Supervised Wrapper Generation with Lixto

Robert Baumgartner
DBAI, TU Wien
Favoritenstr. 9
1040 Vienna
Austria
baumgart@dbai.tuwien.ac.at

Sergio Flesca
DEIS, Univ. della Calabria
Via Pietro Bucci, 41C-42C
87030 Rende (CS)
Italy
flesca@deis.unical.it

Georg Gottlob
DBAI, TU Wien
Favoritenstr. 9
1040 Vienna
Austria
gottlob@dbai.tuwien.ac.at

Abstract

We illustrate basic features of the *Lixto* wrapper generator such as the user and system interaction, the capacious visual interface, the marking and selecting procedures, and the extraction tasks by describing the construction of a simple example program in the current *Lixto* prototype.

1. Introduction

Lixto is a fully visual and interactive wrapper generation tool whose features are described in [1]. *Lixto* allows a designer to visually create a program for regularly extracting relevant information from a changing web page or similarly structured web pages, and to generate *XML companions* of web pages. In the background, invisible to the user, the declarative logic-based language *Elog* works. A client and a servlet version of *Lixto* are implemented.

Users interact with the system to identify regions of interest and to create information filters. Wrapper generation with *Lixto* is based on examples selected by the user. First, the user opens an example page, and then she adds patterns (Fig. 1) to extract relevant information. Patterns carry user-defined names which are used as default names of XML elements. Each pattern characterises one kind of information, and consists of several filters. Each filter consists of various conditions. The instances of a pattern are the actual targets (HTML elements, list of elements, strings) which satisfy all conditions of at least one filter of the respective pattern.

When defining a filter, the user selects an example target (by simply clicking into the displayed web page) and an attribute selection mechanism, while in the background the system generates a basic *Elog* rule by choosing a suited element path and default attributes. Then the user is shown all targets matching the current filter (Fig. 2). If undesired targets are matched, she has the choice to refine the rule (for instance by specifying an element which must occur before the desired target pattern; Fig. 3). If no undesired targets are matched, but only a subset of desired ones, she can save the filter and construct another filter by using as a new example target one of the targets not matched so far. Alternately imposing conditions and adding new filters can perfectly characterise the desired information.

After finishing program generation, the program can be regularly used with the continual extractor of *Lixto* to generate XML output of a class of similar web pages.

2. Examples

This very simple example shall serve to illustrate pattern creation in detail and show how to add external conditions. Assume the user is interested in the current bestsellers of various bookshops. Therefore, she regularly checks the page *www.books.co.uk*. She does not want to revisit the page every time, but be informed with a short message to her cell-phone as soon as the first rank changes (here, we are interested in the extraction part of this information flow). First, she wants to write a program which extracts the current top-ranked bestseller. This can be achieved as follows:

1. Create a new program.
2. Open an example page (e.g. this week's page).
3. Add a new pattern to the root pattern <*document*>, call it for instance <*bestseller*> (Fig. 1).

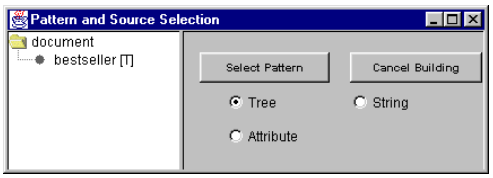

Figure 1: Adding a new pattern

Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001

Figure 2: Testing a filter

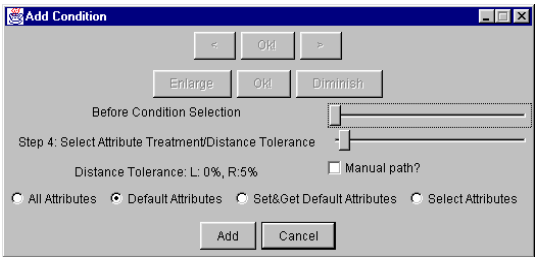

Figure 3: Adding a before element

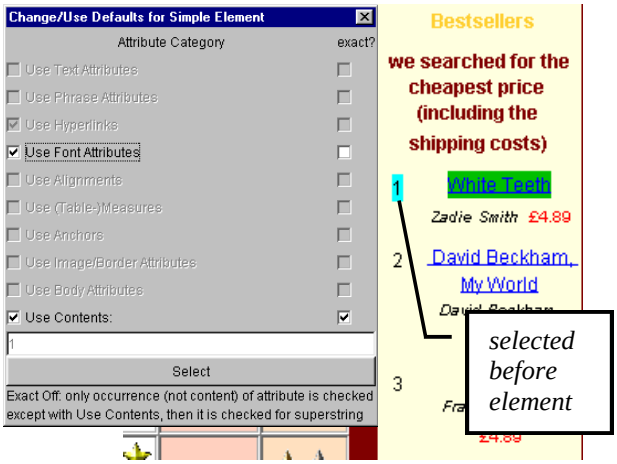

Figure 4: Marking a before element

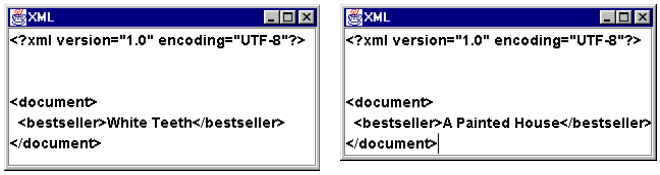

Figure 5: XML Companions

4. With two mouse clicks, mark the example target.
5. Choose default attribute settings (for this example, this means that links are characteristic attributes).
6. Test the current filter (Fig. 2).

In this case, testing the current filter returns too many targets as highlighted in Fig. 2. Each book in the Top 10 list matches this filter. As here, the user is merely interested in a single match, she has to restrict this pattern:

7. Opt to add a *before condition* expressing that a certain element must appear immediately before the desired target pattern.
8. Select the desired before element; the system allows to navigate for fine-tuning (Fig. 3).
9. Select distance tolerance settings, i.e. in which distance interval the element may occur (Fig. 3).
10. Choose *Set & Get default attributes* setting (Fig. 3).

In this kind of attribute selection (Fig. 4) we put attributes into several groups, so that users without HTML knowledge can specify that for instance, the font of a particular element is characteristic for the target and hence shall be used for extraction.

11. Choose "Use contents", and use the exact contents "1" (Fig. 4). Exact, because "1" might appear inside other elements, too, such as in some prices.
12. Test the filter. Now it extracts the desired target.
13. Save the filter, the pattern and the program.

This program can be applied to new versions of the page repeatedly by using *Lixto*'s continual extractor (Fig. 5 shows such XML companions of two different weeks).

In the same fashion, a wrapper for extracting the first three bestsellers (using a range condition) and extracting bestsellers of a particular author (after condition) can be created. Moreover, hierarchical extraction can be used to first define a book entry, and inside of it, extract the author, title and price information. Additionally, with string extraction, price and currency can be split. Other interesting examples include wrapper generation for *eBay* and *yahoo* auctions which map into a common scheme, wrapping *DBLP*, and using a simple TV query mask which accesses various *Lixto* wrappers for different TV program web sites. In such complex examples, more advanced facilities of *Lixto*, such as concepts (e.g. *date, city;* new concepts can be added interactively), following links and recursive extraction are very useful.

References[1]

[1] R. Baumgartner, S. Flesca, G. Gottlob. *Visual Web Information Extraction with Lixto*, Proc. of VLDB, 2001

[1] Consult [1] for references to related work. Future developments of *Lixto* are reported at *www.lixto.com* together with additional references.