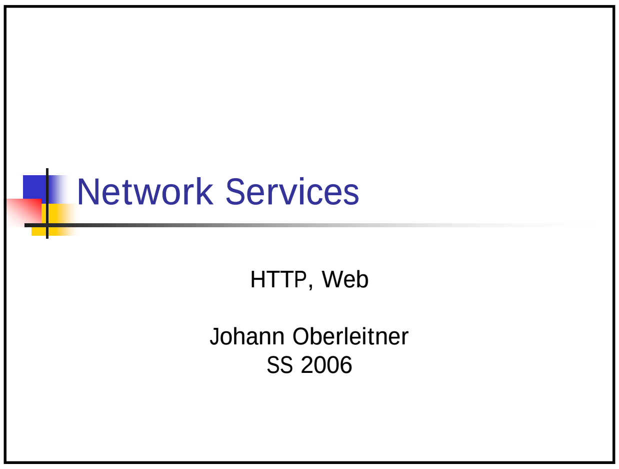
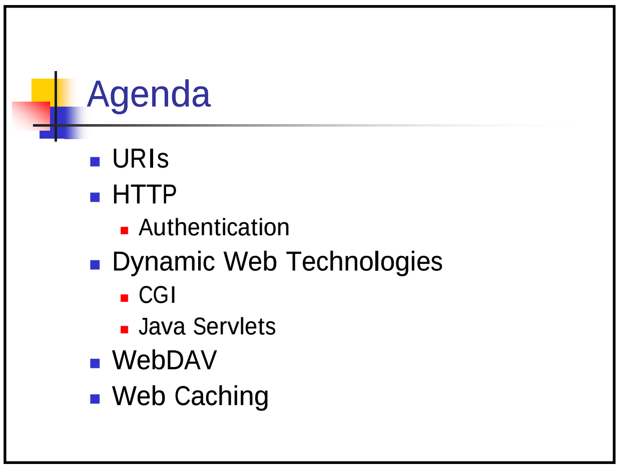


Network Services

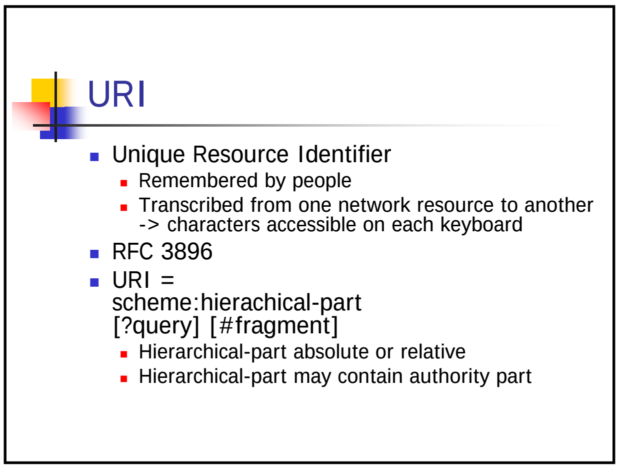
HTTP, Web

Johann Oberleitner
SS 2006



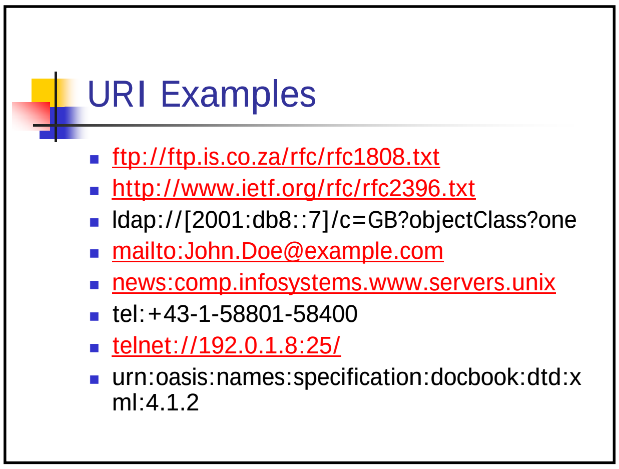
Agenda

- URIs
- HTTP
  - Authentication
- Dynamic Web Technologies
  - CGI
  - Java Servlets
- WebDAV
- Web Caching



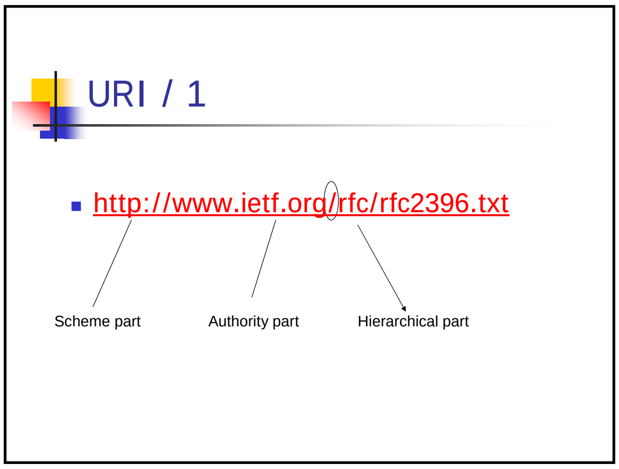
URI

- Unique Resource Identifier
  - Remembered by people
  - Transcribed from one network resource to another -> characters accessible on each keyboard
- RFC 3896
- URI = scheme:hierachical-part [?query] [#fragment]
  - Hierarchical-part absolute or relative
  - Hierarchical-part may contain authority part



URI Examples

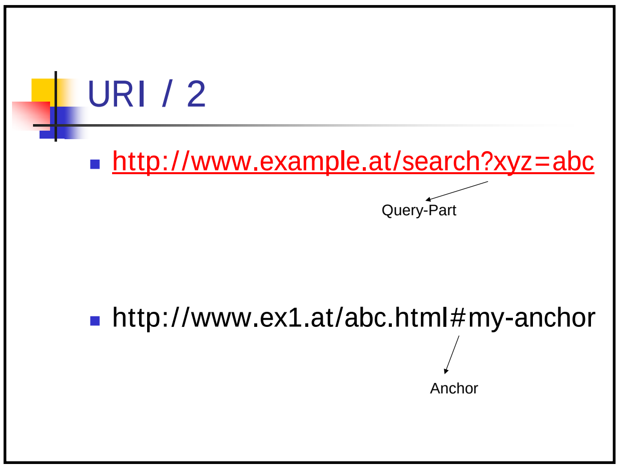
- ftp://ftp.is.co.za/rfc/rfc1808.txt
- http://www.ietf.org/rfc/rfc2396.txt
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:John.Doe@example.com
- news:comp.infosystems.www.servers.unix
- tel:+43-1-58801-58400
- telnet://192.0.1.8:25/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2



URI / 1

- http://www.ietf.org/rfc/rfc2396.txt

Scheme part | Authority part | Hierarchical part



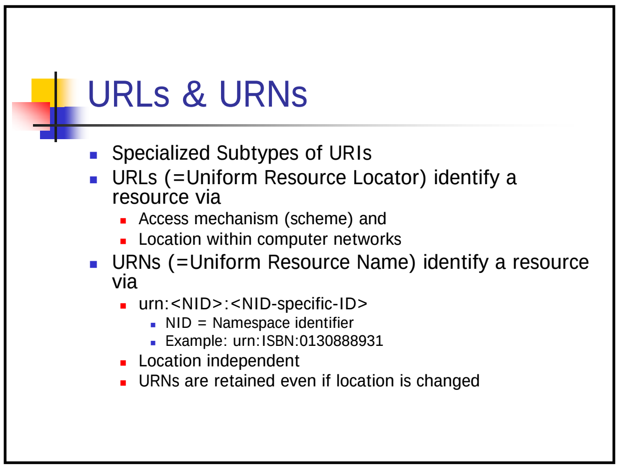
URI / 2

- http://www.example.at/search?xyz=abc

Query-Part

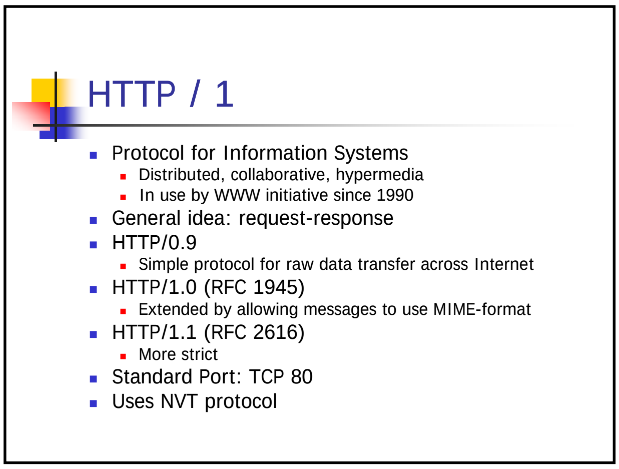
- http://www.ex1.at/abc.html#my-anchor

Anchor



URLs & URNs

- Specialized Subtypes of URIs
- URLs (=Uniform Resource Locator) identify a resource via
  - Access mechanism (scheme) and
  - Location within computer networks
- URNs (=Uniform Resource Name) identify a resource via
  - urn:<NID>:<NID-specific-ID>
    - NID = Namespace identifier
    - Example: urn:ISBN:0130888931
  - Location independent
  - URNs are retained even if location is changed



HTTP / 1

- Protocol for Information Systems
  - Distributed, collaborative, hypermedia
  - In use by WWW initiative since 1990
- General idea: request-response
- HTTP/0.9
  - Simple protocol for raw data transfer across Internet
- HTTP/1.0 (RFC 1945)
  - Extended by allowing messages to use MIME-format
- HTTP/1.1 (RFC 2616)
  - More strict
- Standard Port: TCP 80
- Uses NVT protocol

HTTP / 2

- HTTP Request sends
  - Request method (GET,POST, ...)
  - URI (what is requested)
  - Protocol version
  - MIME-like message
    - Request modifiers
    - Client information
    - Body content
  - Generic syntax: "Method Request-URI HTTP-Version"

HTTP / 3

- HTTP Response
  - Status line
    - including message protocol version
    - Success or error code
  - MIME-like message
    - Server information
    - Entity metainformation (content-type, length, date of modification, ...)
    - Entity-body content

HTTP / 4 – Request methods

- GET
  - Retrieve information identified by Request-URI
  - May refer to a process instead to a data entity
    - See Dynamic Web
  - Conditional GET
  - if request message contains additional header fields
    - Eg. If-Modified Since, If-Match, If-None-Match, If-Range
    - Goal to reduce bandwidth
- HEAD
  - Like GET but does not return message-body
  - HTTP header identical

HTTP / 5 – Request methods

- POST
  - Requests entity enclosed in request as additional item for entity identified in Request-URI
  - URI determines handler for the post
  - Examples
    - Annotation of existing resources
    - Posting a message to bulleting boards, newsgroups, ...
    - Providing a block of data, such as the result of submitting a form, to a data-handling process
    - Extending a database through append operation
  - Actual Function determined by server
  - Response contains result of the action

HTTP / 6 – Request methods

- OPTIONS
  - Communication options availabe on the request/response chain identified by URI-Request
- PUT
  - Enclosed entity shall be stored under supplied Request-URI
- DELETE
  - Delete resource identified by Request-URI
- TRACE
  - Debugging method
- CONNECT
  - For proxies to dynamically switch being a tunnel (SSL)

HTTP – Status Codes

- Informational 1xx
  - Prior regular response
  - If unexpected May be ignored
  - Proxies must forward 1xx responses
  - 100 Continue
    - Client SHOULD continue with its request
- Successful 2xx
  - Request successful
  - 200 OK
  - 201 Created, 202 Accepted,…

HTTP - Status Codes

- Redirection 3xx
  - Further actions need to be taken by user to fulfill request
  - 301 Moved Permanently
    - New URI given in Location field of response
    - If possible client shall change link
  - 302 Found
    - New URI given in Location field of
  - 303 See Other
    - Similar to 302 but different URI should be retrieved with GET
    - Primarily to allow output of POST-activated script to redirect user agent
  - 304 Not Modified
    - For conditional GET requests

HTTP – Status Codes

- Client Error 4xx
  - 400 Bad request
  - 401 Unautorized
  - 402 Forbidden
    - Authorization won't help, shall not be repeated
  - 404 Not Found
    - No match found for Request-URI
  - 408 Request Timeout
  - 410 Gone
    - Resource no longer at server

HTTP – Status codes

- Server Error 5xx
  - 500 Internal Server Error
  - 501 Implementation
  - 503 Service Unavailable
    - Overloading of server
  - 505 HTTP Version Not supported

HTTP – Persistent Connections

- HTTP connection closed after one request
  - Assumption that client has more requests from same server
    - Standard in HTTP/1.1: persistent connection desired
  - Controlled with Header field
    Connection: close / keep-alive header
  - Server time-out closes connection automatically
- Advantages
  - Opening/closing fewer TCP connections
    - CPU time saved in routers and all participating hosts
    - Fewer packets caused by TCP opens
  - HTTP requests/responses pipelined
    - Client make multiple requests on same TCP connection without waiting for a response
  - Latency of subsequent requests reduced
    - No time spent in TCPs connection opening handshake

HTTP State Management

- HTTP Sessions to manage state
  - HTTP is stateless
    - Server Requires HTTP session to maintain variables for one user
  - Server manages variables for each session
  - Session-ID used to identify session in requests
- Identification of session
  - URL-Rewriting
    - Appends sessionID at request URI
    - http://www.example.com?sessionID=SID1234
  - HTML Hidden Field
    - Special field in HTML forms
    - <input type="hidden" name="sessionID" value="SID1234"/>
  - Cookies
    - Additional Request-Header-Field
      - Cookie: $Version="1"; sessionID="SID1234"
    - Cookie generated by server
    - Sent to user agent in response field
      - Set-Cookie2: $Version="1"; sessionID="SID1234"

HTTP Authentication

- Methods to authenticate users
  - Restrict access to resources
- Not secure unless used with external secure system (eg. SSL)
- Based on challenges
  - Server poses a challenge to client
  - Client has to response with correct answer
- Restriction is based on realms
  - String value
  - Defines/Names protection space (=realm)
    - = Set of documents

HTTP Authentication

- C: requests protected resource
- S: 401 Unauthorized
  - WWW-Authenticate header field includes at least one challenge that must be fulfilled by client
- C: Authorization header field in request
  - Contains credentials containing authentication information for a realm
- Server responds with resource

Basic Authentication

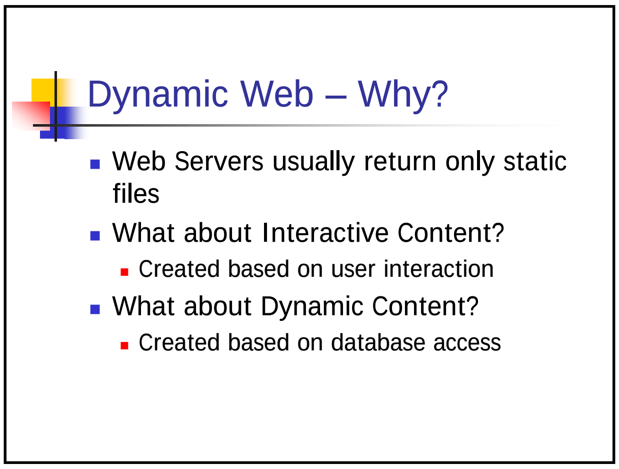
- Client identifies itself with UserID & Password
- Challenge: "Basic" realm
  - WWW-Authenticate: Basic realm="WaynesWorld"
- Credentials
  - "UserID:Password" base64 encoded
  - Authorization: Basic XYZ1235456==
- Weak
  - Problem: Base64 bijective
    - Inverse application of base64 algorithm leads to Password

Digest Authentication

- Challenge
  - contains a "nonce" value
- Valid response contains a checksum
  - Username + Password + nonce + HTTP method + Request-URI
  - Default uses MD5 checksums (128bit)
- Password never sent in the clear
- Quality of Protection (qop)
  - Different protection levels
    - Authentication, Integrity checking, Confidentiality checking

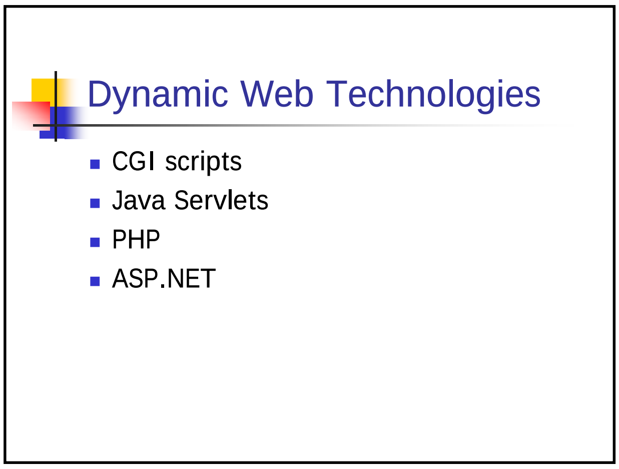
Digest Authentication / 2

- WWW-Authenticate: Digest
  - realm="WaynesWorld",
  - nonce="dcd98b1234567890acd23467",
  - opaque="12345",
- Authorization: Digest username="Wayne"
  - realm="WaynesWorld",
  - nonce="dcd98b1234567890acd23467",
  - uri="/index.html",
  - response="67890abcdef1234567890ab"



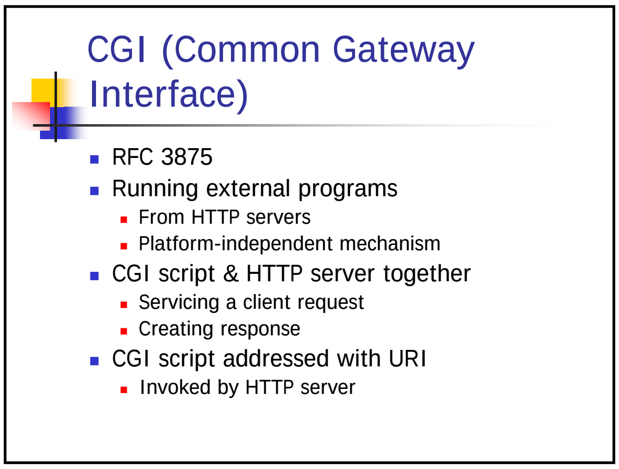
Dynamic Web – Why?

- Web Servers usually return only static files
- What about Interactive Content?
  - Created based on user interaction
- What about Dynamic Content?
  - Created based on database access



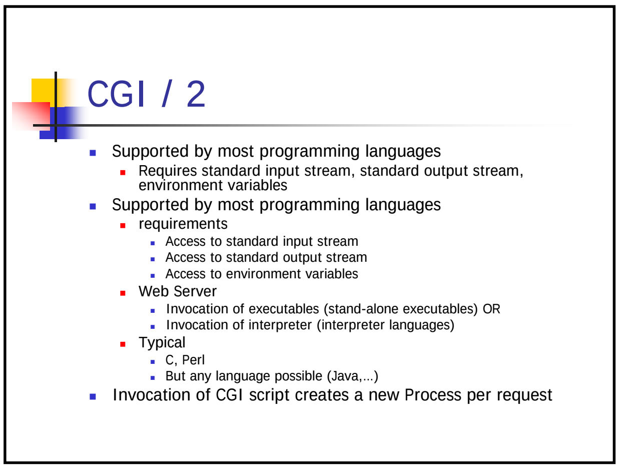
Dynamic Web Technologies

- CGI scripts
- Java Servlets
- PHP
- ASP.NET



CGI (Common Gateway Interface)

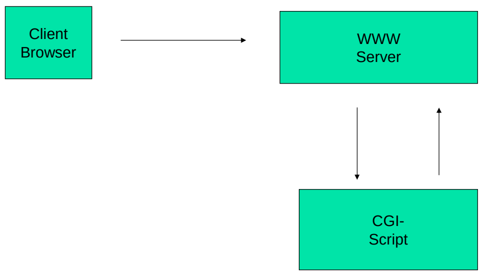
- RFC 3875
- Running external programs
  - From HTTP servers
  - Platform-independent mechanism
- CGI script & HTTP server together
  - Servicing a client request
  - Creating response
- CGI script addressed with URI
  - Invoked by HTTP server



CGI / 2

- Supported by most programming languages
  - Requires standard input stream, standard output stream, environment variables
- Supported by most programming languages
  - requirements
    - Access to standard input stream
    - Access to standard output stream
    - Access to environment variables
  - Web Server
    - Invocation of executables (stand-alone executables) OR
    - Invocation of interpreter (interpreter languages)
  - Typical
    - C, Perl
    - But any language possible (Java,...)
- Invocation of CGI script creates a new Process per request

CGI / 3



CGI / 4 – Exampel in C

```
void main(void)
{
  printf("Content-type: text/html\r\n");
  printf("\r\n");
  printf("Hello world!<br>\r\n");
  exit(0);
}
```

Fast-CGI

- CGI performance problem:
  - Many requests require multiple processes
  - Initialization of connections/resources (database)
- FastCGI
  - Script remains in memory (via endless loop)
  - Requires Predefined protocol/API for communication with HTTP server
    - Standard CGI uses just StdIn/StdOut

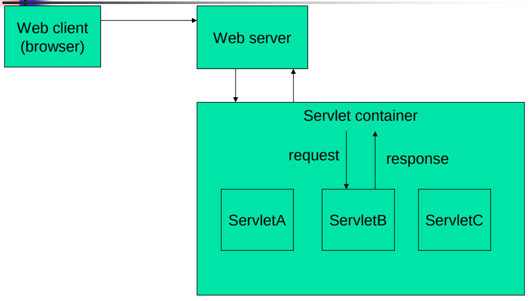
Fast-CGI / 2

```
void main(void)
{
  int count=0;
  while(FCGI_Accept() >= 0) {
      printf("Content-type: text/html\r\n");
      printf("\r\n");
      printf("Hello world!<br>\r\n");
  }
  exit(0);
}
```

Java Servlets

- Web component
  - implemented in Java
    - Implements interface javax.servlet.Servlet
- Generates dynamic content
- Managed by a servlet engine (container)
  - Web server extensions
- Request/response paradigm
  - Interaction with Web clients

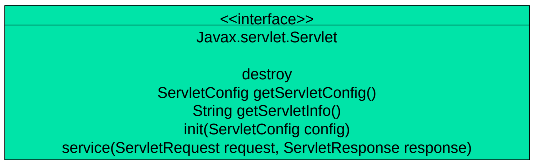
Request/response Interaction



Servlet characteristics

- Much faster than CGI scripts (in general)
  - different process model is used
- Standard API supported by many Web servers
- Supports Java and ist API's
  - Server sets on Java bytecode
  - not interpreted

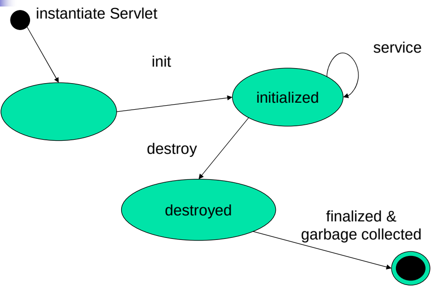
Servlet interface / 1


<<interface>>
Javax.servlet.Servlet
destroy
ServletConfig getServletConfig()
String getServletInfo()
init(ServletConfig config)
service(ServletRequest request, ServletResponse response)

Servlet implementation

- Server implements servlet interface
- Typically by inheriting from (predefined) implementation classes
  - GenericServlet
  - HttpServlet

Servlet lifecycle



Request Handling

- Through Service method
- ServletRequest object used
- Concurrent requests to same servlet
  - Concurrent execution of service method on different threads
- HTTP specific Request Handling
  - HttpServlet adds HTTP specific methods
    - primarily doGet & doPost,
    - doPut, doDelete, doHead, doOptions, doTrace
    - getParameterXXX methods provide
      - from URI query string and POST-ed data
    - getHeaderXXX methods

Response Generation

- By using methods of ServletResponse object
- Manual generation of any response
- HttpServletResponse interface
  - sendRedirect
  - sendError

Servlet example

```
public void doGet(HttpServletRequest req,
   HttpServletResponse res) throws ...
{
   res.setContentType("text/html");
   PrintWriter out= res.getWriter();

   out.println("<HTML>");
   out.println("<HEAD>");
...
   out.close();
}
```

Other Dynamic Web Technologies / 1

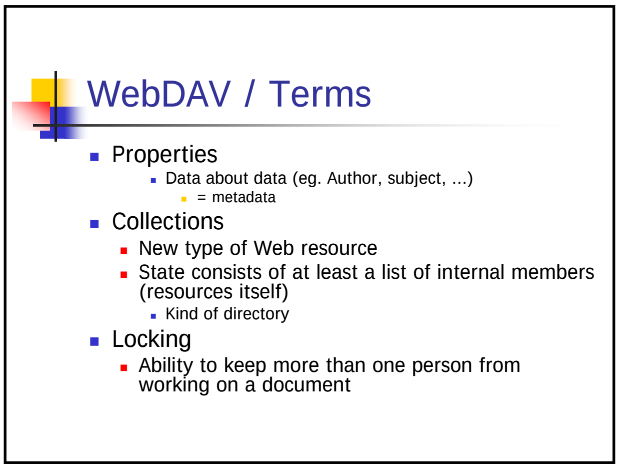
- Web Server Extensions
  - Based on callbacks
  - Example: Apache Modules
  - Example: ISAPI (Internet Server API = MS Internet Information Server)
- JSP (Java Server Pages)
  - Embeds Java code within HTML code
  - Usually compiled to servlet code
  - Taglibs = new HTML tags that contain functionality
- JSF (Java Server Faces)
  - Component model for JSP and Servlets
  - Allows construction of web pages based on prebuilt JSF components
- PHP (=Pre-HyperText Preprocessor)
  - Scripting Language used on Server
  - Embedded in HTML code
  - Performance very good (Zend engine)
  - Most frequently used technology for dynamic Web applications today

Other Dynamic Web Technologies / 2

- ASP (Active Server Pages)
  - Interpreted Scripting Language used on Server
    - Either VBScript or JavaScript
  - Embedded in HTML code
  - Builds on MS COM components
- ASP.NET
  - .NET based (not interpreted)
  - Similar to Servlets and JSP
    - Has nothing to do with ASP

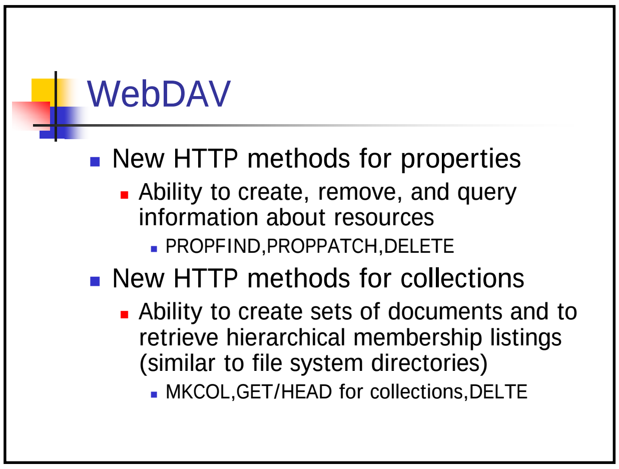
WebDAV

- Digital Authoring & Versioning (RFC 2518)
- Extends HTTP
  - Authoring of documents via HTTP
    - Directly at web server
      - Instead of using FTP
  - Provides kind of file system
      - Accessible in the Internet
    - HTTP URL namespace model
    - Accessible via HTTP (hence, Internet)
    - Platform independent
- Method parameter information
    - Either in HTTP header (like in HTTP/1.1)
    - Or Encoded in XML request entity body



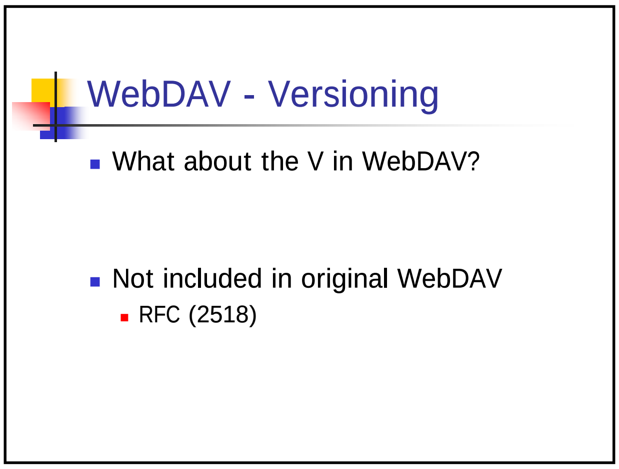
WebDAV / Terms

- Properties
  - Data about data (eg. Author, subject, ...)
    - = metadata
- Collections
  - New type of Web resource
  - State consists of at least a list of internal members (resources itself)
    - Kind of directory
- Locking
  - Ability to keep more than one person from working on a document



WebDAV

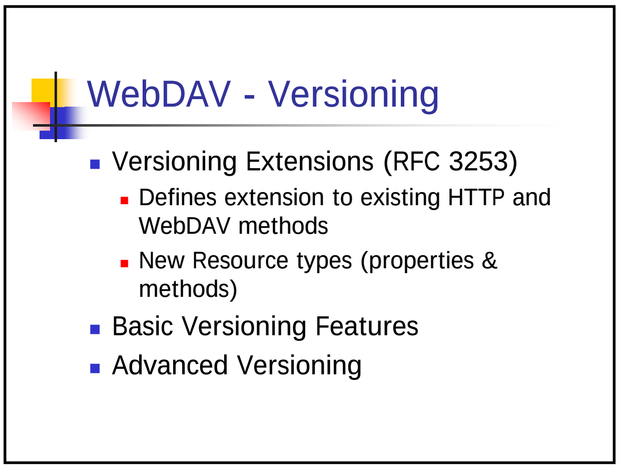
- New HTTP methods for properties
  - Ability to create, remove, and query information about resources
    - PROPFIND,PROPPATCH,DELETE
- New HTTP methods for collections
  - Ability to create sets of documents and to retrieve hierarchical membership listings (similar to file system directories)
    - MKCOL,GET/HEAD for collections,DELTE



WebDAV - Versioning

- What about the V in WebDAV?

- Not included in original WebDAV
  - RFC (2518)



WebDAV - Versioning

- Versioning Extensions (RFC 3253)
  - Defines extension to existing HTTP and WebDAV methods
  - New Resource types (properties & methods)
- Basic Versioning Features
- Advanced Versioning

WebDAV – Basic Versioning

- Goals
  - Put a resource under version control
  - Determine whether a resource is under version control
  - Determine whether a resource update will automatically be captured as a new version
  - Create and access distinct versions of a resource

WebDAV – Basic Versioning

- Methods
  - VERSION-CONTROL
    - Create a version-controlled resource at Request-URI
  - REPORT
    - Returns information about a resource (infos about multiple versions)
  - CHECKOUT
    - Applied to a checked-in version-controlled resource to allow modifications
  - CHECKIN
    - Applied to a checked-out version-controlled resource to produce a new version

WebDAV – Advanced Versioning

- Goals
  - Parallel development
  - Configuration management of sets of web resources
  - Similar what CVS,Subversion,Perforce,etc can already do
- Methods
  - MERGE simultaneous changes

WebDAV - Extensions

- WebDAV Ordered Collections Protocol
  - RFC 3648
  - Server-side support for ordering of collection members
  - Client may change order
- WebDAV Access Control Protocol
  - RFC 3744
  - Permits clients to read and modify access control lists with permissions for resources on the server

WebDAV – Request Sample

```
PROPFIND /mydocs/thebible HTTP/1.1
Host: www.server.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"?>
<D:propfind xmlns:D="DAV:">
    <D:prop xmlns:R="http://www.server.com/mydocs/>
          <R:author/>
          <R:creation-date/>
    </D:prop>
</D:propfind
```

Retrieves Named Properties

WebDAV – Response Sample

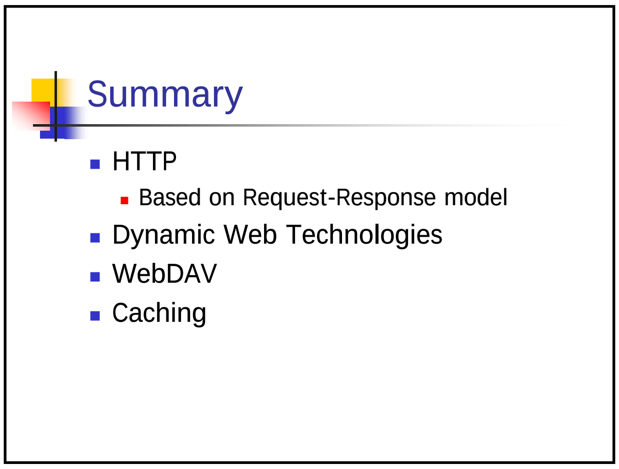
```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
    <D:response>                <D:href>http://www.server.com/mydocs/thebible.doc</D:href>
                          <D:propstat>
                                      <D:prop xmlns:R=http://www.server.com/mydocs/>
                                      <R:author>
                                                  <R:Name>unknown</R:Name>
                                      </R:author>
                                      ...
                          <D:status>HTTP/1.1 200 OK</D:status>
                          </D:propstat>
              </D:href>
    </D:response>
</D:multistatus>
```

WWW Caching

- Browser cache
  - Included in Web browser
  - Checks if representation stored on local disc is up-to-date
- Proxy cache
  - Larger scale (100-1000s users)
  - Good at reducing latency and network traffic
  - For Popular representations used in departments/companies, …
  - Examples
    - Squid (www.squid-cache.org),
    - MS Internet Security and Acceleration Server
- Gateway cache
  - To make sites themselves more scalable
  - Eg. Akamai

WWW Caching

- HTML Meta Tag
  - META No-cache
  - Problem: not all browsers support it
- HTTP Header
  - Expires: Thu, 2 Jun 2005 13:10:00 GMT
    - Good for files that change rarely
    - Clock synchronisation of WebServer and cache
  - Cache-Control response Header
    - no-store, max-age (similar to expires but relative)
    - no-cache (cache submits request to server)
- Internet Cache Protocol (RFC 2186, RFC 2187)
  - Synchronisation of Caches
  - More lightweight than HTTP
    - On miss a cache submits an ICP request to cache siblings
    - Returns HITs and/or MISSes
    - Original cache uses these returns to resolve its own miss (via HTTP)



Summary

- HTTP
  - Based on Request-Response model
- Dynamic Web Technologies
- WebDAV
- Caching