Supporting Hierarchical Guided Tours in the World Wide Web

Franz J. Hauck

Dept. of Math. and Computer Science, Vrije Universiteit
Amsterdam, The Netherlands

Email: fjh@cs.vu.nl

Abstract: Guided tours are well known from hypertext systems as a means to collect hypertext documents in a sequence with a certain meaning. This can be followed by readers using *next* and *previous* links. As there is almost no support for guided tours in the World Wide Web so far, these structures have to be built up manually, which not only causes a lot of work on updates but is also almost impossible in the case of non-local documents. This paper presents a simple server-side concept for building guided tours using any existing Web pages. The pages used within a tour need not be aware of being used, and thus, need no special preparation at all. Pages may occur multiple times within several tours or even within one tour. We present a server-based implementation demonstrating the concept.

Keywords: Guided Tours, Authoring Environments, WWW Server

---

1 Introduction

The authoring of hypertext documents requires some thought about how to split information into different documents, usually called nodes, and how to interconnect the nodes using hyperlinks, which allow readers to navigate through the entire system. Usually, hypertext systems, and also new hypermedia design methods such as RMM [ISB95], allow authors to give hypertext links a special meaning. For example, a node representing a painting may have a link to the node representing its painter. In this case, the meaning of the links is ‘painted by’. In [GMP95] these links are called *schema links*, in contrast to *generic links*, because they result from a schematic design process of the presented information. It is up to the hypertext system in companion with the author to render schema links in such a way that the reader immediately understands their meaning.

One often used pattern is a guided tour [Trigg88]. This is a sequence of nodes. There are schema links which allow readers to follow the sequence back and forth. As nodes may be part of several guided tours these nodes may have several schema links for each tour. Since it is confusing to show readers all these links, only the one which is necessary to follow the current tour will be displayed.

In contrast, the World Wide Web provides only one type of hypertext link and it is completely up to the author to decorate these links in a consistent way, such that readers are aware of the meaning of the link. As the HTTP protocol [HTTP96] for the transportation of Web pages is stateless, Web pages are almost static and all their possible dynamic behaviour can hardly depend on the history of the reader’s actions. This does not allow the usage of a single Web page in multiple guided tours without showing all the schema links of other guided tours the page is involved in.

While modern hypertext systems support authors by building up schema links, there are almost no such means for Web pages. The organization of a guided tour requires the author to insert the schema links manually in each page. Imagine a department of a university which wants to serve the WWW documents of its staff members. Each staff member has his or her own Web page. These pages are linked together, so that a reader can follow hypertext links to the next or previous staff member in alphabethical order: a guided tour. Whenever a new member is employed, or another leaves the department, hypertext links in several documents have to be updated. Thus, manually maintaining the schema links has a serious drawback. As Web pages may be distributed over the entire world, this update may be impossible because it is often not possible to change non-local documents for inserting the schema links. This happens, for example, when an author wants to show his readers a sequence of selected search engines used in the WWW.

This paper elaborates a server-side mechanism to create the schema links automatically for lists, sequences, and even hierarchical guided tours, according to some meta-information stored in a separate configuration file. These structures can enumerate arbitrary WWW documents on arbitrary servers while the configuration remains on the local server, and thus with the administrator or creator of the structure.

The paper is organized as follows. Section 2 explains the structures which need to be supported and what this support looks like. Section 3 presents our implementation which is based on an experimental WWW server called *Perplex*. Section 4 compares our mechanism to related work. Finally, Section 5 gives some conclusions.

2 Structuring Web Documents

The mechanism we want to provide allows authors to configure hypertext structures in terms of predefined hypertext links (schema links), like *next* and *previous*, connecting ordinary WWW documents. In the following we call these structures *tour structures*. The documents used in a tour structure need not be aware of being used. As HTTP does not carry the information about the history of the client, we have to give the original document an additional URL which names it inside a tour structure. A document may be used several times in one or in different structures which adds one more URL for each occurrence. Retrieving the document using the new URL automatically adds the schema links to the contents of the original document, according to its position inside the tour structure.

Thus, the server side mechanism is mainly to retrieve the original document, add some decoration to it--the schema links--, and send it on to the client.

2.1 Tour Structures

Guided tours and lists have a quite simple sequential structure, as shown in Fig. 2.1. The boxes represent hypertext nodes or WWW pages respectively, as used inside the structure. Each node is related to an *original node* which may be located on some server. The arrows show hypertext links. These links allow readers to navigate through the entire structure, e.g., a guided tour.

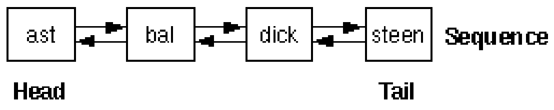


Figure 2.1: A simple sequential hypertext structure.

For example, this list may represent the staff members of a university. The nodes are named from the user’s login names and ordered alphabethically. A sequence has two designated elements: the *head* and the *tail* node which do not have a predecessor or successor, respectively. The links between the nodes have the meaning *next* and *previous* and allow readers to follow the sequence from the head to the tail and vice versa. Thus, node **ast** has a *next* link to node **bal**, node **bal** a *next* link to **dick** and a *previous* link to **ast**, etc. Additionally, sequences may be cyclic which adds a *next* link from the tail to the head and a *previous* link in the opposite direction.

Each node of the tour structure of Fig. 2.1 needs an additional URL which is composed of a URL prefix assigned to the structure and the node’s name in the structure. In our case the structure may have the URL prefix **http://sunray.cs.vu.nl/tour/staff/**. The tour structure is located on the WWW server of host **sunray.cs.vu.nl**. This server identifies the local URL **/tour/staff** as the starting point of a tour structure and acts appropriately. In the example in Fig. 2.1, the first node would have the URL **http://sunray.cs.vu.nl/tour/staff/ast**.

For the configuration of this tour structure on server **sunray.cs.vu.nl** we need to provide the URL prefix of the entire structure (here: **/tour/staff/**), the structure itself in terms of names and node order (here: **ast**, **bal**, **dick**, **steen**), and finally, a mapping of each node to its original URL (here e.g.: **ast** -> **http://www.cs.vu.nl/%7East/** etc.). This information is sufficient for our system to retrieve the original document and determine the structural *next* and *previous* links for it. It is obvious that changing the order of the nodes in the tour structure simply requires some changes in this configuration and none in the original documents.

In fact, our system allows more complex tour structures than sequential ones. To that end, we introduce some hierarchy. Fig. 2.2 shows a hierarchical structure of hypertext nodes.

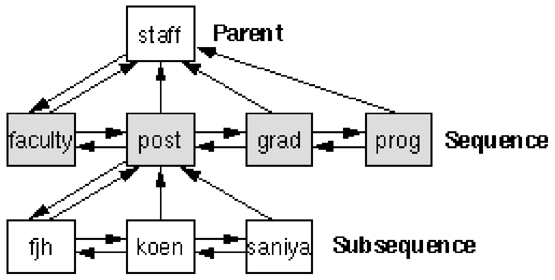


Figure 2.2: Hierarchical tour structure.

The hierarchical tour structure consists of sequences which are linked together using *next* and *previous* links. A sequence may have a parent node. All members of a sequence have an *up* link to the parent node. The parent has an additional *down* link to the head node of its subsequence. This scheme applies recursively. Thus, each member of a sequence can itself be a parent of a subsequence and so on. The simple list shown in Fig. 2.1 is only a special case of this general structure. The single parent node **staff** shown in Fig. 2.2 is itself a sequence with only one element.

Hierarchical tour structures can be used to implement guided tours with different levels of detail. In our example, the **staff** node provides a subsequence of different groups of staff members: faculty members, postdocs, graduate students, and programmers. Here, the postdoc node **post** has a subsequence providing all the postdocs in alphabethical order. The reader may stop following the subsequence and returning to the subsequence's parent node using the *up* link.

The URL of each node is composed of the URL of the sequence's parent node and an additional syllable denoting the node in a sequence, similar to the composition of UNIX pathnames using nested directories. Given the URL prefix **http://sunray.cs.vu.nl/tour**, the URL for node **fjh** in Fig. 2.2 would be **http://sunray.cs.vu.nl/tour/staff/post/fjh**. Thus, nodes have to have unique names in a sequence but not necessarily a unique name in the entire tour structure.

2.2 Decoration

The tour structure mechanism has to decorate the original document automatically with the schema links of the structure. One place to do this is the top of the document. This part is even visible if the document is longer than the browser's canvas. An alternative would be the bottom part of the document which may not be visible without scrolling on long documents. It may also be appropriate to decorate both sides of the document. Another alternative is the use of Netscape's Frames [Net96] which will be described in Section 3.4.

The decoration should be configurable by the author of the structure. It has to contain at least the *next*, *previous*, *up*, and *down* schema links. Authors may want to add short descriptive texts that name sequences and individual nodes. Additional links are introduced in Section 2.3. Fig. 2.3 shows the rendered decoration of node **post** of Fig. 2.2 containing schema links and descriptions. Here, the links are represented by clickable arrow images, but the author can configure arbitrary icons or text strings to represent schema links.

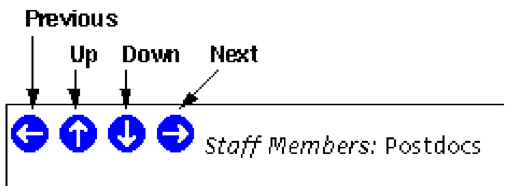


Figure 2.3: Decoration of a document’s head as part of a tour structure.

The *next* link leads to the graduate students’ node, the *previous* link to the faculty members’ node. The *down* link leads to the head of a sequence of postdocs, here to the node **fjh**. The *up* link points to the staff members’ parent node **staff**.

2.3 Meta Nodes

Sometimes authors may want to add additional schema links to every node in a tour structure. Examples are links to a detailed description of the node or to a help document explaining tour navigation. We call these linked nodes *meta nodes*.

An author of a tour structure simply adds a schema link to the configured decoration giving it a general name like **description**. Additionally, the author has to provide a mapping for each node of the tour structure to a URL representing the original contents of the meta node.

Meta nodes are addressed using the URL of the node they refer to, suffixed by an additional syllable containing the name of the schema link and the letters **%7e** which represent the tilde character. We disallow ordinary nodes to have names ending with a tilde character such that URLs of meta nodes are distinguishable from URLs of ordinary nodes. Thus, the URL for the description meta node of the node **post** of Fig. 2.2 is **http://sunray.cs.vu.nl/tour/staff/post/description%7e**.

Meta nodes are rendered with a special decoration containing an *up* link back to the node of the tour structure to which the meta node belongs.

A special meta node is the **index** node which is an automatically generated index of the current sequence. An index is a list of all nodes of the sequence listed in the order of appearance. Each entry in the list describes the node using the node’s description text provided by the author. It is clickable and leads immediately to the corresponding node of the tour structure. This allows readers to skip certain nodes of the sequence or to immediately go back to a previously visited node.

Sometimes, it is useful to have a parent node present an index of its subsequence. To that end, parent nodes may be configured individually to be decorated with an additional index of all the nodes of the subsequence. The decoration is added immediately after the top decoration. Often, a parent node with index decoration does not need any further information, and thus may have an empty node as its original node.

There are more implicit meta nodes called **head**, **tail**, and **origin**. The head and tail meta nodes refer to the head and tail nodes of the current sequence. Thus, **http://sunray.cs.vu.nl/tour/staff/post/head%7e** redirects to the URL of node **faculty** in Fig. 2.2. The meta node **origin** refers to the original node.

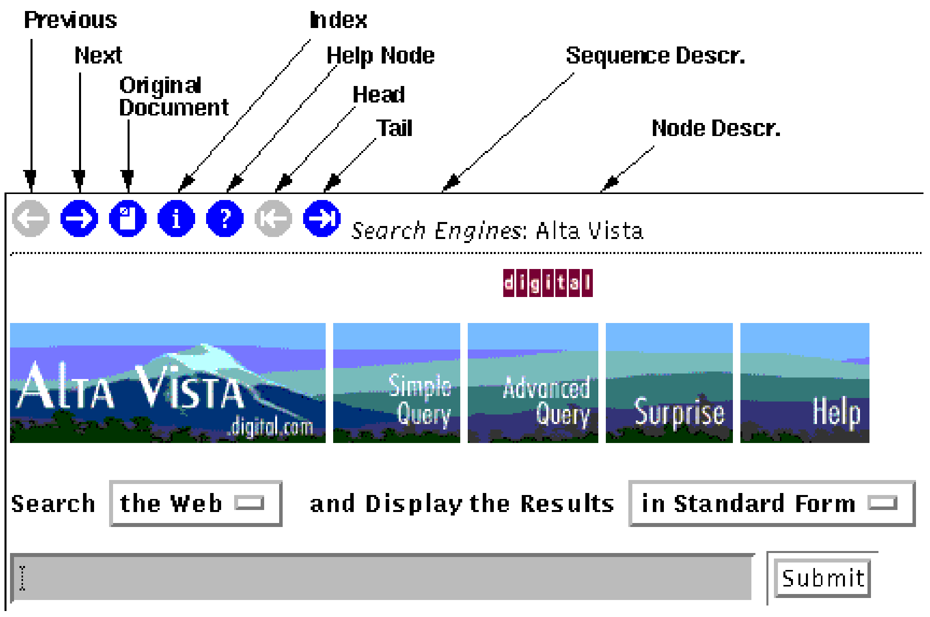
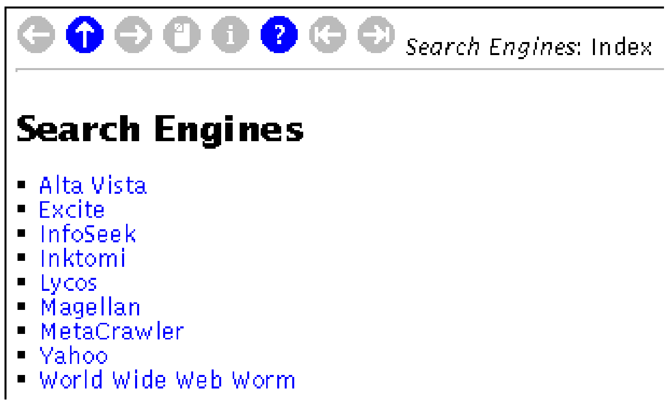


Figure 2.4: Exemplified decoration of a tour stop.

Fig. 2.4 shows the rendered first node of a sequence of the Web's search engines. Schema links are rendered in two different styles at the top of the document. When the link is not available or refers to the node itself it is rendered by a grey icon (disabled) otherwise by a blue icon (enabled).



Search Engines

- Alta Vista
- Excite
- InfoSeek
- Inktomi
- Lycos
- Magellan
- MetaCrawler
- Yahoo
- World Wide Web Worm

Figure 2.5: Exemplified decoration of an index meta node.

Fig. 2.5 shows the index node of the sequence which is shown when the reader clicks on the index icon of Fig. 2.4. The *up* link leads back to the node shown in Fig. 2.4.

3 Implementation

The implementation is based on the *Perplex* server software which is an experimental WWW server written by the author [Hau96]. The server is completely written in the *Perl 5* language [Perl96]. In principle, it would be possible to implement the ideas presented in Section 2 in any WWW server, but we will see that the special structure of Perplex's stackable DFS modules simplified the implementation task.

3.1 The Perplex Server

The Perplex server contains a main module which waits for a request from a client. Furthermore, there are a configurable number of so-called *Document File System* (DFS) modules which serve parts of the server's URL space. The name DFS is related to file systems in operating systems. In fact, DFS modules act like file systems, not on files but on Web documents. DFS modules have a generic interface which represents the contents of HTTP request/reply messages [HTTP96] by a suitable data structure.

This structure allows DFS modules to be stacked, as was done with file systems in various operating systems, e.g., in the *Spring* system [KhNe93]. An example of this is the Perplex root DFS. The main module of Perplex forwards a request from a client to the root DFS through its request interface and waits for an answer from its reply interface, which is then returned to the client. In most configurations the root DFS reads its configuration file and forwards the request to one of several other DFS modules. These serve different purposes, such as mapping URLs to UNIX files, gatewaying to the *finger* daemon, serving *CGI* scripts, and many more. These DFS modules reply to the root DFS which returns the response to the main module. Fig. 3.1 shows a typical Perplex server configuration.

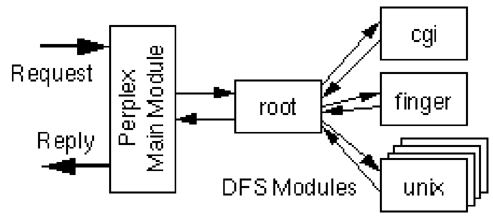


Figure 3.1: Perplex server configuration of DFS modules.

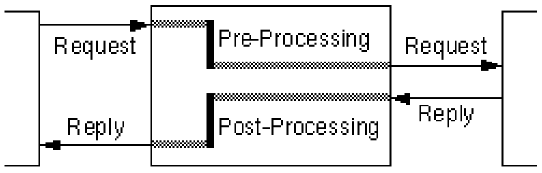


Figure 3.2: Pre- and post-processing in a stacked DFS module.

In general, stackable DFS modules can be used to pre-process a request before sending it on to another DFS, or post-process the response of an other DFS before sending the reply back to the caller, or both. For the implementation of the desired tour structure support, we will use both possibilities. Fig. 3.2 presents a stacked DFS module and shows the pre- and post-processing phases. For example, in the root DFS the pre-processing consists only of selecting the DFS to process the request according to the requested URL.

3.2 The Tour DFS

The DFS implementing our tour structure mechanism is named *Tour DFS*. When it gets a request it translates the URL of the request to their original nodes (pre-processing) according to the mapping of nodes in the tour structure. It forwards the translated request to another DFS module which retrieves the original document. The returned document is then decorated by the Tour DFS with the schema links such as *next* and *previous* (post-processing). Then, the changed document is returned to the calling DFS module, and finally to the main module which sends it to the client.

If the original document resides on the same server, the request is forwarded back to the root DFS.

In this case, a forward-trace logs all forwardings and thus detects endless recursion in bad configurations.

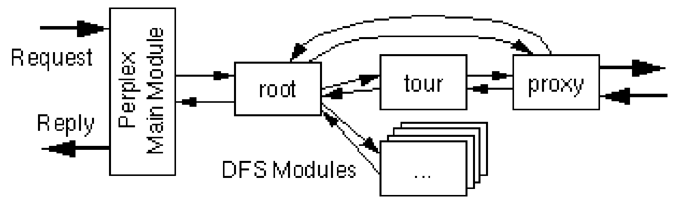


Figure 3.3: The Tour DFS configuration inside of a Perplex server.

Since we want to incorporate documents from all over the Web into a tour structure, we need to forward the request for these documents to a proxy DFS which can retrieve the documents from any server using the HTTP protocol [HTTP96]. In the current implementation the proxy DFS is also used to handle local documents because it automatically detects them and forwards requests for them back to the root DFS. Fig. 3.3 shows the configuration for the tour DFS in relation to other DFS modules.

3.3 Decoration and Configuration

As a Web page may have hypertext links expressed as so-called relative URLs [Bern94] we cannot simply assign a new URL to an original document. In this case, a relative URL would not refer to the right resource because the browser would use the new tour URL to compose an absolute URL from a relative one. Fortunately, the Hypertext Markup Language HTML [BeCo95] knows the `<BASE>` tag for providing the base URL for building absolute URLs from relative ones. In our case, it has to name the URL of the original document. It is added as part of the decoration process. If there is already a base tag in the original document it must not be changed.

If the original node is not available because its server is down or not reachable, the tour DFS creates a dummy page containing an appropriate error message. Thus, a sequence cannot be broken and a reader may go on navigating through the entire tour structure because the schema links are provided in any case, or at least as long as the tour server itself is available.

The basic decoration of all nodes of a tour structure is configured using a single configuration. This minimizes configuration work and achieves consistency. The author specifies a single string which expands to the complete decoration placed on the top or the bottom of the document or at both sides. This string may contain arbitrary HTML tags. Thus, it is easy to include static links, for example, to the author’s Web page.

On the other hand, the author may use several special commands inside of the decoration string which are expanded by the system individually on a per-node basis. To place one of the schema links, the author simply places a corresponding special command into the string.

The following schema links are available:

next, **previous**, **up**, and **down**
    These correspond to the inter-node links of a tour structure as explained in Section 2.
head and **tail**
    These schema links expand to links to the head or tail node of the current sequence.
index

This schema link connects to an automatically generated index of the current sequence.

origin

This is the schema link referring to the original node as configured within the tour structure.

user defined schema links

As explained in Section 2.3, authors may define meta nodes with corresponding schema links.

The rendering of each schema link is configured independently. It may be done in two different styles:

if-present

The decoration shows the configured rendering only if the node's position needs the schema link or if the link is available. Otherwise, the schema link is simply not rendered. In this mode head and tail nodes do not show a *previous* or *next* link, respectively.

indicating

In this mode a schema link is always shown but rendered in different styles indicating whether the schema link is available for this document or not. If a schema link is not available, e.g. the *previous* link in the head node, the author may configure its rendering as a grey shaded arrow instead of a blue one.

Additionally, the user may use commands to render the description text of the current node and/or the current sequence in the configuration string. As an example, the configured decoration string used in Fig. 2.4 and 2.5 is:

```
'%prev% %up% %down% %next% %origin% %index% %help% %head% %tail%
<I>%seqdesc%</I>: %desc%'
```

Special commands are bracketed by % characters. The *up* and *down* links in the examples are rendered using 'if-present' style, all other links use 'indicating' style. This is configured outside of the decoration string.

3.4 Frames

In the tradition of inventing its own HTML tags, Netscape Comm. Corp. added some new functionality to the newest Netscape browser software: Frames [Net96]. Frames allow several HTML windows at once in a browser's canvas, each showing a different HTML document. Clicking links in one of the frames may display a new document in another frame. The big advantage is that you can display tables of contents or other information in one frame and the main document in another frame, and each frame is independently scrollable.

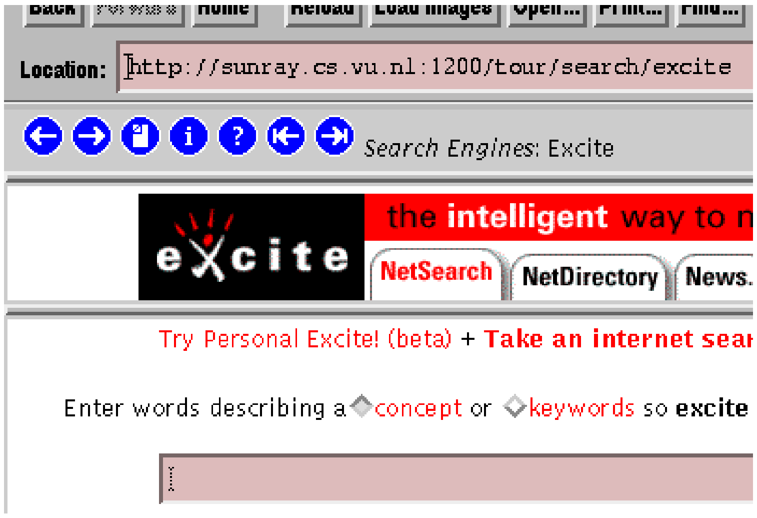
Framed documents would not show our decoration. That is why we adopt to the concept of Frames. In the current implementation of the Tour DFS, there are two ways to cope with Framed documents:

- If the original document uses Frames, we put the decoration in a static, extra Frame at the top of the document. All other documents are decorated as described so far (at the top and/or bottom of the document).

- We always turn the original document into a Framed document and display the decoration in a static, extra Frame.

Putting the tour decoration in an extra Frame has the big advantage that it is always visible even when the main Frame was scrolled.

In every case, browsers not capable of the Frame concept will get the same result as they would get with the original document but with an added decoration. The Tour DFS is thus compatible with old browser software.



http://sunray.cs.vu.nl:1200/tour/search/excite

the intelligent way to n

Try Personal Excite! (beta) + Take an internet sear

Enter words describing a concept or keywords so excite

Figure 3.4: Tour decoration using Netscape's Frames.

Fig. 3.4 shows a tour stop with an original node using Frames. The tour decoration is added as an extra Frame on top of the original Frames of the document.

3.5 Recursive Tours

Tour structures may be recursive. That means that the URL of a tour structure node is mapped itself to a URL of a node within another tour structure. This case needs special handling. Otherwise, the decoration of schema links may appear twice or more in the top and/or bottom of the document each with different links.

First, the Tour DFS signals the server of the original document that it expects it not to decorate the document as part of a tour. This is done using an additional HTTP header as shown in Fig. 3.5.

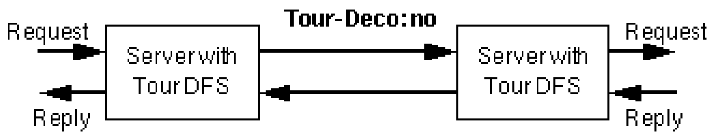


Figure 3.5: Additional HTTP header used in requests between two Tour servers.

A Tour DFS receiving this header will not decorate the original document, but send it unchanged to the first server which finally decorates it. Thus, we can pick up one node of another tour and insert it in a new tour as if it were the original node.

Second, we allow inheritance of subsequences if the author maps a tour node to another tour node. This has to be explicitly enabled on a per node basis to distinguish this situation from the first case. Imagine that the node **p** in the tour structure of Fig. 3.6a is mapped to node **post** in the structure of

Fig. 2.2. If inheritance is enabled for node **p**, the resulting structure would be as seen in Fig. 3.6b.

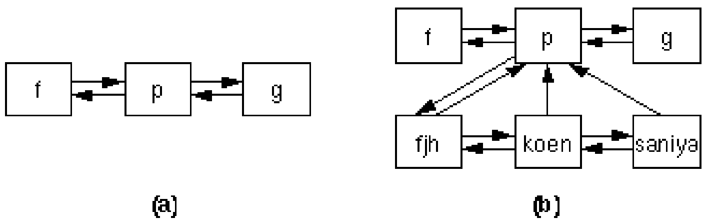


Figure 3.6: Tour structure using inheritance of subsequences.

For node **p** the first server sends a modified HTTP header to signal that inheritance is allowed:

```
Tour-Deco: no;allowdown
```

If the second server knows about a subsequence he returns the name of the first node in that sequence in a response header to the first server:

```
Tour-Deco: down="fjh"
```

Thus, the first server can add a *down* link to its decoration and can generate an own URL for the first node of the inherited subsequence.

All nodes of inherited subsequences are recognized because there is no information about them in the first server. Imagine that the URL prefix for the structure of Fig. 3.6 is **http://gig.cs.vu.nl/tour/new/**. If the first server gets the URL **http://gig.cs.vu.nl/tour/new/p/fjh**, it knows that this node is inherited via node **p**. It forwards the request to the second server and adds an additional HTTP header:

```
Tour-Deco: root="http://gig.cs.vu.nl/tour/new/p"; ref="fjh"
```

The second server can retrieve the document and decorate it. For all schema links it uses the URL prefix provided by the first server so that all schema links can be processed by this server first. The second server acknowledges that it decorated the node by an additional response header:

```
Tour-Deco: done
```

Thus, the *next* link in the decoration will have the URL **http://gig.cs.vu.nl/tour/new/p/koen** which causes similar processing in the first server again. All URLs, including meta node URLs, prefixed with **http://gig.cs.vu.nl/tour/new/p** are forwarded to the second server. So, all meta nodes of the second server are reachable as well.

This handling allows nodes of tour structures to be mapped to other nodes of tour structures. The decoration is done by the first tour structure in the chain. Subsequences may be inherited by the mapped structure and will be decorated by it. The usage of the HTTP headers also ensures that non-local servers using the same or a compatible tour structure mechanism can work together.

4 Related Work

Brooks et. al. describe in [BMMM95] the usage of proxies as application-specific transducers. In fact, a Tour DFS is such a transducer in their senses specialized for building tour structures. Brooks et. al. implemented a toolkit for building transducers which allows much more processing than is needed for tour structures. Their approach can even carry state from one request to the other while requests for tour nodes are processed without any extra state. However, a Tour DFS is a transducer on a finer level of granularity because it can transduce original documents within one server without using HTTP at all. Thus, if the nodes of tour structure are locally available, the Tour DFS does not need to call another server.

The *Boomerang* tool [DySl95] allows the reconfiguration of original Web pages using a powerful editing language. Indeed, Boomerang could be configured to do almost the same work as our tour structure mechanism. Nevertheless, Boomerang has some serious disadvantages. Boomerang is not able to fetch non-local pages. Of course, it would be possible to enhance the CGI based Boomerang tool to fetch non-local pages but the fetch engine must be completely integrated into the CGI script so that almost the same code may appear several times in the server software. All needed parameters for Boomerang have to be passed by query strings or hidden variables in form postings. This makes the URLs used for derived pages somewhat messy, and finally, it sometimes disallows original nodes from being CGI scripts.

HSDL [Kesse95] goes one step further than our tour structure mechanism and allows authors to generate a skeleton of HTML pages with all the necessary schema links out of a sophisticated graphical hypertext design tool. Thus, there is more expressivity and some guaranteed consistency for schema links in HSDL. On the other hand, HSDL does not allow the integration of non-local Web pages and it is not known how to incorporate existing local documents into the system. Finally, reconfiguring the HSDL definitions of a large hypertext document requires regeneration of the HTML pages, while with our tour structure mechanism the final pages are generated on the fly.

5 Conclusion

We introduced a mechanism for the automatic generation of schema links for hierarchical tour structures. Mostly, these structure are used to build guided tours. The nodes collected for a guided tour can be arbitrary even non-local Web pages and need not be aware of their usage in a tour structure.

The tour structure concept is a mechanism not a policy. Authors of complex hypertext documents may use our mechanism to design good or bad documents. Nevertheless, we believe that the introduced concept will help in the design of better hypertext documents.

The implementation was done using the experimental Perplex server software and especially using its stackable Document File Systems (DFS). A demo of the tour structure mechanism is available on the Perplex home page: http://www4.informatik.uni-erlangen.de/Perplex/.

6 Acknowledgments

I like to thank Saniya Ben Hassen, Raoul Bhoedjang, Greg Sharp, Maarten van Steen, Andy Tanenbaum, and the anonymous reviewers for their valuable comments on earlier drafts of this paper. This work was partially supported by the *Deutsche Forschungsgemeinschaft* in grant SFB 182 and by an EC/HCM program.

7 References

[BeCo95] T. Berners-Lee, D. Conolly: *Hypertext Markup Language - 2.0*. Request for Comments 1866, Nov. 1995. <URL:http://info.internet.isi.edu/in-notes/rfc/files/rfc1866.txt>

[Bern94] T. Berners-Lee: *Universal resource identifiers in WWW*. Request for Comments 1630, June 1994. <URL:http://info.internet.isi.edu/in-notes/rfc/files/rfc1630.txt>

[BMMM95] C. Brooks, M. S. Mazer, S. Meeks, J. Miller: "Application-specific proxy servers as HTTP stream transducers". In: Proc. of the *4th Int. World Wide Web Conf.* (Boston, Mass., Dec. 1995). <URL:http://www.w3.org/pub/Conferences/WWW4/Papers/56/>

[DySl95] C. E. Dyreson, A. M. Sloane: "The Boomerang white paper: a page as you like it". In: Proc. of the *4th Int. World Wide Web Conf.* (Boston, Mass., Dec. 1995). <URL:http://www.w3.org/pub/Conferences/WWW4/Papers/203/>

[GMP95] F. Garzotto, L. Mainette, P. Paolini: "Hypermedia design, analysis, and evaluation issues". In: *Comm. of the ACM*, **39**(8), Aug. 1995, pp. 74-86. <URL:http://space.njit.edu:5080/papers/overview.html>

[Hau96] F. Hauck: *The Perplex Home Page*. March 4, 1996. <URL:http://www4.informatik.uni-erlangen.de/Perplex/>

[HTTP96] IETF HTTP Working Group: *Current work-in-progress*. March 4, 1996. <URL:http://www.ics.uci.edu/pub/ietf/http/>

[ISB95] T. Isakowitz, E. A. Stohr, P. Balasubramanian: "RMM: a methodology for structured hypertext design": In: *Comm. of the ACM*, **39**(8), Aug. 1995, pp. 34-44. <URL:http://space.njit.edu:5080/papers/overview.html>

[Kesse95] M. Kesseler: "A schema based approach to HTML authoring" In: Proc. of the *4th Int. World Wide Web Conf.* (Boston, Mass., Dec. 1995). <URL:http://www.w3.org/pub/Conferences/WWW4/Papers2/145/>

[KhNe93] Y. A. Khalidi, M. N. Nelson: "Extensible file systems in Spring". In: Proc. of the *14th Symp. on Operating Sys. Principles* (Asheville, NC, Dec. 1993)

[Net96] Netscape Comm. Corp.: *Frame Basics*. Feb. 17, 1996. <URL:http://home.netscape.com/assist/net_sites/frame_syntax.html>

[Perl96] *The Perl 5 Home Page*. Feb. 17, 1996. <URL:http://www.metronet.com/perlinfo/perl5.html>

[Trigg88] R. H. Trigg: "Guided tours and Tabletops: tools for communicating in a hypertext environment". In: *ACM Trans. on Office Inform. Sys.*, **6**(4), Oct. 1988, pp. 398-414