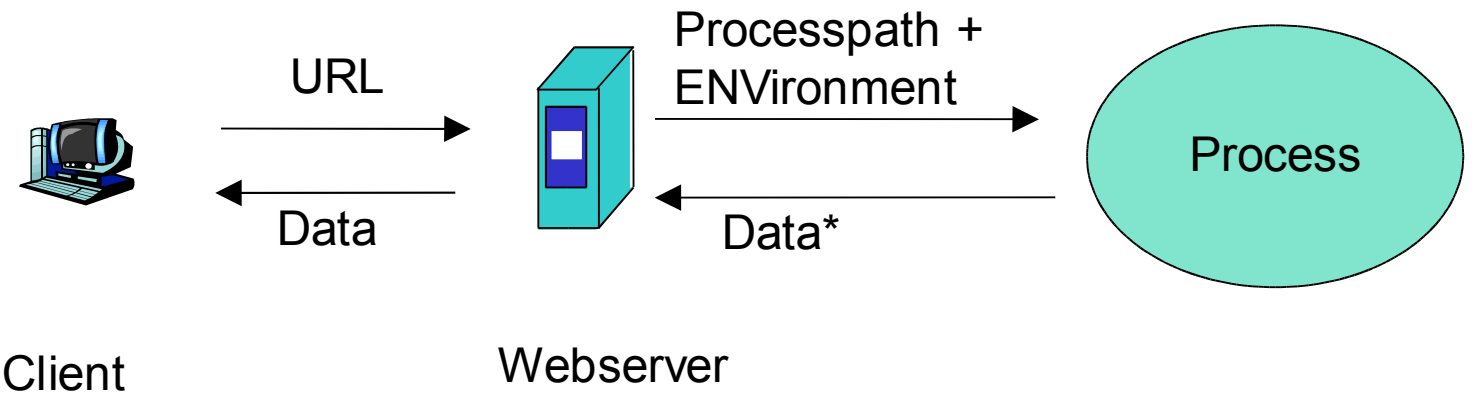
- Accessing a static page (cont.)
  - Document Root: „Starting point“ (path) within the filesystem
  - Data of a webpage consists of:
    - Header-Informations
      - Example:

```
Content-type: text/html
Server: Apache/1.3.27
Title: Portal
Status: 200
Content_length: 6675
```

    - Body (Plain Text, HTML, XML, ...)

Server-Side Programming 4

- CGI (Common Gateway Interface)



  - 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-Request:
    - 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 the standard environment set of the server: *$ENV{'QUERY_STRING'}*
      - Example: *QUERY_STRING = "data=value"*
    - Optional use: Sending data on $ENV{'PATH_INFO'} by using pathes:
      - http://www.uni-erlangen.de/cgi-bin/webenv.pl/pathinfo?data=value

Server-Side Programming 6

- CGI (cont.)
  - POST-Request:
    - Data will be transmitted to the script on <STDIN>
    - Information wont get saved within the URL
    - Length of transmitted data: *$ENV {'CONTENT_LENGTH'}*

Server-Side Programming 7

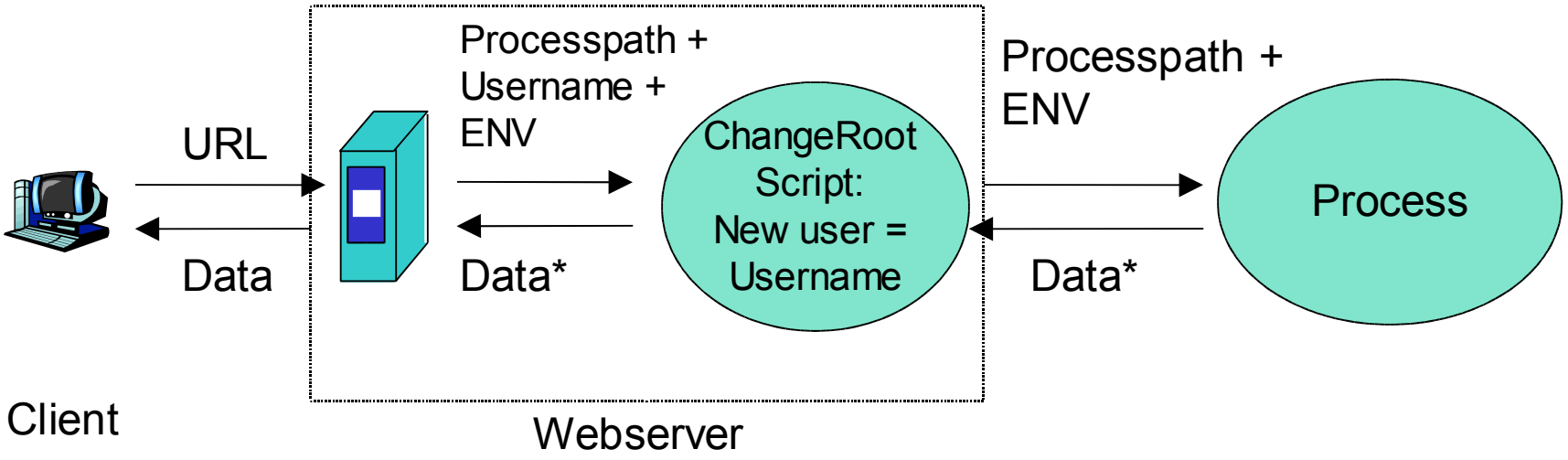
- CGI with User-Environment
  - Reason: Security problems at webservers running as special user (e.g. root !)
  - Several modules to solve this: *CGIWrap, suEXEC, sBox*
  - Base idea: Script is executed by a user without admin-rights

Server-Side Programming 8

- 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 9

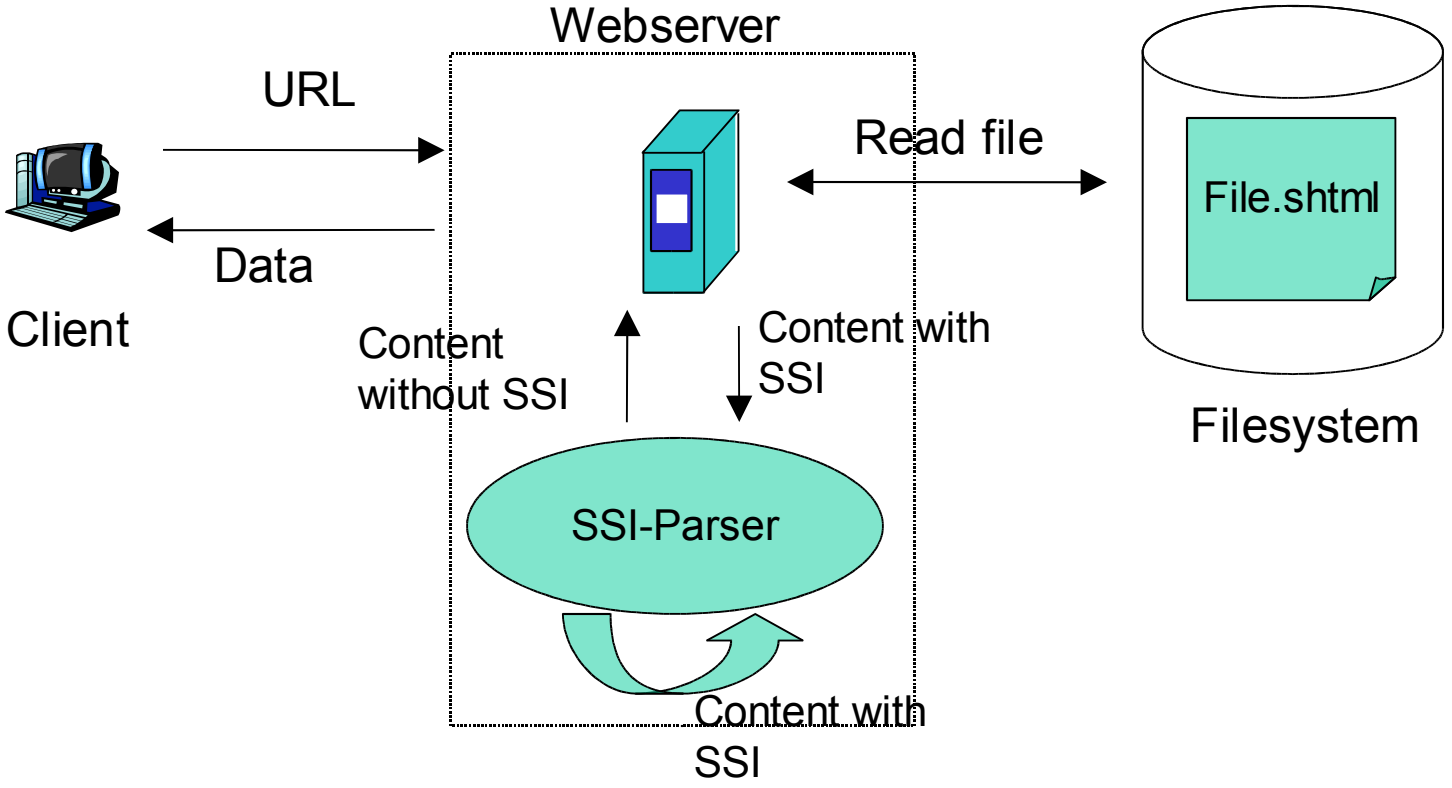
- CGI with User-Environment (cont.)
  - suEXEC: Apache-module (http://httpd.apache.org/docs/suexec.html)



  - Allows the execution of all CGI-Scripts, SSI and PHP-CGI on a defined user ID
  - No special syntax for cgi-directories
  - Supports the use for virtual hosts

Server-Side Programming 10

- SSI (Server Side Includes)

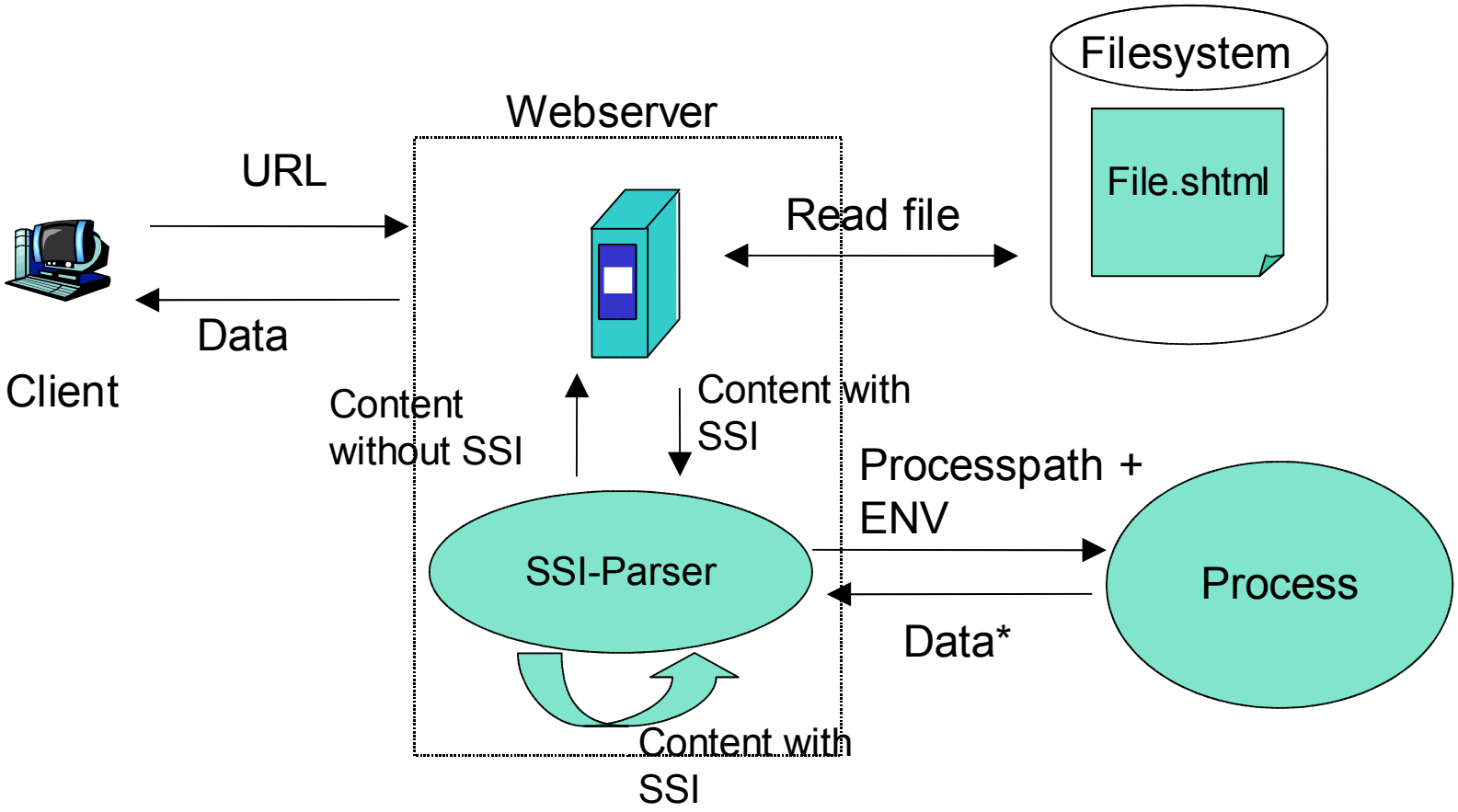


Server-Side Programming 11

- SSI (cont.)
  - SSI-Tags are parsed by the server
  - SSI-Tags are parsed as long as there are no SSI-Tags anymore within the document
  - 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 often use the suffix „.shtml“ as default
  - SSI works together with suEXEC, but not with CGIWrap or sBox

Server-Side Programming 12

- SSI + CGI (without suEXEC)



Server-Side Programming 13

- 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.de/ssi

Server-Side Programming 14

- SSI + CGI (cont.)
  - Content send to the UserAgent by the web server:

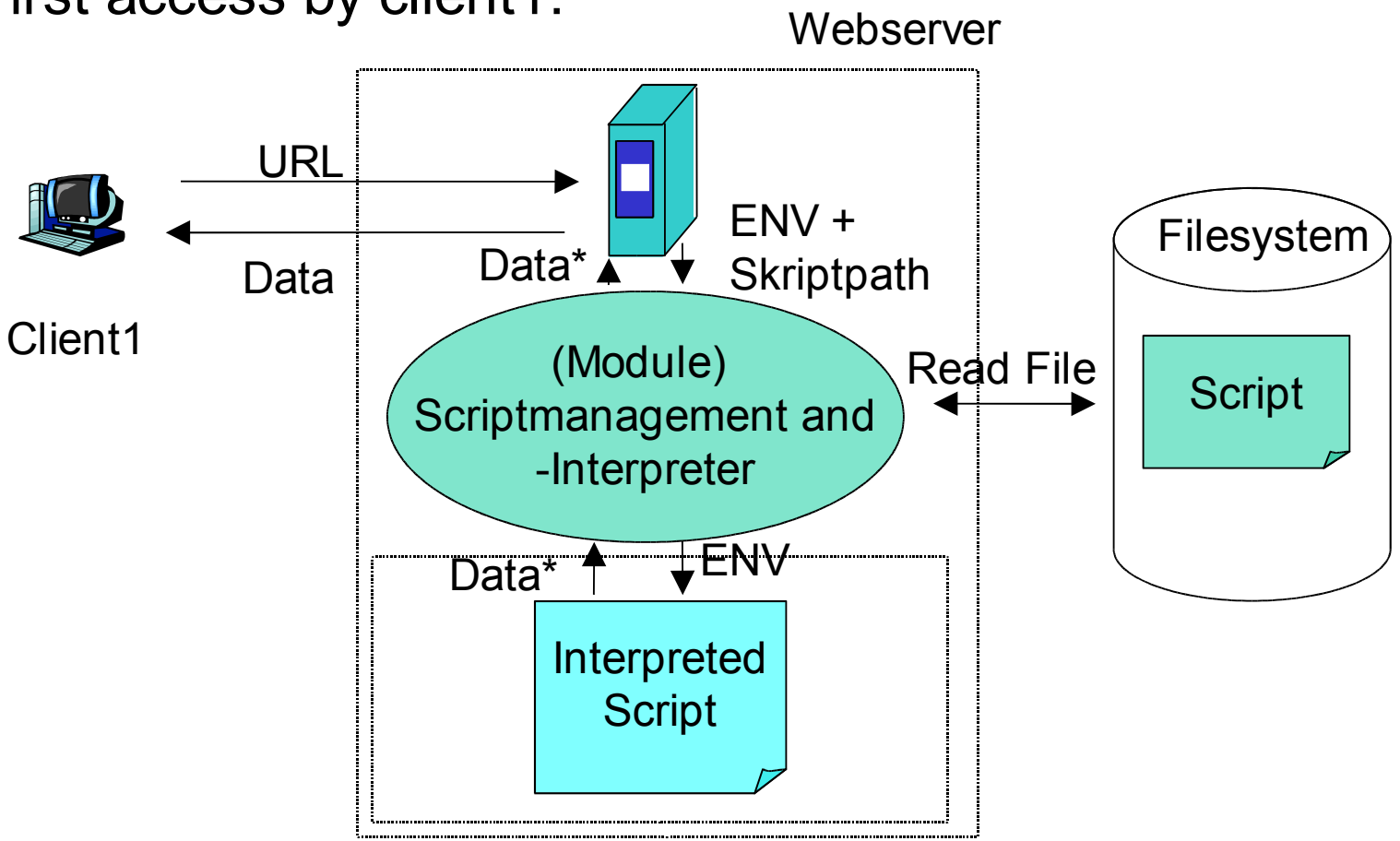
```
<body>
        <hr><a href=„http://www.fau.de“>FAU</a>
<a href=„http://www.web.de“>Web.de</a>  Zeit:
26.06.2003, 13.17<hr>

        Hallo, <br>
        willkommen auf meiner Seite.
</body>
```

Server-Side Programming 15

- 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 (compiled) part of the webserver or implemented as module (like in Apache later Version 1.3.12)
  - Popular in use with PHP
  - Also in use for Perl-CGI-scripts and Databases

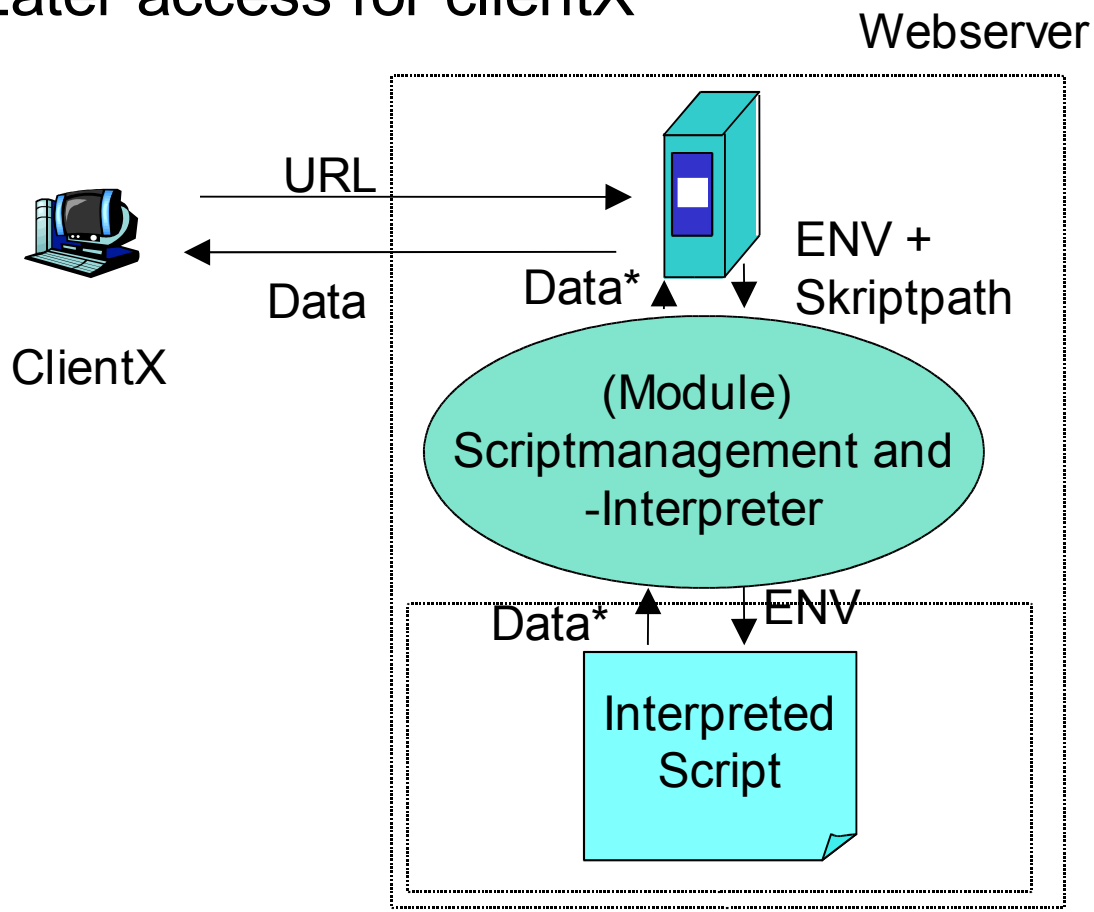
Server-Side Programming 16

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



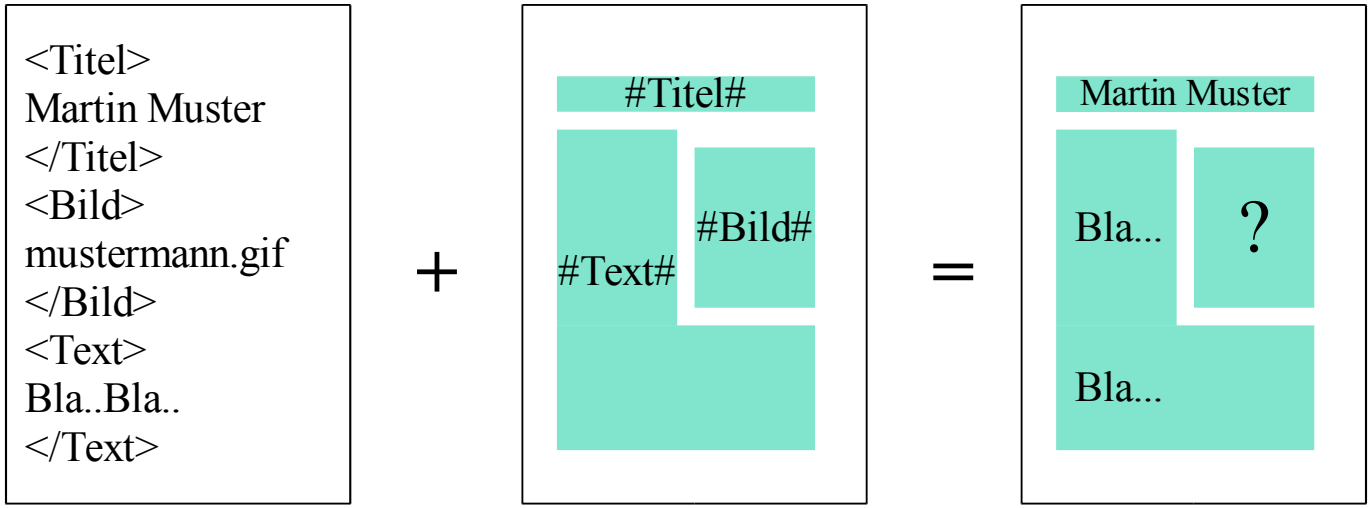
Server-Side Programming 17

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



Web-Content-Management 1

- Basic Principle:
  - Partitioning Content and Layout



Web-Content-Management 2

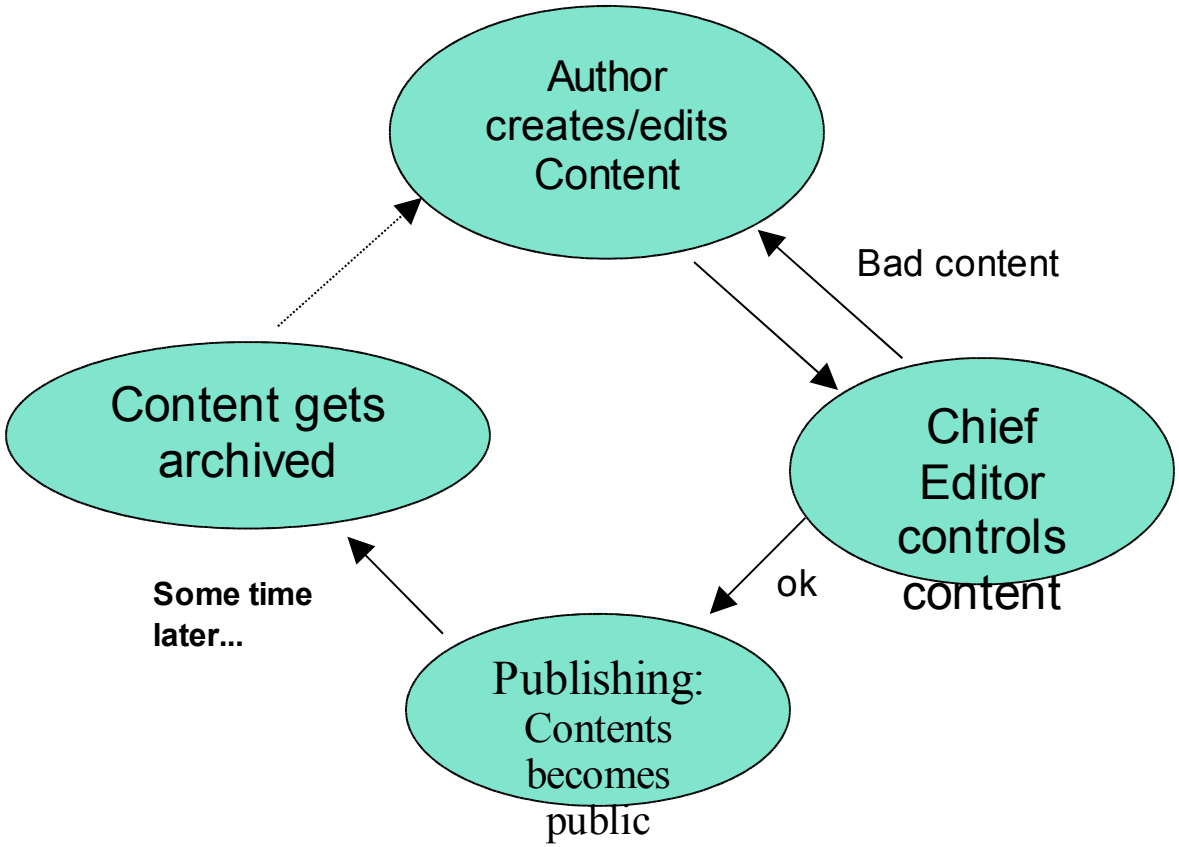
- Content-Management is needed for:
  - Huge amount of information, gathered and created by many people
  - Information with references to many other information, that might refer back: complex link-trees
  - Information with a limited lifetime: Content-lifecycle
- Web-Content-Management
  - Information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

Web-Content-Management 3

- Web-Content-Management-Systems (WCMS) are using several techniques 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 in respect of Web-Requests
  - User-Management
  - Workflow for content-lifecycle

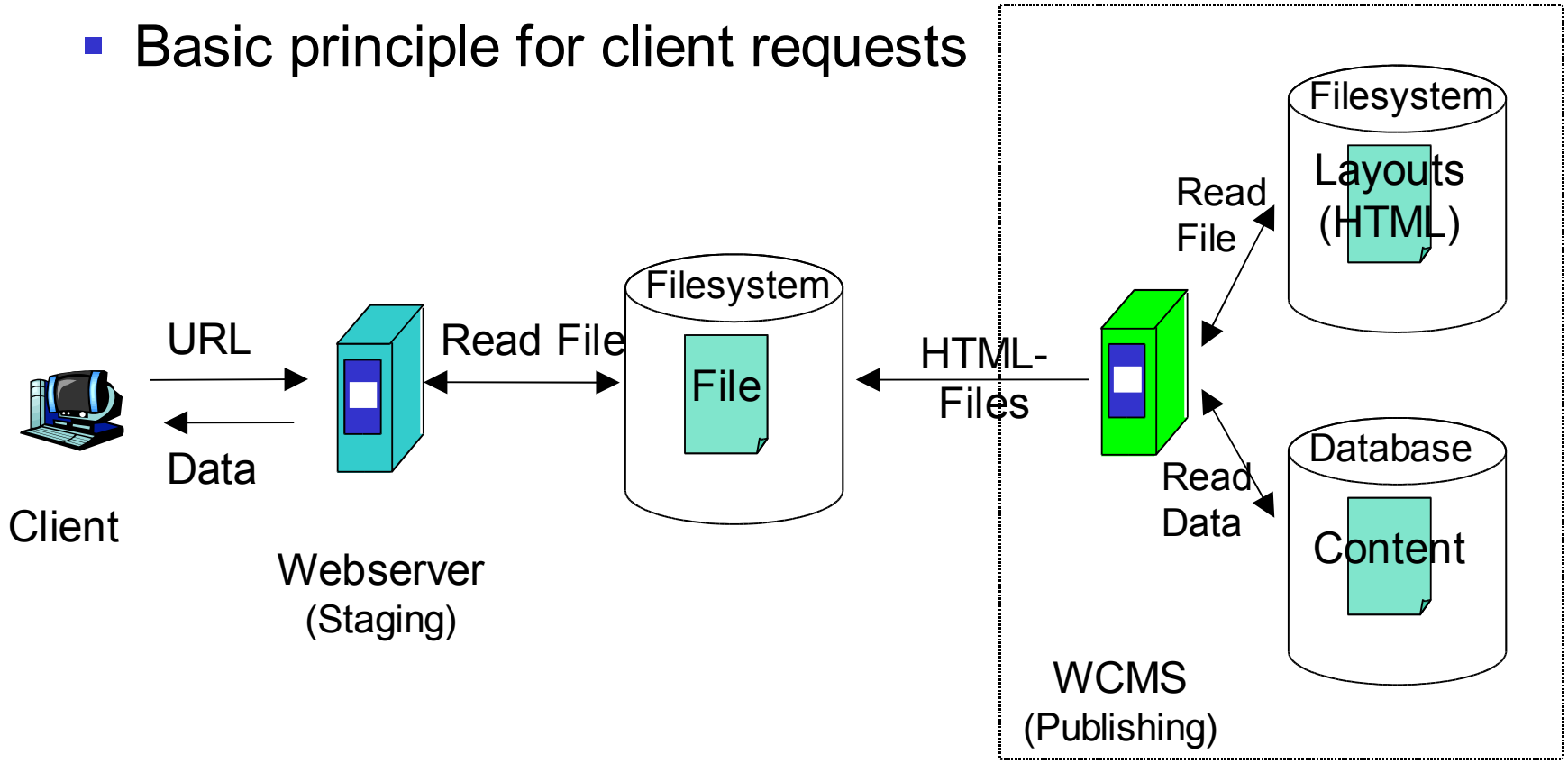
Web-Content-Management 4

- Content lifecycle



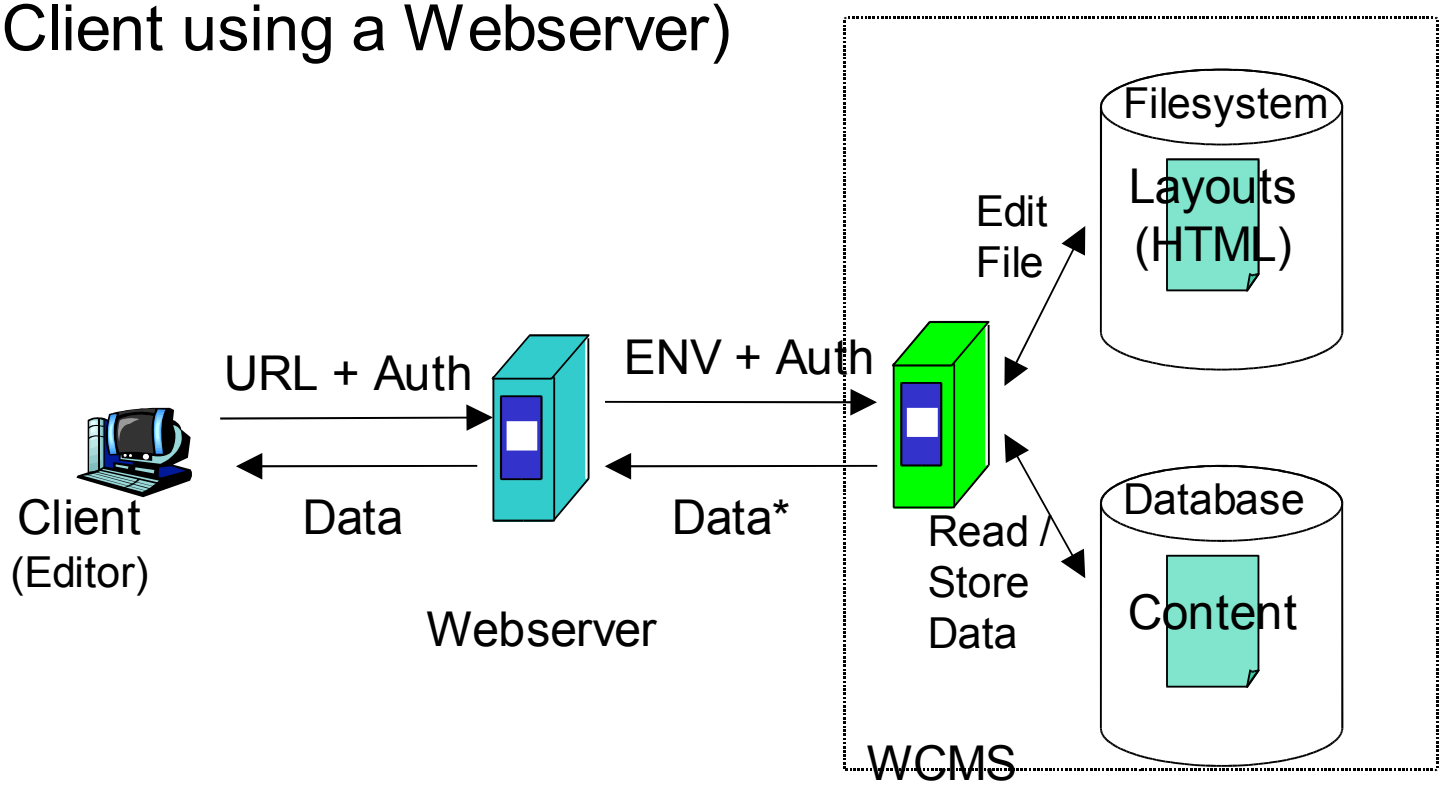
Web-Content-Management 5

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



Web-Content-Management 6

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

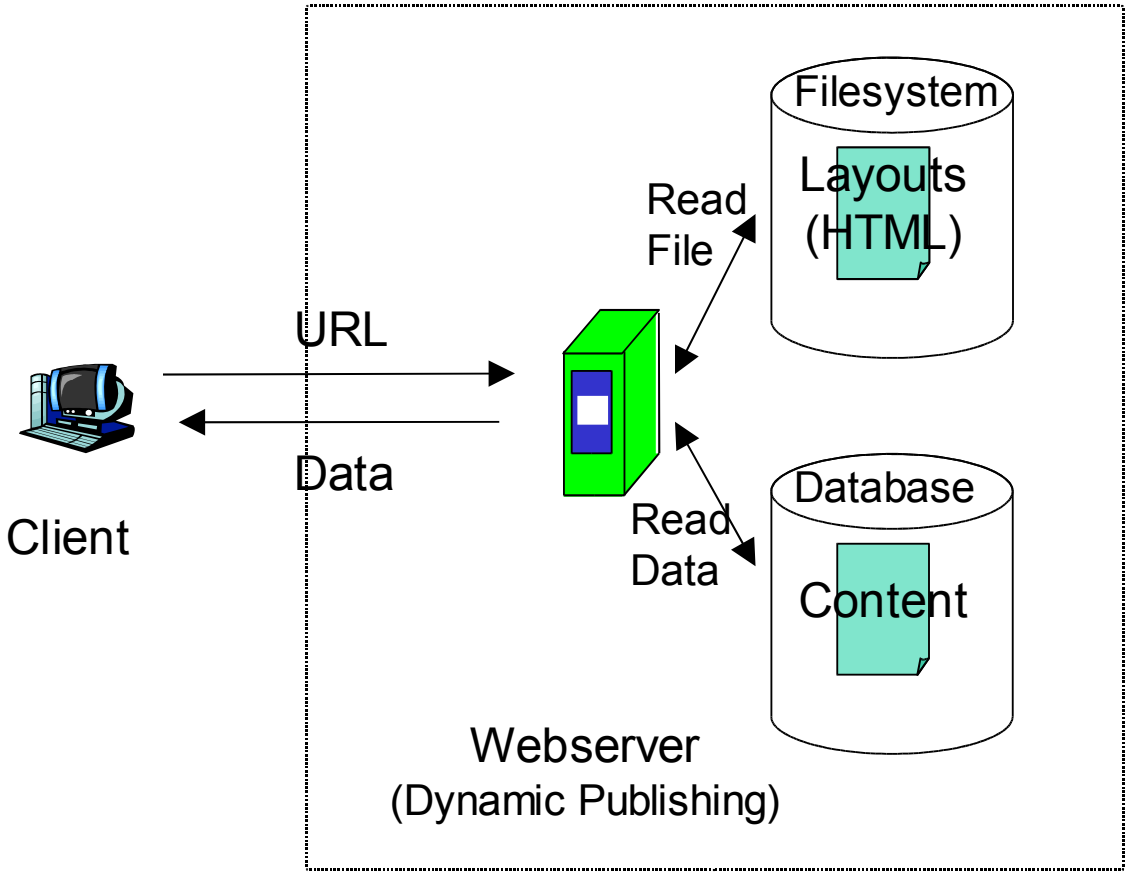


Web-Content-Management 7

- Publishing-/Staging-Server (cont.)
  - On editor command or time interval, WCMS will dump new HTML-files on Webserver‘s filesystem
  - The use of WCMS with this principle is (mostly) transparent to 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

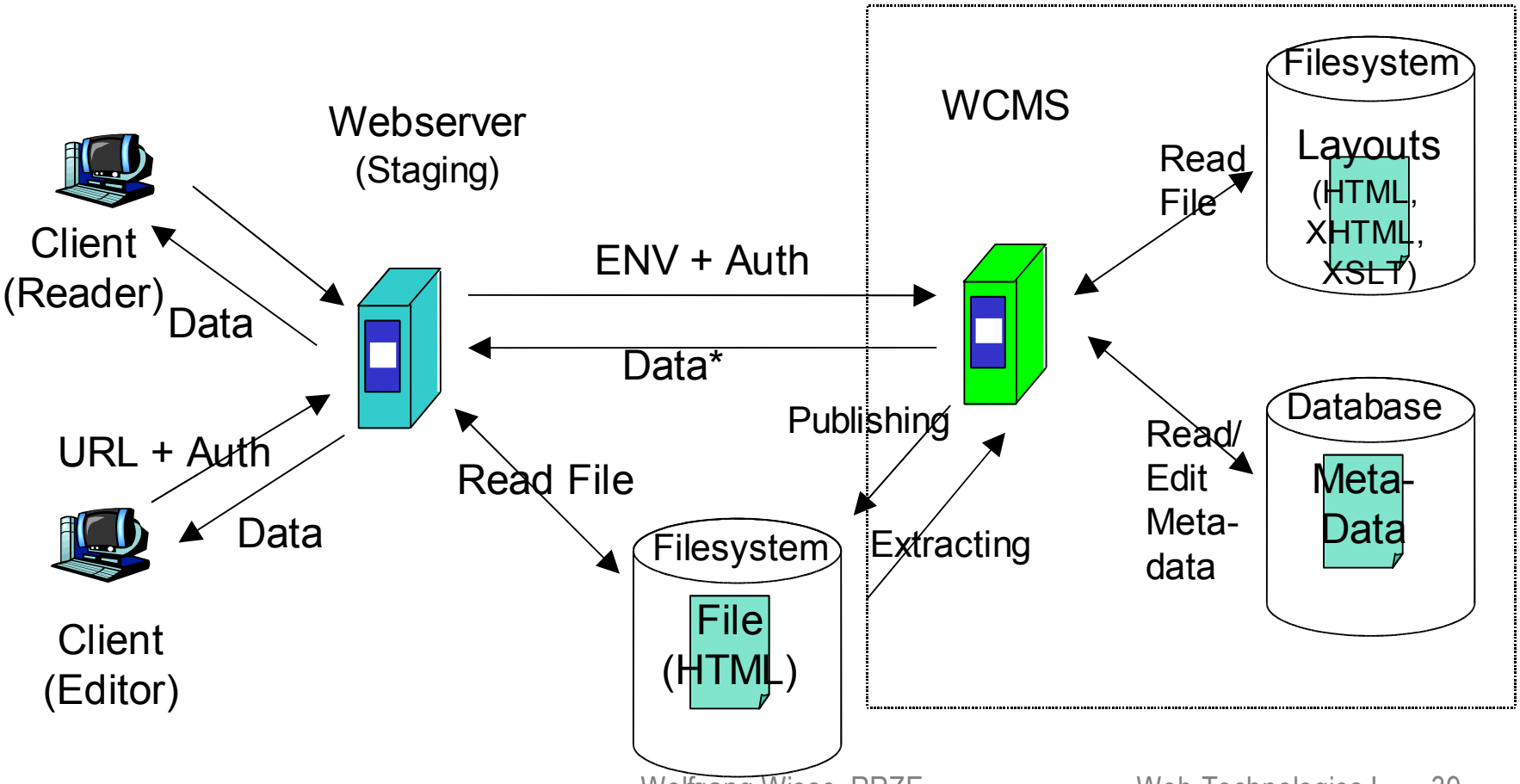


Web-Content-Management 9

- 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: Dynamic pages will not get listed in search engines (!)

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 parallel use of other WCMS and Webeditors onto the same files
  - Problems with 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