Teil 14: URIs — Uniform Resource Identifier

Literatur:

- 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 In part superseded by RFC 2396.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]

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- den Zweck von Schemata (schemes) im Aufbau von URIs erklären.
- einige URI Schemata aufzählen.
- HTTP URIs auf syntaktische Korrektheit prüfen und die verschiedenen Bestandteile erklären.
- Vorteile und Syntax von relativen URIs erklären.
- die Beziehung zwischen URIs, URLs und URNs erklären.

Inhalt

Motivation (1)

- Das WWW ist ein Hypermedia System. Es enthält:
  - ◇ Resourcen (Multimedia Dokumente)
  - ◇ Verweise (Links) zwischen diesen Dokumenten.
- Uniform Resource Identifiers identifizieren die Resourcen eindeutig und werden als Links verwendet.
- URIs finden sich z.B. in
  - ◇ Hypermedia Dokumenten (HTML Webseiten),
  - ◇ Bookmark Dateien (vom Browser verwaltet),
  - ◇ gedruckte Dokumente (Bücher).

Motivation (2)

- In früheren Zeiten wurden viele verschiedene Netzwerkprotokolle benutzt, z.B. FTP, Gopher, News.

  Heute sind die meisten Dokumente über HTTP verfügbar.

- Es war entscheidend für den Erfolg des Webs, daß
  - ◇ es eine einheitliche Schnittstelle für viele verschiedene Arten von Netzwerk-Diensten anbot,
  - ◇ Verweise in HTML Seiten nicht nur auf andere HTML Seiten (auf HTTP Servern) verweisen konnten (es gab am Anfang zu wenige), sondern mehr oder weniger auf jede Resource im Netz.

Motivation (3)

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

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

- Wenn man früher ein Dokument von einem FTP Server abrufen wollte, mußte man das Programm `ftp` aufrufen, sich einloggen, Kommandos wie `dir`, `cd`, `get` eingeben, und wieder ausloggen.
- Zum Lesen einer News Nachricht gab es ein anderes Programm mit anderen Kommandos.
- Hat man heute dagegen URIs für die beiden Dokumente, so braucht es jeweils nur einen Mausklick.

URI-Schemata (1)

- Damit die Syntax universell und erweiterbar ist, beginnen URIs mit einem Namensschema (z.B. `http`): identifiziert Resourcen-Typ und Zugriffsmethode.
- Nach dem Schema kommt ein ":". Aufbau und Interpretation des Rests hängt dann vom Schema ab.

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

- Die URI Specifikation (RFC 2396) schränkt nur die möglichen Zeichen für den Schema-spezifischen Anteil ein. Ansonsten kann die Syntax beliebig sein.

URI-Schemata (2)

- Die Internet Assigned Numbers Authority (IANA) verwaltet eine Liste offiziell registrierter Schemata:

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

  Die Liste enthält für jedes Schema einen Verweis auf einen RFC der Syntax und Semantik des schema-spezifischen Anteils definiert.

- Sie enthält im Moment 38 Schemata, z.B.
  - ◇ `ftp`: File Transfer Protocol.
  - ◇ `http`: Hypertext Transfer Protocol.
  - ◇ `https`: Hypertext Transfer Protocol Secure

URI-Schemata (3)

- Beispiele für URI-Schemata, Forts.:
  - ◇ `mailto`: EMail Adresse.
  - ◇ `news`/nntp: USENET news.
  - ◇ `telnet`: Interaktive Sitzung / Remote Login.
  - ◇ `file`: Dateien auf lokalem Rechner / Fileserver.
  - ◇ `data`: Daten (in der URI codiert)
  - ◇ `tel`: Telephone
  - ◇ `fax`: Fax
  - ◇ `modem`: Modem

URI-Schemata (4)

- Eine inoffizielle Liste, die vom W3C verwaltet wird, enthält schon 84 Schemata:

  [`http://www.w3.org/Adressing/schemes`]

  Dort wird auch gesagt, daß Microsoft ca. 20–40 private Schemata verwendet (ständig wachsend), und WebTV 24.

- Es hängt natürlich vom Web Browser ab, welche Schemata er versteht.

  Einige Browser sind erweiterbar: Man kann sie so konfigurieren, daß für ein neues URI-Schema ein bestimmtes Programm aufgerufen wird. Die W3C Liste enthält Registry Einstellungen für den Internet Explorer.

Inhalt

Zeichen in URIs (1)

- Die erlaubten Zeichen in URIs sind beschränkt, weil:
  - ◇ URIs sollten sich in allen Zeichensätzen und auf allen Tastaturen schreiben lassen (US-ASCII).
  - ◇ Steuerzeichen sind ausgeschlossen, weil sie sich nicht in Büchern drucken lassen.
  - ◇ Leerzeichen/Zeilenumbrüche dürfen die Bedeutung nicht ändern (also ausgeschlossen), weil sie beim Drucken in Büchern ggf. eingefügt werden.
  - ◇ “`"`”, “`<`”, “`>`” sind ausgeschlossen, weil sie als Begrenzer um URIs verwendet werden.

Zeichen in URIs (2)

- Weitere ausgeschlossene Zeichen:
  - ◇ “#” trennt den “Fragment Identifier” (s.u.) von der URI, und kann deshalb in der eigentlichen URI nicht auftreten.
  - ◇ Auch die Zeichen “{”, “}”, “|”, “\”, “^”, “[”, “]”, “‘” werden in URIs nicht verwendet.
- Das Zeichen “%” hat immer eine besondere Bedeutung in URIs, die Zeichen “;”, “/”, “?”, “:”, “@”, “&”, “=”, “=”, “+”, “$”, “,” können eine besondere Bedeutung in bestimmten Teilen von URIs haben.

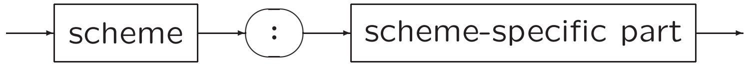
Zeichen in URIs (3)

- Man muß zwischen der externen Repräsentation und den intern gespeicherten Daten unterscheiden.
- Auch Zeichen, die in der externen Repräsentation verboten sind, können mit der Codierung “%XY” in die URI eingefügt werden, dabei sind XY zwei Hexadezimalziffern (Zahldarstellung zur Basis 16).

  Neben den normalen Ziffern (`0`...`9`), können die Buchstaben `A`...`F` (oder `a`...`f`) als Ziffern mit den Werten 10 bis 15 verwendet werden.
- Z.B. muß das Prozentzeichen selbst (ASCII 37) als “%25” codiert werden ($2*16+5$), und “%0a” ist der Zeilenumbruch (ASCII 10).

Syntax (1)

- Wie oben erläutert, besteht eine URI aus einem Schema-Namen und einen schema-spezifischen Anteil (getrennt durch “:”):



- Schema-Namen müssen mit einem Kleinbuchstaben beginnen, gefolgt von null oder mehr Kleinbuchstaben, Ziffern, und den Zeichen “–”, “+” “.”.

  Großbuchstaben sind eigentlich ein Fehler, sollten aber automatisch in Kleinbuchstaben korrigiert werden.

Syntax (2)

- Es gibt zwei Arten von schema-spezifischen Teilen:
  - ◇ “opaque” (beginnt nicht mit “/”): Die allgemeine URI-Spezifikation schränkt solche URIs nicht ein: Aufbau hängt von spezifischem Schema ab.

    Aus Sicht der URI-Spezifikation ist es eine beliebige Folge von Groß- und Kleinbuchstaben, Ziffern, und folgenden Sonderzeichen: “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”, “,”, “–”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”. Aber erstes Zeichen nicht “/”.
  - ◇ “hierarchisch” (beginnt mit “/”): Die allgemeine URI Spezifikation definiert den grundsätzlichen Aufbau sowie relative URIs.

    Die festgelegte Basis-Struktur ist nötig zur Definition relativer URIs und hilft, die Syntax einiger Schemata zu vereinheitlichen.

Syntax (3)

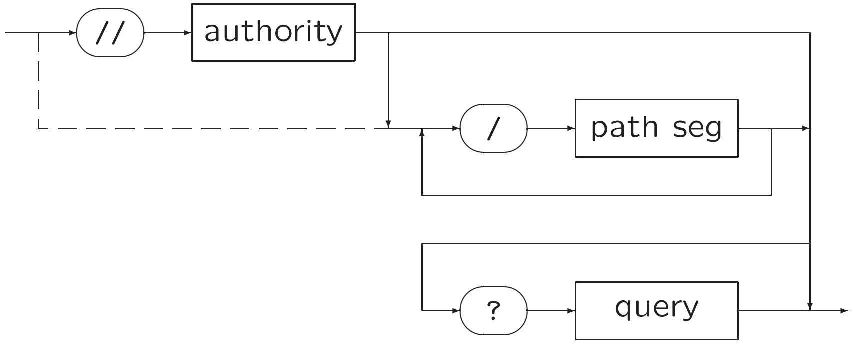
- Hierarchische URIs bestehen aus einem Netzwerk-Pfad oder einem absoluten Pfad, optional gefolgt von einer Anfrage.

  Es hängt vom Schema ab, welche Teile tatsächlich verwendet werden.

- Ein Netzwerk-Pfad beginnt mit "//", gefolgt von einer "Authority" und optional gefolgt von einem absoluten Pfad.
- Ein absoluter Pfad beginnt mit "/" und kann aus mehreren Segmenten bestehen, getrennt durch "/".
- Die Anfrage ist mit "?" markiert.

Syntax (4)

Hierarchical Scheme-Specific Part:



Syntax (5)

- Pfad-Segmente entsprechen Verzeichnis- / Dateinamen.

  Sie können aus Klein- und Großbuchstaben bestehen, sowie Ziffern, und diesen Sonderzeichen: ":", "@", "&", "=", "+", "$", ",", "-", "_", ".", "!", "~", "*", "'", "(", ")".

  Optional kann jedes Pfadsegment von ein oder mehr Parametern gefolgt sein, jeweils durch ";" abgetrennt.

- Obwohl ein Pfad wie ein Dateipfad aussieht, ist es möglich, daß
  - ◇ die Resource nicht einer Datei entspricht (sondern von einem Programm berechnet wird), oder
  - ◇ an einer ganz anderen Stelle im Dateibaum steht.

Syntax (6)

- Der "authority" Teil beschreibt, wer den Namensraum administriert, aus dem der Pfad stammt:
  - ◇ Für die meisten heutigen Schemata ist das der Rechner, auf dem die Server Software läuft (plus Port, ggf. Login): "server-based authorities".

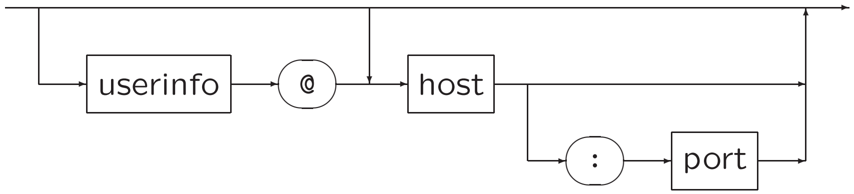
    Wer immer den Server betreibt, bestimmt die Pfade darauf.

  - ◇ Ein URI Schema kann auch "registry-based authorities" verwenden. Dann kann ist die genaue Syntax und Bedeutung abhängig vom Schema.

    Die Menge der zulässigen Zeichen ist natürlich eingeschränkt, damit die anderen Teile der URI gefunden werden können.

Syntax (7)

Authority (Server-Based):



- Host: Domain-Name, z.B. “`www.acm.org`”, oder eine IP-Nummer, z.B. “134.176.28.60”.

  Man kann auch “`www.acm.org.`” schreiben (mit einem Punkt am Ende). Im Domain-Anteil einer URI ist die Groß-/Kleinschreibung egal.

- Port: Zahl kleiner als 65536, z.B. “8080”.

Syntax (8)

- Der “userinfo”-Anteil kann ein Benutzername sein, oder die Form “Benutzername:Passwort” haben.

  Er kann aus Groß- und Kleinbuchstaben bestehen, sowie Ziffern, und folgenden Sonderzeichen: “;”, “:”, “&”, “=”, “+”, “$”, “,”, “–”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.

- URIs sind im Klartext in den Dokumenten enthalten, daher kann das Passwort jeder sehen, der Zugriff auf das Dokument hat.

Syntax (9)

- Manchmal identifiziert der Pfad ein Programm, das die verlangte Resource berechnet.
- Dann kann eine Anfrage (query) angefügt werden (als Teil der URI).
- Sie enthält die Argumente für das Programm.

  Statt Argumente sagt man auch Parameterwerte oder Eingabewerte.

  Eine Anfrage kann jeder String aus folgenden Zeichen sein: Groß- und Kleinbuchstaben, Ziffern und die Zeichen “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”, “,”, “–”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.

  RFC 2396 besagt, daß die folgenden Zeichen in einer Anfrage reserviert sind, aber er definiert nicht ihre spezielle Bedeutung: “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”

Anfragen (1)

- Für Anfragen sind zwei Formate üblich:
  - ◇ Das Programm hat ein oder mehrere Argumente, die durch “+” getrennt sind.
  - ◇ Die Eingabe des Programms ist eine Menge von Attribut-Wert-Paaren, getrennt durch “&”, z.B.

    `Vorname=Stefan&Nachname=Brass`

- Man beachte, daß in XML/SGML/HTML das Zeichen “&” eine spezielle Bedeutung hat, man muß es daher “`&`” schreiben (siehe Kap. 16).

Anfragen (2)

- Der übliche Escape-Mechanismus funktioniert auch hier, z.B. schreibt man ein richtiges "+" als "`%2B`".
- In beiden Formen wird das Pluszeichen intern auf ein Leerzeichen abgebildet.

  In der ersten Form trennt das Leerzeichen die Argumente.
- Z.B. ist auch dies möglich:

  `Titel=Grundlagen+des+World+Wide+Web`
- Z.B. wird der Inhalt eines Web-Formulars in Form von Attribut-Wert-Paare an den Server geschickt.

  Jedes Eingabefeld entspricht einem Attribut.

Inhalt

HTTP URIs (1)

- Das Schema "`http`" verwendet "server-based authorities".
- Der Teil "userinfo" wird dabei nicht genutzt.
- HTTP URIs beginnen also mit "`http://`", gefolgt von Domain-Namen oder IP-Nummer des Servers.
- Pfad und Anfrage sind optional.
- Eine HTTP-URI mit allen Teilen ist z.B.

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

HTTP URIs (2)

- Nur der Server-Anteil ("host") ist notwendig, z.B.:

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

- Der Rechner kann auch über eine IP-Nummer angegeben werden:

  `http://134.176.28.60/`
- Ist kein Port angegeben, so wird Port 80 benutzt.
- Abhängig von der Konfiguration des Web-Servers kann es einen Unterschied machen, ob ein "/" am Ende des Pfads steht oder nicht.

FTP URIs (1)

- Das Schema "`ftp`" nutzt "server-based authorities".
- Die Teile "userinfo" und "path" sind optional.
- Der Anfrage-Teil wird in FTP URIs nicht genutzt.
- Eine FTP-URI mit allen Teilen ist z.B.

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

- Meist enthält die URI nicht Benutzer und Passwort. Dann versucht der Browser, sich als "`anonymous`" mit der EMail-Adresse des Nutzers als Passwort anzumelden.

FTP URIs (2)

- Eine typische FTP URI ist:

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

- FTP hat verschiedene Transfer-Modi z.B. für
  - ◇ ASCII-Daten (Texte) und
  - ◇ binäre Daten (Programme, Bilder, etc.).

  Für ASCII-Daten wandelt der Server die Zeilenenden in CR LF um, und der Client wandelt sie in die lokale Konvention um, z.B. nur LF auf UNIX Systemen. Eine Binärdatei würde dabei zerstört werden. Normalerweise rät der Browser den Transfer-Modus aus der Endung der übertragenen Datei. Man kann aber einen Typ-Parameter zum Pfad hinzufügen, z.B. "`ftp://ftp.isi.edu/in-notes/rfc959.txt;type=I`". Der Typ "`I`" (image) bedeutet binär, "`AN`" bedeutet ASCII ohne spezielle Behandlung von Steuerzeichen wie FF.

File URIs (1)

- Mit File URIs kann man auf Dateien auf dem lokalen Rechner zugreifen. Dazu muß kein Webserver auf diesem Rechner laufen.

  Selbst wenn man auf seinem lokalen Rechner einen Webserver laufen hat, erlaubt er den Zugriff normalerweise nur auf einen kleinen Teil der Dateien (die auch von der allgemeinen Öffentlichkeit gelesen werden können). Andere Dateien des lokalen Systems, auf die der Benutzer Zugriff hat, können in den Browser mit File URIs geladen werden.

- Z.B. kann ich HTML Dateien auf meiner Platte mit URIs der folgenden Art lesen:

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

File URIs (2)

- Z.B. unter UNIX:

  `file:///home/brass/xml07/ex/ex1.xml`

- File URIs erlauben eine Rechner-Angabe ("host"), aber sie ist normalerweise leer oder "`localhost`".

  Wenn man einen bestimmten Rechner angibt, und dem Link wird auf einem anderen Rechner gefolgt, sollte der Benutzer eigentlich eine Fehlermeldung bekommen. Vielleicht können bestimmte Betriebssysteme auch von der "host"-Komponente Gebrauch machen. Aber es wird sehr selten benutzt. Offenbar haben alte Browser (Mosaic) File URIs mit Rechnerangabe als FTP-URIs verstanden.

- Z.B. funktioniert Folgendes:

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

Mailto URIs (1)

- Mailto URIs enthalten EMail Adressen, z.B.

  `mailto:Stefan.Brass@informatik.uni-giessen.de`

- Wenn man im Browser auf einen Link klickt, der eine Mailto URI enthält, öffnet der Browser ein EMail-Programm und trägt die gegebene Adresse als Empfänger ein.

  Sofern der Browser dieses URI Schema unterstützt. Die meisten modernen Browser tun das, aber etwa der originale Mosaic Browser nicht.

- Mailto URIs sind ein Beispiel für opaque URIs.

  Relative URIs machen keinen Sinn für das Mailto Schema.

Mailto URIs (2)

- Der erste Vorschlag für `mailto` URIs (RFC 1738) erlaubte nur eine EMail-Adresse in der URI.
- Dies wurde durch RFC 2368 erweitert, der beliebige "EMail Header" erlaubt:

  `mailto:brass@acm.org?subject=URI-Test`

- Auch mehrere Header und ein Rumpf sind möglich:

  `mailto:a@b.de?subject=s&cc=c@d.com&body=hello`

- Leerzeichen in den Daten müssen als "%20" geschrieben werden.

Mailto URIs (3)

- Es ist relativ leicht möglich, eine große Anzahl von Webseiten mit einem Programm ("Web-Robotor") zu besuchen, und EMail-Adressen zu extrahieren.
- Man bekommt dann viele Werbe-EMails ("SPAM").
- Es ist daher zu empfehlen,
  - ◇ auf Webseiten und insbesondere in `mailto`-URIs nur kurzzeitig gültige Adressen zu verwenden,
  - ◇ oder die EMail-Adresse so zu verschleiern, daß zwar ein Mensch sie erkennt, aber ein entsprechendes Programm zu aufwendig wäre.

Telnet URIs

- Wenn man ein Link mit einer Telnet URI aktiviert, wird normalerweise ein Telnet Fenster geöffnet, das ein Remote Login auf einem Rechner erlaubt, ggf. nur mit einem speziellen Programm.

  `telnet:www.informatik.uni-giessen.de:80`

- Es ist möglich, Benutzername und Passwort in der URI anzugeben, aber Passworte im Klartext zu speichern, auch nur lokal, ist immer eine schlechte Idee:

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

Inhalt

Fragment Identifier (1)

- Eine URI identifiziert ein Dokument, also typischerweise den Inhalt des Browserfensters.
- Je nach Medientyp des Dokumentes ist es eventuell möglich, zu verlangen, daß der Browser automatisch an eine bestimmte Position scrollt.

  Der Browser springt also zu einem bestimmten Abschnitt des Dokumentes, anstatt immer den Anfang zu zeigen. Der Browser läd aber das komplette Dokument, und man kann zum Anfang zurück-scrollen.
- Auf diese Art kann man nicht nur auf vollständige Dokumente verweisen, sondern auch auf Teile von Dokumenten.

Fragment Identifier (2)

- Bei HTML Dokumenten ist der Verweis auf eine bestimmte Position grundsätzlich möglich, aber nur auf Stellen, die vom Autor dafür vorgesehen sind.
- Der “Fragment Identifier”, der eine Position markiert, ist formal nicht Teil der URI.
- Die URI Spezifikation definiert aber “URI Reference”, die aus einer URI und einem Fragment Identifier besteht, getrennt durch “#”:

$$\underbrace{\texttt{http://www.x.org:80/a/b?c}}_{\text{URI}}\ \texttt{\#}\ \underbrace{\texttt{Kapitel2}}_{\text{Fragment ID}}$$

Fragment Identifier (3)

- Fragment Identifier sind syntaktisch wenig eingeschränkt.

  Fragment Identifier können jede Folge von Groß-/Kleinbuchstaben, Ziffern, und folgenden Sonderzeichen sein: “;”, “/”, “?”, “:”, “@”, “&”, “=”, “+”, “$”, “,”, “-”, “_”, “.”, “!”, “~”, “*”, “’”, “(”, “)”.
- Die Möglichkeit, Fragment Identifier zu benutzen, hängt vom Medientyp ab, nicht vom URI Schema.

  Wenn auf HTML Dateien über FTP zugegriffen wird, können Sie genauso verwendet werden.
- Für XML-Dokumente sind mit XPath Verweise auf beliebige Positionen im Dokument möglich, sogar ohne Vorbereitung durch den Autor.

Relative URIs (1)

- Oft bestehen große Dokumente aus mehreren einzelnen Webseiten, die auf einander verweisen.
- Viele Webseiten enthalten auch Bilder. Jedes Bild hat seine eigene URI. Die Hauptdatei (in HTML) verweist auf die Bilddateien.
- Normalerweise werden die Webseiten für ein Dokument und die zugehörigen Bilder auf dem gleichen Server gespeichert, oft im gleichen Verzeichnis.
- Man kann daher auf Resourcen relativ zur URI der aktuellen Resource verweisen.

Relative URIs (2)

- Beispiel: Die URI des aktuellen Dokumentes sei:

  `http://www.sis.pitt.edu/~sbrass/dd00/index.html`
- Die relative URI “`h1.pdf`” bezieht sich dann auf:

  `http://www.sis.pitt.edu/~sbrass/dd00/h1.pdf`
- Die relative URI “`../db99/c3.ps`” bezieht sich auf:

  `http://www.sis.pitt.edu/~sbrass/db99/c3.ps`
- Auch ein absoluter Pfad kann eine relative URI sein. Z.B. bezieht sich “`/academics/courses/`” auf:

  `http://www.sis.pitt.edu/academics/courses/`

Relative URIs (3)

- Die leere URI bezieht sich auf das Dokument selbst.

  Dies ist vermutlich nur nützlich, wenn ein Fragment Identifier angegeben ist. Z.B. bezieht sich “`#KAP1`” im Beispiel auf die URI Reference “`http://www.sis.pitt.edu/~sbrass/dd00/index.html#KAP1`”.
- Anfrage und Fragment Identifier werden nie von der URI des aktuellen Dokumentes (Basis-URI) übernommen, sie müssen ggf. in der relativen URI explizit neu angegeben werden.
- Der Algorithmus zur Übersetzung einer relativen URI in eine absolute URI (bei gegebener Basis-URI) steht in RFC 2396, Abschnitt 5.2.

Relative URIs (4)

- Relative URIs sind aus folgenden Gründen nützlich:
  - ◇ Sie sparen dem Dokument-Autor etwas Tippaufwand.
  - ◇ Man kann das Gesamtdokument, das aus vielen einzelnen Dateien besteht, leicht in ein anderes Verzeichnis oder auf einen anderen Rechner verschieben, ohne alle enthaltenen URIs anpassen zu müssen.

    Allerdings ändert sich natürlich die URI des Gesamtdokumentes, was man besser vermeiden sollte (Links von außen funktionieren dann nicht mehr).

Inhalt

URIs, URLs, URNs (1)

- In den formalen Spezifikationen wird der Begriff "Uniform Resource Identifier" (URI) verwendet.
- In der Praxis wird aber meist von "Uniform Resource Locators" (URLs) geredet.
- Dies ist formal nur eine Teilmenge der URIs, aber bisher gibt es im wesentlichen nichts anders.
- Eine URL beschreibt die Netzwerk-Adresse, unter der eine Resource zur Verfügung steht.

URIs, URLs, URNs (2)

- Probleme von URLs:
  - ◇ Manche Resourcen sind von mehreren Servern verfügbar ("gespiegelt"), man soll den nächsten benutzen. URLs enthalten aber einen Server.
  - ◇ Manchmal werden Rechner oder Domains aufgegeben, oder eine Resource ist auf einem Webserver nicht mehr erwünscht. Ggf. wird die Resource dann aber auf anderen Servern angeboten.
  - ◇ Es gibt auch Resourcen, die noch nicht elektronisch im Web verfügbar sind, auf die man sich aber auch beziehen muß (z.B. Bücher).

URIs, URLs, URNs (3)

- Die obigen Beispiele zeigen, daß Resourcen eigentlich auf eine Art identifiziert werden sollten, die unabhängig vom konkreten Speicherort ist.
- Uniform Resource Names (URNs) sind in Entwicklung, um diese Probleme zu lösen.
- Sie werden persistent und orts-unabhängig sein.
- Obwohl darüber schon lange diskutiert wird, haben sie sich noch nicht durchgesetzt.

URIs, URLs, URNs (4)

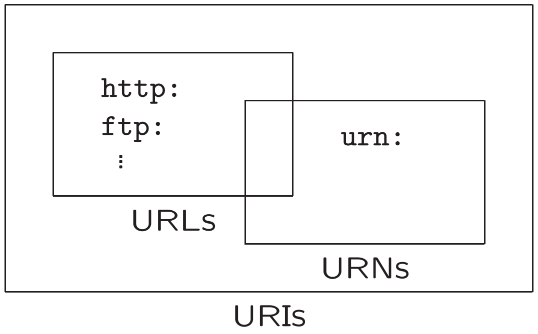


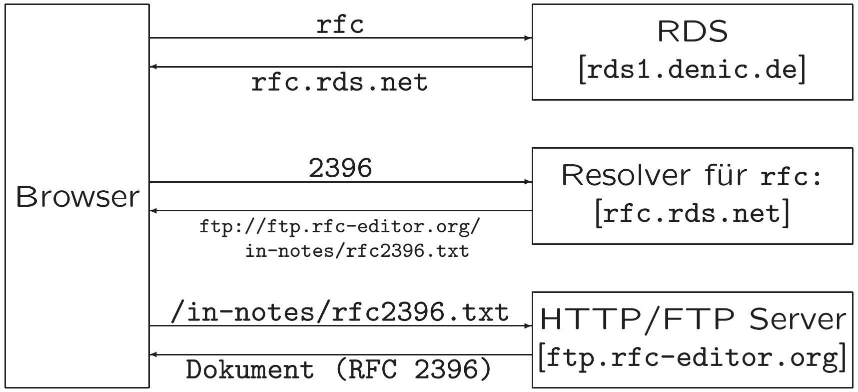
Bild in Anlehnung an [http://www.w3.org/Addressing/]

URIs, URLs, URNs (5)

- URNs benutzen das Schema “urn:”. Dies wird gefolgt von einem Namespace Identifier N, dann einem “:”, und dann einer namespace-spezifischen Zeichenfolge S. Der Aufbau ist also “urn:N:S”.
- Um eine Resource zu bekommen, muß man
  - ◇ erst einen “Resolver Discovery Service” (RDS) befragen, um einen “Resolver” für N zu finden,
  - ◇ dann den Resolver für N mit der Zeichenfolge S befragen, um eine URL U zu erhalten, und
  - ◇ schließlich die Resource mit der URL U holen.

URIs, URLs, URNs (6)

Hypothetisches Beispiel: [urn:rfc:2396]



Persistent URLs (1)

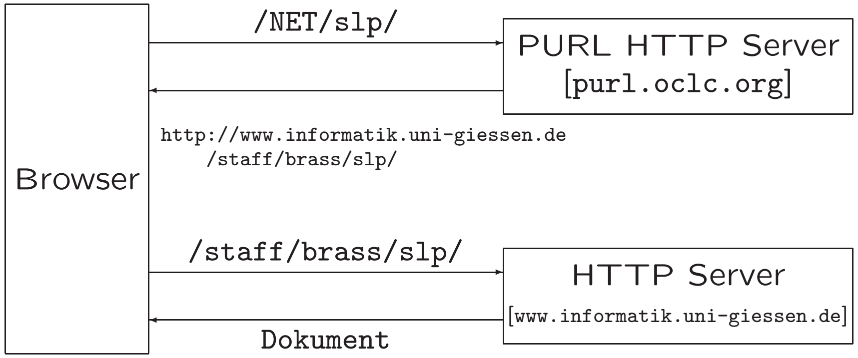
- Das OCLC (Online Computer Library Center) hat PURLs (Persistent URLs) als einen Zwischenschritt vorgeschlagen (bis URNs fertig/etabliert sind).
- Auch hier ist die Idee eine zusätzliche Indirektion:
  - ◇ PURLs sind URLs für den Server vom OCLC, z.B.

    http://purl.oclc.org/NET/slp/
  - ◇ Dieser Server liefert automatische Weiterleitung in HTTP (siehe Kapitel 15), im Beispiel auf:

    http://www.informatik.uni-giessen.de/staff/brass/slp/
  - ◇ Der Browser holt dann automatisch diese Seite.

Persistent URLs (2)

Beispiel:



Persistent URLs (3)

- Wenn dieses Dokument mit mir an die Universität Halle wechselt, kann ich die URL in der Datenbank des OCLC aktualisieren, so daß sie verweist auf:

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

- Wenn die URNs funktionieren, wird es ein URN Schema "`purl`" geben. Voraussichtliche URN:

  `urn:purl:/NET/slp/`

- Im April 2003 gab es 568 642 PURLs.

  Statt seine eigene Domain zu reservieren (oder zusätzlich dazu), sollte man eine PURL reservieren.

Entwurf stabiler URIs (1)

- Das WWW ist voll von Referenzen auf URIs, die nicht mehr existieren ("broken/dangling links").
- Oft ist das Dokument selbst gar nicht gelöscht worden, sondern nur an einen andern Platz verschoben.
- Dies ist ein Problem für
  - ◇ Benutzer, die auf einen Link klicken, und nach einiger Wartezeit eine Fehlermeldung bekommen.
  - ◇ Autoren, die ihre Webseiten ständig aktualisieren müssen, nur weil die darin enthaltenen URIs sich geändert haben.

Entwurf stabiler URIs (2)

- Während man auf seinem PC Dateien beliebig verschieben kann, sind URIs automatisch öffentlich.

  Sobald sie in einem öffentlich bekannten Dokument auftauchen.

- Man sollte URIs sorgfältig auswählen, so daß sie noch in 100 Jahren benutzt werden können.

  URIs können auch in Büchern gedruckt werden, und unsere Bibliothek enthält einige sehr alten Bücher. Wenn die Firma oder Universität, die den Webserver betreibt, nicht geschlossen wird, sollte der Fortschritt in der Speichertechnologie es möglich machen, die alten Webseiten weiter verfügbar zu halten.

Entwurf stabiler URIs (3)

- In der Konfiguration des Webservers kann man eine beliebige Abbildung von URIs auf Plattendateien definieren.

  Wenn es also vielleicht auch gute Gründe gibt, die Verzeichnisstruktur auf der Platte zu ändern, heißt das nicht unbedingt, daß die alten URIs nicht weiter unterstützt werden können.

- Man sollte unterschieden zwischen URIs für
  - ◇ die aktuelle Version eines Dokumentes, oder was wir jetzt unter einem Begriff verstehen, und
  - ◇ Archiv-URIs, die ein spezifisches Dokument identifizieren (ggf. nur von historischem Interesse).

Entwurf stabiler URIs (4)

- "Designing [URIs] mostly means leaving information out." [Tim Berners-Lee: Cool URI's don't change.]
- Folgendes sollte nicht in URIs enthalten sein:
  - ◇ Der Name der Person, die die Seite verwaltet.

    Es sei denn, das Dokument gehört immer nur zu dieser Person.
  - ◇ Das Programm / die Technologie, die benutzt wird, um die Seite zu berechnen (z.B. `/cgi-bin/`).
  - ◇ Dateiendung: Vielleicht wird HTML in 20 Jahren nicht mehr verwendet (alles in PDF konvertiert).
  - ◇ Status, Zugriffsrechte: Kann sich ändern.