Chapter 3: Uniform Resource Identifiers

References:

- Erik Wilde: World Wide Web — Technische Grundlagen (in German). Springer, 1999, ISBN 3-540-64700-7, 641 Seiten.
- NCSA Mosaic Team: A Beginner's Guide to URLs. [http://archive.ncsa.uiuc.edu/demoweb/url-primer.html]
- T. Berners-Lee, R. Fielding, L. Masinter: Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, August 1998, 40 pages.
- J. Kunze: Functional Recommendations for Internet Resource Locators. RFC 1736, February 1995, 10 pages.
- K. Sollins, L. Masinter: Functional Requirements for Uniform Resource Names. RFC 1737, December 1994, 7 pages.
- T. Berners-Lee, L. Masinter, M. McCahill: Uniform Resource Locators (URL). RFC 1738, December 1994, 24 pages. In part superseded by RFC 2396.
- R. Moats: URN Syntax. RFC 2141, May 1997, 8 pages.
- P. Hoffman, L. Masinter, J. Zawinski: The mailto URL scheme. RFC 2368, July 1998.
- Tim Berners-Lee: Cool URIs don't change. [http://www.w3.org/Provider/Style/URI]
- Uniform Resource Identifier (URI) Schemes (April 2002) [http://www.iana.org/assignments/uri-schemes]
- Dan Connolly: Addressing Schemes. [http://www.w3.org/Adressing/schemes]

Objectives

After completing this chapter, you should be able to:

- explain the use of schemes in the URI definition.
- enumerate some schemes and explain the syntax of URIs for these schemes.
- analyze HTTP URIs for syntactic correctness and explain the various parts.
- explain the advantages and syntax of relative URIs.
- explain the relationship between URIs, URLs, and URNs.

Overview

Motivation (1)

- The WWW is a hypermedia system. It contains:
  - ◇ Resources (multimedia documents)
  - ◇ Links between these resources.
- Uniform Resource Identifiers (URIs) uniquely identify a resource and are used as links.
- URIs are used e.g. in
  - ◇ hypermedia documents (e.g. HTML web pages),
  - ◇ bookmark folders managed by a browser,
  - ◇ printed documents (books).

Motivation (2)

- In earlier times, a great variety of different network protocols were used, e.g. FTP, Gopher, News.

  Today, most documents are available via HTTP.

- It was essential for the success of the web that
  - ◇ it offered a uniform interface to many different kinds of network services,
  - ◇ links in HTML pages retrieved via HTTP could not only point to other HTML pages on HTTP servers (there were too few at the beginning), but to (more or less) any resource on the net.

Motivation (3)

- "If it's out there, we can point to it."

  NCSA Mosaic Team: A Beginner's Guide to URLs.

- In earlier times, in order to retrieve a document via FTP, one had to call the `ftp` program, log into a server, enter some `ftp` commands (e.g. `dir`, `cd`, `get`) in order to fetch the file, and log out.
- In order to get a News message, one had to use a different program and different commands.
- Now, if one has URIs for the two documents, it requires only a single mouse click.

Schemes (1)

- In order to be universal and extensible, URIs start with the naming scheme like "`http`" that identifies the type of the resource and the access method.
- The scheme is followed by a colon ":". The rest of the URI is interpreted according to the scheme.

$$\underbrace{\texttt{http}}_{\text{scheme}} : \underbrace{\texttt{//www.uni-giessen.de/hrz/}}_{\text{scheme-specific part}}$$

- The URI Specification (RFC 2396) only limits the character set for the scheme-specific part of the URI, otherwise it can be any string.

Schemes (2)

- The Internet Assigned Numbers Authority (IANA) manages a list of officially registered schemes at

  [`http://www.iana.org/assignments/uri-schemes`]

- Currently, it contains 38 schemes, e.g.
  - ◇ `ftp`: File Transfer Protocol.
  - ◇ `http`: Hypertext Transfer Protocol.
  - ◇ `gopher`: The Gopher Protocol.
  - ◇ `mailto`: Electronic Mail Address.
  - ◇ `news`: USENET news.

Schemes (3)

- More examples of schemes:
  - ◇ `nntp`: USENET news using NNTP access.
  - ◇ `telnet`: Interactive session.
  - ◇ `wais`: Wide area information servers.
  - ◇ `file`: Host-specific file names.
  - ◇ `imap`: Internet message access protocol.
  - ◇ `nfs`: Network file system protocol.
  - ◇ `pop`: Post Office Protocol v3
  - ◇ `data`: Data

Schemes (4)

- Still more examples of schemes:
  - ◇ `tel`: Telephone
  - ◇ `fax`: Fax
  - ◇ `modem`: Modem
  - ◇ `ldap`: Lightweight Directory Access Protocol
  - ◇ `https`: Hypertext Transfer Protocol Secure
- The IANA list contains for each scheme a reference to an RFC that defines syntax and semantics of the scheme-specific part.

Schemes (5)

- An inofficial list managed at the W3C already contains 84 schemes with references to definitions:

  [`http://www.w3.org/Adressing/schemes`]
- It states that probably Microsoft uses 20–40 private schemes (2–3 being added daily) and WebTV 24.
- Of course, it depends on the browser which schemes it implements.

  Some browsers are extensible: One can tell them which program should be called for a given scheme that is not already built in. E.g. the W3C list above contains registry settings for the Internet Explorer.

Overview

Characters in URIs (1)

- URIs use only a limited character set because:
  - ◇ URI characters should exist in all character sets and on all keyboards (US-ASCII).
  - ◇ Control characters are not used because they cannot be printed in books.
  - ◇ Spaces and line breaks are not used in URIs because they may be inserted when the URI is printed or transmitted.
  - ◇ The characters “"”, “<”, “>” are excluded because they are used as delimiters around URIs.

Characters in URIs (2)

- Other excluded characters:
  - ◇ “#” separates a fragment identifier from a URI, therefore it cannot appear in the URI itself.
  - ◇ Also the characters “{”, “}”, “|”, “\”, “^”, “[”, “]”, “`” are not used in URIs for various less strict reasons (e.g. some gateways modify them).
- The character “%” always has a special meaning in URIs, the characters “;”, “/”, “?”, “:”, “@”, “&”, “=”, “=”, “+”, “$”, “,” might have a special meaning in certain parts of URIs (see below).

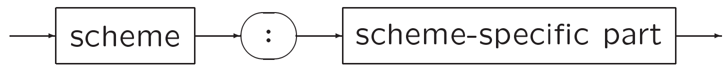
Characters in URIs (3)

- However, a URI can describe data that contains characters which cannot directly appear in URIs.
- Forbidden characters are “escaped”: They are represented by the three characters “%XY”, where XY are two hexadecimal digits.

  Besides the standard decimal digits (0...9), one can use uppercase (A...F) or lowercase (a...f) letters as “digits” with values 10 to 15.
- E.g. the percent sign itself (ASCII 37) is represented as “%25”, and “%0a” stands for the newline character (ASCII 10).

Syntax (1)

- As explained above, a URI consists of a scheme and a scheme-specific part, separated by a colon:



- Scheme names must start with a lowercase character followed by zero or more lowercase characters, digits, hyphens “–”, plus signs “+”, or periods “.”.

  Uppercase characters should be automatically mapped to lowercase.

Syntax (2)

- The scheme-specific part can be
  - ◇ “opaque” (not starting with “/”): The URI specification does not specify the format of these.
  - ◇ “hierarchical” (starting with “/”): The specification defines certain parts and relative URIs.
- Opaque parts are any sequence of uppercase and lowercase letters, digits, and these special characters: “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”, “,”, “-”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.

  The first character may not be a slash “/”.

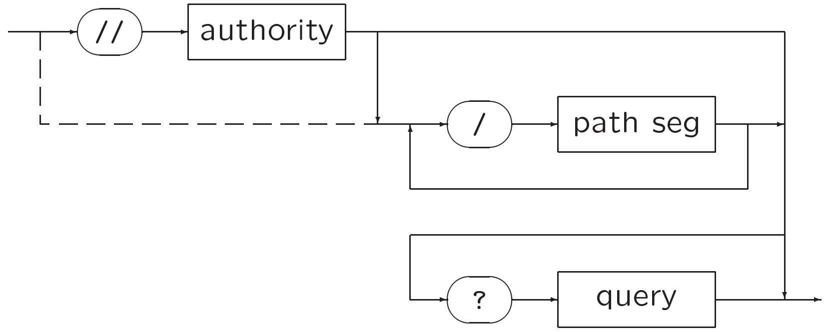
Syntax (3)

- Hierarchical URIs consists of a network path or an absolute path, optionally followed by a query.
- A network path starts with “//”, followed by an “authority” and optionally an absolute path.
- An absolute path starts with “/” and can consist of several segments, separated with “/”.
- The query is marked with “?”.
- It depends on the scheme which parts are used.

  The syntax defined in the URI specification is in part necessary for defining relative URIs, and helps to unify the syntax of some schemes.

Syntax (4)

Hierarchical Scheme-Specific Part:



Syntax (5)

- Path segments can consist of uppercase and lowercase letters, digits, and these special characters: “:”, “@”, “&”, “=”, “+”, “$”, “,”, “-”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.
- Optionally, each path segment can be followed by one or more parameters, each separated with “;”.
- Although the path looks basically like a UNIX pathname for a file, the named resource may not directly correspond to a file on the server, or may be stored in a different location.

Syntax (6)

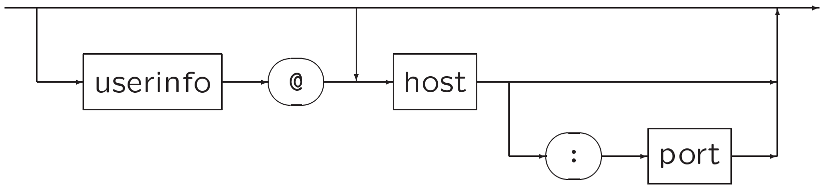
- The “authority” part describes who administers the namespace described by the path:
  - ◇ For most current schemes, it is simply the host on which the server software runs (plus possibly port and maybe a login name).

    Whoever runs the server has authority over the paths on it.
  - ◇ A URI scheme can also use “registry-based authorities”. Then the authority can be any string and is interpreted as defined for that scheme.

    Of course, the set of characters is restricted in order to be still able to parse URIs (e.g. in order to handle relative URIs).

Syntax (7)

Authority (Server-Based):



- The host can be a domain name like “`www.pitt.edu`” or an IP number like “`134.176.28.60`”.

  One can also write “`www.pitt.edu.`” (with a period at the end).
- The port is identified by number, e.g. “`8080`”.

Syntax (8)

- The userinfo can consist of uppercase and lowercase letters, digits, and these special characters: “;”, “:”, “&”, “=”, “+”, “$”, “,”, “-”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.
- It can e.g. be a username or have the form “username:password”.

  URIs are contained in clear text in other documents, so the password can be read by anybody who has access to the document.

Syntax (9)

- Sometimes the path specifies a program that computes the requested resource.
- Then a query can be added to the URI. It specifies parameters/arguments for the program.
- A query can be any string consisting of uppercase and lowercase letters, digits, and the special characters “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”, “,”, “-”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.

  RFC 2396 states that “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$” are reserved within the query, but does not define their meaning.

Queries (1)

- For queries, two formats are common:
  - ◇ The program has several arguments which are separated by "+".
  - ◇ The program gets a set of attribute-value pairs separated by "&", e.g.

    `FirstName=Stefan&LastName=Brass`
- Note that in HTML, the character "&" has a special meaning, one must write it as "`&`" (see below).

Queries (2)

- The usual escape mechanism applies, e.g. a real "+" is written as "%2B".
- In both forms, the plus sign is mapped to a space.

  In the first form, the space separates command line arguments.
- E.g. this is also possible:

  `Title=Fundamentals+of+the+World+Wide+Web`
- Attribute-value pairs are e.g. generated when the contents of a form are submitted to the server.

  Each input field corresponds to an attribute.

Overview

1. Motivation, Schemes
2. General URI Syntax
3. Some Specific Schemes
4. Fragment IDs, Relative URIs
5. URIs, URLs, URNs

HTTP URIs (1)

- The scheme "`http`" uses server-based authorities.
- The "userinfo" part is not used by HTTP URIs.
- Thus, all HTTP URIs start with "`http://`" followed by the domain name or IP number of the server.
- Path and query are optional.
- An HTTP-URI with all parts looks as follows:

$$\underbrace{\texttt{http}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{www.uni-giessen.de}}_{\text{host}}\texttt{:}\underbrace{\texttt{80}}_{\text{port}}\underbrace{\texttt{/hrz/}}_{\text{path}}\texttt{?}\underbrace{\texttt{abc}}_{\text{query}}$$

HTTP URIs (2)

- Only the host part is required, e.g.:

$$\underbrace{\texttt{http}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{www.uni-giessen.de}}_{\text{host}}$$

- The host can be specified with an IP number:

`http://134.176.28.60/`

- If the port is not specified, the default port 80 for the HTTP protocol is used.
- Depending on the server configuration, it might make a difference whether the path ends in a "/" or not.

FTP URIs (1)

- The scheme "`ftp`" uses server-based authorities.
- The "userinfo" and "path" parts are optional.
- The query part cannot be used in FTP URIs.
- An FTP-URI with all parts looks as follows:

$$\underbrace{\texttt{ftp}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{brass}}_{\text{user}}\texttt{:}\underbrace{\texttt{lisa}}_{\text{password}}\texttt{@}\underbrace{\texttt{ftp.x.de}}_{\text{host}}\texttt{:}\underbrace{\texttt{21}}_{\text{port}}\underbrace{\texttt{/gnu/gcc.tar}}_{\text{path}}$$

- Often, no user name and password are specified in the URI. The default for the user is "`anonymous`" and the email address of the user is sent as password.

FTP URIs (2)

- A typical FTP URL is:

`ftp://ftp.isi.edu/in-notes/rfc959.txt`

- FTP has different transfer modes e.g. for ASCII data and for binary data.

  In the ASCII case, the server transforms line ends into CR LF sequences, and the client translates them back into the local convention, e.g. only LF on UNIX systems. A binary file would be destroyed by this transformation. Normally, the browser guesses the transfer mode from the file extension in the URI. However, one can add a type parameter to the path: "`ftp://ftp.isi.edu/in-notes/rfc959.txt;type=I`". The type "`I`" (image) means binary, "`AN`" means ASCII without special treatment of print control characters like FF.

File URIs (1)

- File URIs can be used to access files on the local computer without going through a web server.

  The web server is normally configured such that it permits to access only a small subset of the files on the computer (those that are intended to be read by the public). Other files to which one has access as user of the local system can be read into the browser with file URIs.

- On my PC, no web server runs, but I can read HTML files stored on my disk:

`file:///C:/Stefan/courses/www02/doc/cool_uri.html`

File URIs (2)

- E.g. Oracle documentation on our UNIX systems:
  `file:///ora8/app/product/oracle8.1.6/doc/index.html`
- File URIs use the host field, but normally it is empty or "`localhost`".

  One can specify a specific host such that if the link should be followed on another host, the user gets an error message. It might also be that some operating systems have a distributed file system that can make use of the host component. But in general, it is used very rarely. It seems that old browsers (Mosaic) understood file URIs with host as access via ftp.
- The following also works:

  `file://localhost/C:/Stefan/fireworks.xml`

Mailto URIs (1)

- Mailto URIs contain email addresses, e.g.

  `mailto:Stefan.Brass@informatik.uni-giessen.de`
- If one clicks in a browser on a link that refers to a mailto URI, the browser opens a window for composing an email and enters the given address as recipient.

  Assuming of course that the browser supports this URI scheme. Most modern browsers do, but e.g. the original Mosaic browser did not.
- Mailto URIs are an example of opaque URIs.

  Relative URIs make no sense for the mailto scheme.

Mailto URIs (2)

- The first proposal for `mailto` URIs (RFC 1738) permitted only an email address in the URI.
- This was extended by RFC 2368, which permits to specify arbitrary headers:

  `mailto:brass@acm.org?subject=URI-Test`
- One can also specify several headers and a body:

  `mailto:a@b.de?subject=s&cc=c@d.com&body=hello`
- Note that spaces in data must be escaped as "%20".

Telnet URIs

- Clicking on a telnet URI usually will open a telnet window that gives a remote login to a machine or maybe only a specific program:

  `telnet:www.informatik.uni-giessen.de:80`
- In theory, it is possible to specify username and password, but storing passwords in clear text, even locally, is a bad idea:

$$\underbrace{\texttt{telnet}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{brass}}_{\text{user}}\texttt{:}\underbrace{\texttt{lisa}}_{\text{password}}\texttt{@}\underbrace{\texttt{unsecure.com}}_{\text{host}}\texttt{:}\underbrace{\texttt{23}}_{\text{port}}$$

Overview

1. Motivation, Schemes
2. General URI Syntax
3. Some Specific Schemes
4. Fragment IDs, Relative URIs
5. URIs, URLs, URNs

Fragment Identifiers (1)

- An URI specifies a document, i.e. typically the contents of a browser window.
- Depending on the documents' media type, it might be possible to instruct the browser to automatically scroll to a certain position in the document.

  I.e. the browser automatically jumps to a certain section of the document, instead of always showing the document from the beginning. However, the browser nevertheless has loaded the complete document, one can scroll up to the beginning if necessary.
- In this way, one can not only refer to complete documents, but also to parts of it.

Fragment Identifiers (2)

- HTML documents support this feature, but one can jump only to positions that were explicitly named by the author of the document.
- The fragment identifier is formally not part of the URI, but the URI specification defines the notion of a "URI Reference" which consists of a URI and a fragment identifier: separated by "`#`":

$$\underbrace{\texttt{http://www.x.org:80/a/b?c}}_{\text{URI}}\,\texttt{\#}\,\underbrace{\texttt{section2}}_{\text{Fragment ID}}$$

Fragment Identifiers (3)

- Fragment identifiers can be any sequence of uppercase and lowercase letters, digits, and the characters "`;`", "`/`", "`?`", "`:`", "`@`", "`&`", "`=`", "`+`", "`$`", "`,`", "`-`", "`_`", "`.`", "`!`", "`~`", "`*`", "`'`", "`(`", "`)`".
- Fragment identifiers depend on the media type and not the URI scheme: When HTML files are accessed via ftp, they also apply.
- For XML, XPath was defined to be able to link to any part of a document even without support in the document.

Relative URIs (1)

- Often, a large document is structured into several web pages that reference each other.
- Today, many web pages contain pictures. Each picture has its own URI and is referenced from the main HTML file.
- Normally, related web pages and supporting image files are stored on the same server, often even in the same directory.
- It is possible to reference resources relative to the URI of the current resource.

Relative URIs (2)

- E.g., suppose the URI of the current document is:

  `http://www.sis.pitt.edu/~sbrass/dd00/index.html`
- Then the relative URI "`h1.pdf`" refers to

  `http://www.sis.pitt.edu/~sbrass/dd00/h1.pdf`
- Then the relative URI "`../db99/c3.ps`" refers to

  `http://www.sis.pitt.edu/~sbrass/db99/c3.ps`
- An absolute URI starts with a scheme name (consisting of letters, digits, "-", "+", ".") followed by a colon. Relative URIs do not contain a scheme.

Relative URIs (3)

- The basic algorithm for this typical case is:
  - ◇ Delete from the given base URI everything after the last slash.
  - ◇ For every ".." at the beginning of the relative URI, delete one more segment at the end of the base URI.
  - ◇ Append the rest of the relative URI (without the "..") to whatever remains from the base URI.
- The general algorithm is contained in RFC 2396, Section 5.2.

Relative URIs (4)

- An empty URI refers to the document itself.

  This is probably only interesting together with a fragment identifier.
- It is also possible to use an absolute path as URI. E.g. `/academics/courses/` refers to

  `http://www.sis.pitt.edu/academics/courses/`
- One can also use the authority portion, but that is probably never used in practice.

  In this case the path must be absolute, only the typing effort for the scheme is saved.

Relative URIs (5)

- Query and fragment identifier are never inherited from the base URI to the new URI. They must be explicitly specified in the relative URI.
- Relative URIs are useful because:
  - ◇ They save some typing effort for the document author.
  - ◇ One can move/copy the entire document consisting of several linked resources to another machine/directory without need to change the URIs in the files.

Overview

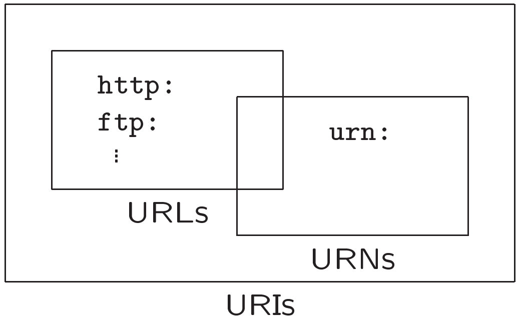
URIs, URLs, URNs (1)

- The formal term used in specifications is "Uniform Resource Identifier" (URI).
- However, people usually talk about "Uniform Resource Locators" (URLs). This is only a subset of URIs, but basically all there is today.
- A URL describes a network address where the resource is available.
- This is a problem if the resource is mirrored, and one should get it from the nearest mirror.

URIs, URLs, URNs (2)

- Also sometimes machines and domains are given up, or the institution that runs the server does not want to support a resource further, but the documents continue to exist in other places.
- Finally, also resources that are not (yet) available electronically need to be referred to (such as books).
- Uniform Resource Names (URNs) are being developed to solve these problems.
- They will be persistent and location-independent.

URIs, URLs, URNs (3)



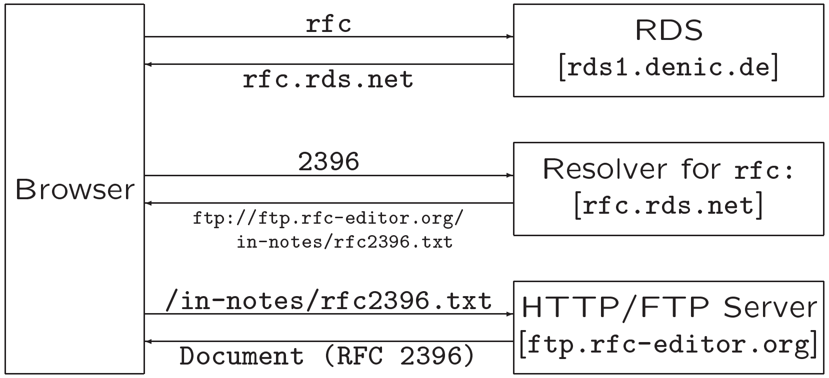
Picture similar to one shown in [http://www.w3.org/Addressing/]

URIs, URLs, URNs (4)

- URNs use the scheme “`urn:`”.
- This is followed by a namespace identifier, followed by a colon, followed by a namespace-specific string.
- In order to get the resource, one needs to
  - ◇ First query a Resolver Discovery Service (RDS) with the namespace identifier to find a resolver.
  - ◇ Then query a resolver for the namespace with the namespace-specific string to get a URL.
  - ◇ Then use the URL to fetch the resource.

URIs, URLs, URNs (5)

Hypothetical Example: [`urn:rfc:2396`]



Persistent URLs (1)

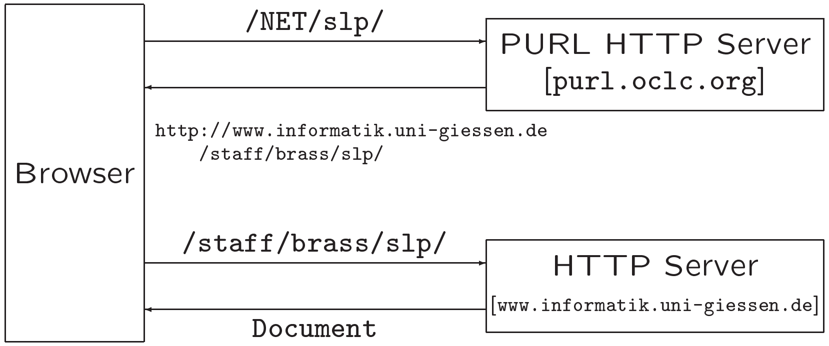
- While URNs are not yet deployed, OCLC (Online Computer Library Center) has proposed PURLs (Persistent URLs) as an intermediate step.
- The idea is again an additional level of indirection:
  - ◇ The PURL is an URL for OCLC's server, e.g.

    `http://purl.oclc.org/NET/slp/`
  - ◇ This returns an HTTP redirect message (see Part 4), in this case to

    `http://www.informatik.uni-giessen.de/staff/brass/slp/`
  - ◇ The browser automatically fetches this page.

Persistent URLs (2)

Example:



Persistent URLs (3)

- When this page moves with me to the University of Halle, I can update the URL in OCLC's database, so that now it refers to

  `http://www.informatik.uni-halle.de/~brass/slp/`

- Once URNs work, there will probably be a PURL URN scheme, i.e. the URN will probably be

  `urn:purl:/NET/slp/`

- Currently (April 2003) 568 642 PURLs exist.

  Instead of or in addition to reserving one's own domain, one should also reserve one's own PURL.

Design of Stable URIs (1)

- The WWW contains many references to documents under URIs that no longer exist (dangling links).
- Often the document itself was not deleted, it was only moved to a different place.
- This is a problem for
  - ◇ users, who click on a link, wait for some time, and then get an error message.
  - ◇ authors of web pages, who are forced to constantly update their documents, only because referenced documents move.

Design of Stable URIs (2)

- While on one's personal PC, one can move files around as one likes, URIs become automatically public (once their appear in a document that is already known to the public).

  And furthermore the document described by the URI also becomes public unless it is specially access-protected.

- One should carefully select URIs in such a way that they can be supported for the next 100 years.

  URIs might also be printed in books, and our libraries contain some very old books. Unless the company goes out of business or the university is closed, the progress in storage technology will make it easy to keep the old web pages.

Design of Stable URIs (3)

- An arbitrary mapping between URIs and files on the disk of the web server can be defined in the web server configuration.

  Although there might be good reasons for changing the file structure on the server, this does not mean that URIs have to change.

- One should distinguish between URIs for
  - ◇ The latest version of a document or what we currently mean by a subject.
  - ◇ Archival URIs that mean one specific document that might be only of historical interest.

Design of Stable URIs (4)

- "Designing [URIs] mostly means leaving information out." [Tim Berners-Lee: Cool URI's don't change.]
- E.g. the following should not be part of an URI:
  - ◇ The name of the person who manages the page.
  - ◇ The program/technology that is currently used to compute the web page (e.g. `/cgi-bin/`).
  - ◇ File name extension: Maybe HTML will no longer be used in 20 years. E.g. converted to PDF.
  - ◇ Status, who may access: This might change.