URIs
and
RFC 3986

Noah Mendelsohn
Tufts University
Email: noah@cs.tufts.edu
Web: http://www.cs.tufts.edu/~noah
Goals

- What is *named* when we use the Web
- Learn the detailed design of URIs
- See how the naming principles we’ve explored are reflected in Web architecture and URIs
- Learn to read RFCs and to study the art of writing specifications
- Understand why grammars are important
Review: Naming Questions
Some characteristics of names

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too few names)
- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Review
Web Architecture Basics
Architecting a universal Web

- Identification: URIs
- Interaction: HTTP
- Data formats: HTML, JPEG, etc.
What Happens When We Browse a Web Page?

Consider:
What are all the things that are “named” in the interaction between browser and Web Server?
The user clicks on a link

**URI is** [http://webarch.noahdemo.com/demo1/test.html](http://webarch.noahdemo.com/demo1/test.html)
The http “scheme” tells client to send HTTP GET msg

URI is http://webarch.noahdemo.com/demo1/test.html
The client sends an HTTP GET

URI is http://webarch.noahdemo.com/demo1/test.html

GET /demo1/test.html HTTP/1.0
Host: webarch.noahdemo.com
User-Agent: Noah's Demo HttpClient v1.0
Accept: */*
Accept-language: en-us
The server sends an HTTP Response

HTTP/1.1 200 OK
Date: Tue, 28 Aug 2007 01:49:33 GMT
Server: Apache
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>

HTTP Status Code 200 Means Success!
The server sends an HTTP Response

HTTP/1.1 200 OK
Date: Tue, 28 Aug 2007 01:49:33 GMT
Server: Apache
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
Architecting a universal Web

- Identification: URIs
- Interaction: HTTP
- Data formats: HTML, JPEG, etc.
Assign URIs for all Resources

- A resource is something that has information (e.g. a Web page)

- *If a resource doesn’t have a URI, you can’t link to it...it’s not part of the Web.*
The Structure of URIs
A simple URI

http://uss.tufts.edu/stuserv/acadcal/
A simple URI

http://uss.tufts.edu/stuserv/acadcal/
A simple URI

http://uss.tufts.edu/stuserv/acadcal/
Schemes

http://uss.tufts.edu/stuserv/acadcal/

mailto:noah@cs.tufts.edu

Schemes let us name different kinds of things, accessed in different ways.
A simple URI

http://uss.tufts.edu/stuserv/acadcal/

Authority: who controls allocation of this name?
A simple URI

// Fixed in grammar to indicate authority follows

http://uss.tufts.edu/stuserv/acadcal/
A simple URI

http://uss.tufts.edu/stuserv/acadcal/

Path: provides for hierarchical naming...
... also supports “../xxx” relative syntax
A simple URI

http://uss.tufts.edu/stuserv/acadcal/

Path: provides for hierarchical naming...
... maps well to hierarchical information systems
A more complex URI

http://www.tufts.edu?student=smith
A more complex URI

http://www.tufts.edu?student=smith

The query *is* part of the URI... However, in many cases, all URIs with a common path are processed by the same server-side code.

Also...HTML forms are useful for filling in the query components.
Fragments identify parts of documents


Fragment interpretation depends on the media type of the returned representation (text/html)...this is useful but tricky and causes a variety of problems.
Characteristics of URIs
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)
- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs

- **Absolute vs. relative**
- **Address (locator)?**
- **Human readable?**
- **Short/convenient?**
- **Global (context independent)?**
- **Ensures referent exists?**
- **Aliases? (too many names)**

- **Opaque vs. data-carrying?**
- **Reflect structure of system?**
  - Supports navigation: e.g. “..”?  
- **Who can generate them?**
- **Constraints from environment**
  - E.g. no “-” in C/C++ variable names
- **Indirect identification allowed?**
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. "..")?
- Who can generate them?
- Constraints from environment
  - E.g. no "-" in C/C++ variable names
- Indirect identification allowed?

Depends on scheme
Some characteristics of URIs:

- Absolute vs. relative?
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

Yes...

URIs “on the side of a bus” is an important goal... but some URIs are complex

- Supports navigation: e.g. “..”?

- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs:

- Absolute vs. relative
- Address (locator)?
- Human readable?
- **Short/convenient?**
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

Allowed but not required:

- Reflect structure of system?
  - Supports navigation: e.g. “..”?  
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs:

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

With most schemes, absolute URIs are global

- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs:

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

FILE: scheme is not global!
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)
- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?

NO!!
Status code 404 is key to Web scalability
Some characteristics of URIs:

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

Some aliases required e.g.: http vs. HTTP...

Worst cases depend on users

Some aliases required e.g.: http vs. HTTP...

Worst cases depend on users

Reflect structure of system?
  - Supports navigation: e.g. “..”?

Who can generate them?

Constraints from environment
  - E.g. no “-” in C/C++ variable names

Indirect identification allowed?
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Who can generate them?
- Reflect structure of system?
- Opaque vs. data-carrying?

Indirect identification allowed?
- E.g. no "-" in C/C++ variable names
- Constraints from environment

Constraints from environment
- Supports navigation: e.g. ".."

Who can generate them?
- Supports navigation: e.g. ".."
- Supports navigation: e.g. "..

Reflect structure of system?
- Opaque vs. data-carrying?

Indirect identification allowed?
- E.g. no "-" in C/C++ variable names
- Constraints from environment

Who can generate them?
- Supports navigation: e.g. ".."
- Supports navigation: e.g. "..

Reflect structure of system?
- Opaque vs. data-carrying?

Indirect identification allowed?
- E.g. no "-" in C/C++ variable names
- Constraints from environment

Who can generate them?
- Supports navigation: e.g. ".."
- Supports navigation: e.g. "..

Reflect structure of system?
- Opaque vs. data-carrying?

Indirect identification allowed?
- E.g. no "-" in C/C++ variable names
- Constraints from environment

Who can generate them?
- Supports navigation: e.g. ".."
- Supports navigation: e.g. "..

Reflect structure of system?
- Opaque vs. data-carrying?

Indirect identification allowed?
- E.g. no "-" in C/C++ variable names
- Constraints from environment

Who can generate them?
- Supports navigation: e.g. ".."
- Supports navigation: e.g. "..

Reflect structure of system?
- Opaque vs. data-carrying?
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

URIs are *the* structuring mechanism for the Web as a whole.

- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

Designed to allow mappings to hierarchical systems

- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. ".."?
- Who can generate them?
- Constraints from environment
  - E.g. no "-" in C/C++ variable names
- Indirect identification allowed?
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

- Opaque vs. data-carrying?
- Reflect structure of system?
- Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?

Decentralized allocation except:
Some characteristics of URIs

- Absolute vs. relative?
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)
- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?

scheme names centrally registered with IANA
Some characteristics of URIs
• Absolute vs. relative
• Address (locator)?
• Human readable?
• Short/convenient?
• Global (context independent)?
• Ensures referent exists?
• Aliases? (too many names)

For http and mailto schemes: central Domain Name (DNS) registration required for authority

• Opaque vs. data-carrying?
• Reflect structure of system?
  Supports navigation: e.g. “..”?
• Who can generate them?
• Constraints from environment
  – E.g. no “-” in C/C++ variable names
• Indirect identification allowed?
Some characteristics of URIs:

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)
- Opaque vs. data-carrying?
- Reflect structure of system?
  - Supports navigation: e.g. “..”?
- Who can generate them?
- Constraints from environment
  - E.g. no “-” in C/C++ variable names
- Indirect identification allowed?

Yes. E.g.: ASCII-only, spaces and some punctuation must be %encoded.
Some characteristics of URIs

- Absolute vs. relative
- Address (locator)?
- Human readable?
- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

URIs are silent on this…but HTTP redirection provides for indirect identification

- Short/convenient?
- Global (context independent)?
- Ensures referent exists?
- Aliases? (too many names)

Constraints from environment
- E.g. no “-” in C/C++ variable names

Indirect identification allowed?
Grammars
What are formal grammars?

- Grammars are formal languages for specifying other languages
- A grammar allows you to:
  - Always: determine whether a given string is “in” the specified language
  - Often: associate structures in the grammar with parts of the string
- The Chomsky hierarchy:
  - Different grammars have different expressive power
  - Regular expressions recognize “regular languages” \((ab^*)\) \(\rightarrow\) a, ab, abb, abbb
  - Context-free grammars are more powerful: typically used for programming languages
  - The ABNF used in RFC’s is a context-free grammar
  - Context-free grammars can be recognized (parsed) by a finite-state pushdown automaton

Why use formal grammars for specifying languages?

- Precise and rigorous
  - Less ambiguous than an explanation in English
- Membership of a string in a language can be checked automatically
- Tools to process the language can often be constructed automatically from the grammar
ABNF: the grammar for IETF RFCs

- ABNF example from RFC 3986:
  
  \[ URI = \text{scheme "::" hier-part} \ [ \ "?" \ \text{query} ] \ [ \ "#" \ \text{fragment} ] \]

- ABNF is itself specified in RFC 2234

- ABNF is convenient to use in fixed-font specification documents like RFCs
Summary
Summary

- The structure and interpretation of URIs is set out in RFC 3986
- URIs embody many of the principles we have studied
- Formal grammars are powerful tools for specifying names
- *The design decisions embodied in URIs are keys to the success of the Web!!*