Fragment Identifiers for Plain Text Files

Erik Wilde and Marcel Baschnagel
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH Zürich)
erik.wilde@ethz.ch

ABSTRACT

Hypermedia systems like the Web heavily depend on their ability to link resources. One of the key features of the Web's URIs is their ability to not only specify a resource, but to also identify a subresource within that resource, by using a fragment identifier. Fragment identification enables user to create better hypermedia. We present a proposal for fragment identifiers for plain text files, which makes it possible to identify character or line ranges, or subresources identified by regular expressions. Using these fragment identifiers, it is possible to create more specific hyperlinks, by not only linking to a complete plain text resource, but only the relevant part of it. Along with this proposal, a prototype implementation is described which can be used both as a server-side testbed and as a client-side extension for the Firefox browser.

Categories and Subject Descriptors: I.7.2 [**Document and Text Processing**]: Document Preparation — *Hypertext/Hypermedia*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia — *Navigation*

General Terms: Design, Standardization

Keywords: Plain Text, URI, XLink, Firefox

1. Introduction

The Web heavily depends on its ability to link resources. Even though the vast majority of links on the Web is using references to complete documents, the ability to reference document fragments greatly enhances the power of a hypermedia system, and its usefulness for the users. The generic URI syntax [1] specifies fragment identifiers, and thus enables Web links to point to document fragments. On the Web, fragment identification depends on a resource's MIME type, and the most popular fragment identification method is that of HTML. URIs with fragment identifiers such as `http://www.w3.org/TR/html/#C_8` reference a particular subresource within the resource, in case of HTML fragment identifiers the element with the `id` attribute set to `C_8` (for example the anchor element `<a id="C_8">...</a>`).

Fragment identifiers have been defined for other widely used resource types, such as XML (where fragment identifiers are XPointers [4]) and PDF (where it is possible to identify pages and other subresources). However, since tool support for working with URIs with fragment identifiers is weak, and fragment identifiers often are less robust then references to the resource only, only a small fraction of links on the Web specify fragment identifiers.

As a proof of concept, a prototypical implementation for plain text fragment identifiers has been developed, and therefore it is possible to test and use these fragment identifiers in a server-based scenario as well as in the form of a Firefox extension.

2. Plain Text Files and Hypertext

In terms of hypertext, plain text files have severe limitations, because they cannot contain outgoing links, and they cannot serve as targets for links to subresources. Interestingly, defining fragment identifiers solves both problems, because outgoing links can then be defined by using external links (for example XLinks [3][1]), and incoming links can point to subresources of the plain text file.

The majority of files used on the Web are media types other than plain text, but there still is a large number of plain text files available. These are either legacy files, which were never converted to a richer media type (a non-trivial task, if structure needs to be added), or they are output files from tools or systems which have plain text output, a popular example is the *Comma-Separated Values (CSV)* format (a weakly structured format, but in terms of hypertext usage simply treated as unstructured plain text). For all these plain text resources, fragment identification makes it possible to integrate them better into the hypermedia world of the Web, even if only with incoming links.

In a Web with plain text fragment identification, it would for example be possible to have a link pointing to a particular line in a log file and thus be able to make a statement about this subresource, such as the comment that this log entry requires further investigation.

3. Fragment Identifiers for Plain Text Files

Fragment identifiers are strings appended to a URI, using the `#` character as a separation between the resource identification and the fragment identifier. The generic URI

[1] As pointed out by BRY and ECKSTEIN [2], however, even after 4 years the XLink specification has not had any success as a Web technology.

syntax [1] states that *"the fragment identifier component of a URI allows indirect identification of a secondary resource by reference to a primary resource and additional identifying information. [...] The fragment's format and resolution is [...] dependent on the media type of a potentially retrieved representation, even though such a retrieval is only performed if the URI is dereferenced."*

In an Internet Draft [6] currently under development (and open for discussion on the relevant mailing lists), a fragment identifier syntax and semantics for plain text resources has been defined. The basic building blocks of these fragment identifiers are the following:

- *Positions and Ranges:* Positions identify fragments of length zero, which may be a useful concept if a link expresses the concept of "insert some text here". Ranges, on the other hand, are fragments that span between two positions, and thus can be thought of what users normally can select with a mouse.
- *Characters and Lines:* The only plain text structures that can be identified reasonably across different character encodings are individual characters and lines. Both concept are important when dealing with plain text files, and are usually supported by text tools (for example providing a "go to line" function).
- *Regular Expressions:* While characters and lines easily break when the file is modified, it may be more robust to identify a subresource through string matching. By using a regular expression, a fragment identifier selects all character ranges matching this regular expression.

The concepts of positions/ranges and characters/lines may be combined to use line ranges and similar concepts. Also, different identification methods can be combined, and the fragment identified by such a combination is the union of all individually identified fragments. As with regular expressions, this may easily lead to disjoint subresources, which is not a problem and can be easily represented by using highlighting or similar concepts.

Finally, a hash value can be used to be able to detect changes in the resource. This may be useful to be able to detect potentially broken fragment identifiers. As long as the hash value (with may be the length or the MD5 checksum) has not changed, the resource probably has not been changed and the fragment identifier can safely be applied.

The following examples show how the concepts are used in actual fragment identifiers. They are using some of the fragment identification schemes defined by the fragment identifier syntax. Both examples assume that the resource retrieved by dereferencing the URI is a plain text file.

```
1. http://example.com/text.txt#line=10,20
2. http://example.com/text.txt#match=[rR][fF][cC]
```

In the first example, the URI identifies a fragment consisting of lines 11–20 of the identified resource. If the resource does not have that many lines, the fragment is smaller or may even be empty. The URI of the second example[2] identifies a fragment of the resource which consists of a case-insensitive match of the string "RFC", which is likely to be a disjoint subresource (if there is more than one occurrence of the case-insensitive string "RFC" within the resource).

[2]Please note that the URI syntax requires the brackets to be percent-encoded, which for better readability is not shown in the example.

4. Implementation

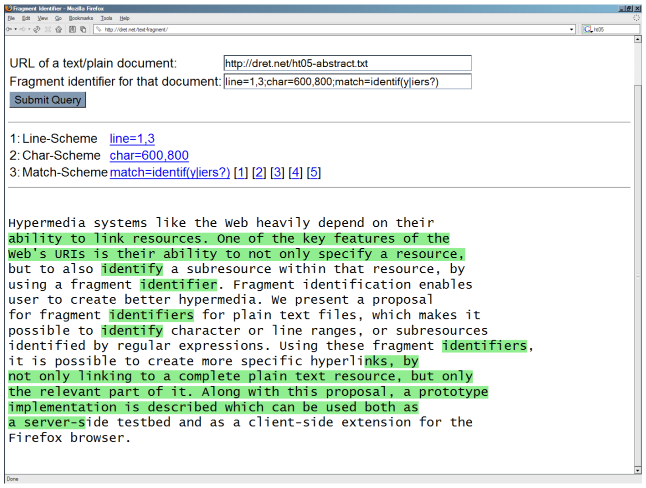
Implementations of the fragment identification scheme for plain text resources should be included in future Web browsers, and in other tools handling plain text files (such as text viewers and editors), so that these tools can generate fragment identifiers for inclusion into Web resources. As a prototype, we have implemented fragment identifier interpretation code that operates in three steps:

1. *Identifier Analysis:* The fragment identifier is analyzed and it is tested whether it conforms to the syntax definition.
2. *Resource Analysis:* The resource is retrieved, tested for its MIME type, tested for its character encoding, and tested against the fragment identifier's hash sum (if present).
3. *Identifier Application:* If the fragment identifier is correct and applicable, the subresource is located and presented to the user.

Since a prototype should be available for as many interested users as possible, it is available in two configurations, as server-side and client-side implementation.

4.1 Server-Side Implementation

The server-side implementation is based on Perl, which provides excellent string handling functionality and thus is ideally suited to support the interpretation of fragment identifiers. The server-side implementation is available for testing at `http://dret.net/text-fragment/` and can be used with any text file.


Fragment Identifier - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://dret.net/text-fragment/
ht05
Submit Query
Done

URL of a text/plain document: http://dret.net/ht05-abstract.txt
Fragment identifier for that document: line=1,3;char=600,800;match=identif(y|iers?)

1: Line-Scheme line=1,3
2: Char-Scheme char=600,800
3: Match-Scheme match=identif(y|iers?) [1] [2] [3] [4] [5]

```
Hypermedia systems like the Web heavily depend on their
ability to link resources. One of the key features of the
Web's URIs is their ability to not only specify a resource,
but to also identify a subresource within that resource, by
using a fragment identifier. Fragment identification enables
user to create better hypermedia. We present a proposal
for fragment identifiers for plain text files, which makes it
possible to identify character or line ranges, or subresources
identified by regular expressions. Using these fragment identifiers,
it is possible to create more specific hyperlinks, by
not only linking to a complete plain text resource, but only
the relevant part of it. Along with this proposal, a prototype
implementation is described which can be used both as
a server-side testbed and as a client-side extension for the
Firefox browser.
```

Figure 1: Demo Text with Fragment Identifier

A simple example is shown in Figure 1, which is the result of applying a three-part fragment identifier to a text file containing this paper's abstract. The `line=1,3` part selects the second and the third line, the `char=600,800` part selects the 601st through the 800th character, and the `match=identif(y|iers?)` part selects all strings which are "identify", "identifier", or "identifiers".

The figure shows the server-side implementation, which displays an input form for selecting a resource and a fragment identifier, an analysis of the fragment identifier, and finally the resource itself with the identified subresource highlighted. In the example, a disjoint subresource is identified, which is not a problem.

While fragment identification typically is a client-side functionality, the server-side implementation is an ideal platform for users to test fragment identifiers for plain text files, and to debug them by using the site's step-by-step output of fragment identifier processing.

4.2 Client-Side Implementation

Fragment identifier support is a typical client-side functionality. As such, it suffers from the same problem as many client-side technologies, namely slow adoption due to the fact that the majority of users only rarely install new browser software. To avoid this pitfall, the current client-side implementation uses the *extension* mechanism of the popular Firefox browser, which makes it possible to extend the browser with new code without having to install a new browser version. Installing extensions is done at run time, and extensions plug into the browser without the need to recompile or even restart the browser.

Another advantage of the extension concept is that new functionality can be nicely packaged into an extension, without a need to be tightly integrated with the browser's code. Naturally, extensions need to interact with the browser and do so through well-defined interfaces, but this interaction is limited and easier to handle than full integration into the browser code.

Due to the way extensions are handled, the fragment identifier processing extension works as follows:

1. *Request Analysis:* After completing a request, the extension is notified and inspects the request's result. Only if the result is of MIME type `text/plain` and a fragment identifier is present, the extension continues with the following steps.

2. *Identifier Processing:* At this point, the basic fragment identifier interpretation process as described above is invoked, analyzing the identifier and the resource, and evaluating the identifier's result (i.e., the subresource).

3. *Identifier Presentation:* Because Firefox renders `text/plain` files as HTML pages with a `pre` element containing the text file's content, the subresource can be easily presented by inserting `span` elements which highlight the subresource.

   Even though we have not done this, this would offer a convenient way of defining a unique representation of identified subresources across different media types (such as HTML and plain text), which could then be set via the browser's preferences.[3]

After these steps, the identified subresource can be displayed as a highlighted part of the plain text file, making it visually clear which fragment of the resource has been selected by the fragment identifier.

[3] Currently, however, Firefox does not properly highlight an identified HTML subresource when navigating to it, but instead the first link inside the identified subresource.

5. Generating Fragment Identifiers

The concept and the implementation of fragment identifiers has been presented in this paper, but the question remains whether these fragment identifiers will be used. Generating fragment identifiers by hand is feasible, due to the simple syntax, but still is more complicated than what most users are used to.

We envisage browser support for the creation of fragment identifiers (this could also cover HTML fragment identifiers), where users identify a fragment by selecting it with the mouse, and then click a button to generate the appropriate fragment identifier. This functionality could also be integrated into Firefox through the extension mechanism, thus making this kind of functionality available through a simple plug-in.

Another and possibly more interesting application area is the area of generated content. Any plain text content that is generated or managed by tools could be made better accessible by also generating toc/index documents (probably in HTML) which add a layer of links to the otherwise unstructured plain text files.

6. Related Work

Starting with Xanadu's *Tumblers* and NLS's addressing scheme (which already used the `#` character for fragment addressing), a lot of addressing schemes for resource fragment identification have been proposed. More recent examples are HyTime's *location addresses* and OHP's *LocSpec* [5]. However, we are not aware of any concrete proposals for fragment identification within plain text resources on the Web.

7. Conclusions and Future Plans

The main goal of this paper is to create awareness for fragment identifiers for plain text files, to use this discussion as input for the IETF process towards publication of an RFC, and to provide interested parties with prototype implementations (most importantly, the extension for the Firefox browser) for experimenting with fragment identifiers.

8. References

[1] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic Syntax. Internet proposed standard RFC 3986, January 2005.

[2] François Bry and Michael Eckert. Processing Link Structures and Linkbases in the Web's Open World Linking. In *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia*, Salzburg, Austria, August 2005. ACM Press.

[3] Steven J. DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[4] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. XPointer Framework. World Wide Web Consortium, Recommendation REC-xptr-framework-20030325, March 2003.

[5] Lloyd Rutledge, Lynda Hardman, and Jacco van Ossenbruggen. Applying the HyTime Model to the Open Hypermedia Protocol LocSpec. In *3rd Workshop on Open Hypermedia Systems*, Southampton, UK, April 1997.

[6] Erik Wilde. URI Fragment Identifiers for the text/plain Media Type. Internet Draft draft-wilde-text-fragment-04, June 2005.