Web Information Acquisition with Lixto Suite: A Demonstration[*]

Robert Baumgartner, Michal Ceresna, Georg Gottlob, Marcus Herzog and Viktor Zigo

DBAI, TU Wien, Favoritenstr. 9, 1040 Vienna, Austria
Lixto Software GmbH, Donau-City-Str. 1/Gate 1, 1220 Vienna, Austria

{baumgart,ceresna,gottlob,herzog,zigo}@{dbai.tuwien.ac.at,lixto.com}

Abstract

We demonstrate the Lixto Suite*, a web data extraction and transformation software kit for retrieving and converting information from various sources to various customer devices. With the* Lixto Suite*, non-technical content managers can rapidly develop applications in the areas of M-Commerce, E-Commerce, content integration and corporate portals.*

1 Introduction and Motivation

The World Wide Web represents a universe of knowledge and information. Unfortunately it is as difficult to use as a labyrinth. *Lixto Suite* technology provides tools to access, transform, and syndicate exactly the information you need. *Wrapper* technology is used to extract the relevant information from HTML documents and translate it into XML which can be easily processed. Based on a new method of identifying and extracting relevant content of HTML documents and translating it to XML format, we designed and implemented the efficient wrapper generation tool *Lixto Visual Wrapper* [0], which is particularly well-suited for visual and interactive creation of HTML to XML wrappers. Due to the internal language *Elog* [0], it is possible to create very expressive wrappers.

Lixto wrappers can be embedded into an information processing framework, the so-called *Lixto Transformation Server* [0]. *Lixto Transformation Server* features a Web-centric adminstration interface and enables application designers to format, transform, merge and deliver XML data to various devices (e.g. HTML, email, VoiceXML). The Transformation Server specifies a set of predefined components (Fig.) that can be used to specify XML data flow applications. Each component features input and output channels which pass XML documents to subsequent components (internally relying on XSLT). Customized services can be controlled from the outside by various communication media (HTTP, SMS, Web Services).

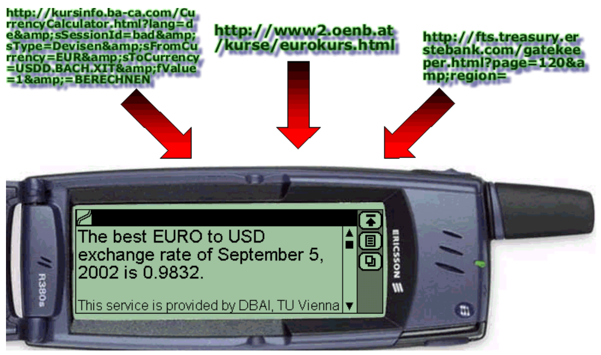

http://kursinfo.ba-ca.com/CurrencyCalculator.html?lang=de&amp;sSessionId=bad&amp;sType=Devisen&amp;sFromCurrency=EUR&amp;sToCurrency=USDD.BACH.XIT&amp;fValue=1&amp;=BERECHNEN
http://www2.oenb.at/kurse/eurokurs.html
http://fts.treasury.erstebank.com/gatekeeper.html?page=120&amp;region=
The best EURO to USD exchange rate of September 5, 2002 is 0.9832.
This service is provided by DBAI, TU Vienna
ERICSSON
R380s

Figure 1. Sample Application Result

2 Demonstration Example

We consider a very simple, yet illustrative example of a real-world problem and show how it can be handled by means of *Lixto Suite* tools. The example is a commercial one, dealing with currency exchange rates of banks: A company is interested in finding out of three banks which offers the most lucrative EUR/USD exchange rate. Each bank publishes this information on the Web. The up-to-date information should be delivered continuously to a WAP-enabled cell phone – as depicted in Figure – and by email.

We describe an informative step-by-step construction of this example within Lixto technology from the viewpoint of an application designer, who creates this application for end customers. First, in the *Lixto Transformation Server*, a designer creates a new empty information service, a so called "information pipe". Next, the operator inserts processing components and interconnects them into a unidirectional graph, reflecting the flow of processed information.

[*] All new methods and algorithms presented in this paper are covered by a pending patent. Future developments of Lixto are reported at `www.lixto.com`. A previous Lixto Suite demonstration showing a prototype of the tools received the best demo award at the WI/IAT-2001.

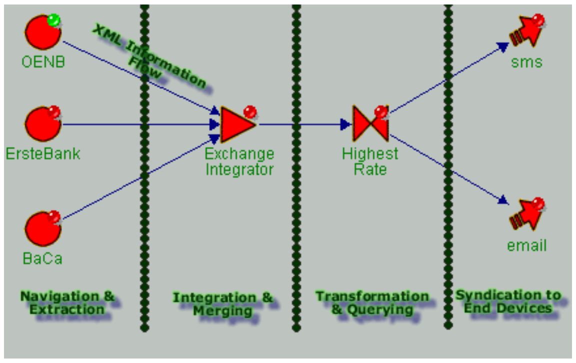


Figure 2. Pipe Architecture

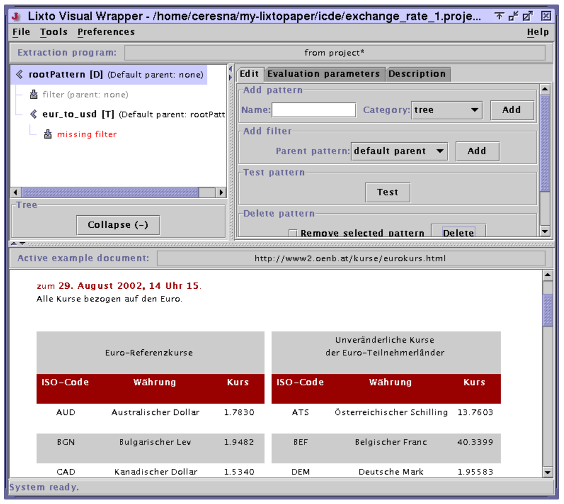
2.1 Navigating the Web

Naturally, each bank's Web site is a source of information. However, these pages have to be wrapped in order to extract the relevant information. In our case, the operator is merely interested in the actual EUR/USD exchange rate and the name of the bank. Thus, (s)he creates a new *Source* component (the disks in Figure , one for each bank), starting with one for the Austrian National Bank OENB. Using this component, the operator can browse the Web to navigate to the appropriate page that contains the rates (one can even fill forms, pass authorizations and handle cookies). The source component records all the actions, and when the service is configured, it automatically replays the navigation. Once the designer has selected the page where the actual rates occur, (s)he needs to specify how to extract the EUR/USD rates and the bank name. This process is covered by *Lixto Visual Wrapper*.

2.2 Visual Wrapper Generation

Human beings assign semantic meaning to parts of a Web page; a designer does not think of *table row* as of a set of text values, but rather of an exchange rate between Euro and foreign currencies. Therefore, the basic building block of a wrapper program is a so-called *pattern*, a container for pieces of information with the same meaning. Examples of patterns are *bank_name*, *currency* etc. Patterns are structured in a hierarchical fashion. In the lower half of the Visual Wrapper's UI one active example Web page is displayed for marking example instances. For each kind of Web page, an own wrapper has to be created; the wrapper creation for OENB is illustrated (Fig.).

In this case, the designer identifies one of the table rows as a pattern *eur_to_usd*. Hence, in the Visual Wrapper a pattern with this name (Fig.) is created. Once a pattern is created, the operator continues with visually defining a filter, a crucial part of the pattern which defines how to extract interested information from its parent pattern instances. Here,


Lixto Visual Wrapper - /home/ceresna/my-lixtopaper/icde/exchange_rate_1.proje...

File Tools Preferences Help

Extraction program: from project*

rootPattern [D] (Default parent: none)
filter (parent: none)
eur_to_usd [T] (Default parent: rootPatt
missing filter

Tree
Collapse (-)

Edit | Evaluation parameters | Description

Add pattern
Name: Category: tree Add

Add filter
Parent pattern: default parent Add

Test pattern
Test

Delete pattern
Remove selected pattern Delete

Active example document: http://www2.oenb.at/kurse/eurokurs.html

zum 29. August 2002, 14 Uhr 15.
Alle Kurse bezogen auf den Euro.

Euro-Referenzkurse			Unveränderliche Kurse der Euro-Teilnehmerländer		
ISO-Code	Währung	Kurs	ISO-Code	Währung	Kurs
AUD	Australischer Dollar	1.7830	ATS	Österreichischer Schilling	13.7603
BGN	Bulgarischer Lev	1.9482	BEF	Belgischer Franc	40.3399
CAD	Kanadischer Dollar	1.5340	DEM	Deutsche Mark	1.95583

System ready.

Figure 3. Pattern Definition

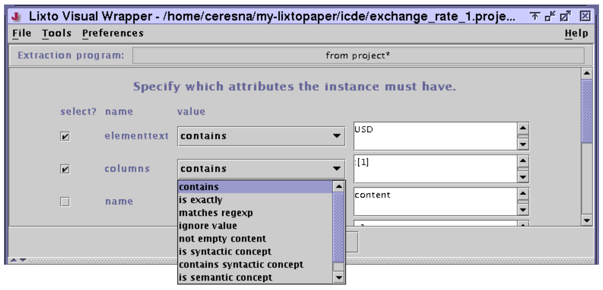
the filter extracts from the single parent instance of the predefined *rootPattern* and defines instances of *eur_to_usd*.

Defining a filter expects the operator to select an example instance with two mouse clicks on the example Web page. Filter definition continues then with optional fine-tuning of properties for the generated generalization of the chosen example. There is a possibility to customize a generated path in the HTML parse tree and the required attributes of the instance to be extracted. However, quite often as in our case it is sufficient just to accept defaults offered by the system.

It is possible to visually debug the wrapper, i.e., to test filters and patterns. Typically, operators test patterns and filters after altering the program, and upon the results decide whether to extend (i.e. add a filter) or shrink (i.e. add a condition to an existing filter) the set of matched instances. The system highlights a list of matched instances for the so-far created filter, e.g. by blinking flashlights on the Web page. In this case, all table rows are detected by the system generalisation of the filter. Because the operator is interested in one particular table row, an additional internal condition is added. While creating such a condition, the designer is asked to choose one, in this case positive, example instance (i.e. table row) and then specify with two mouse clicks what must be contained – in this case, a first cell with substring 'USD'. This is specified in the attribute fine-tuning of this condition (Fig.). Next three child patterns *exchange_rate*, *bank_name* (using the hyperlink as criterion) and *date* are added to complete this wrapper, which is subsequently embedded into a Source component on the Transformation Server.

2.3 Information Integration

After this step, the output of this source component returns the following data:


Lixto Visual Wrapper - /home/ceresna/my-lixtopaper/icde/exchange_rate_1.proje...
File Tools Preferences Help
Extraction program: from project*
contains
is exactly
matches regexp
ignore value
not empty content
is syntactic concept
contains syntactic concept
is semantic concept

Specify which attributes the instance must have.

select?	name	value	
✓	elementtext	contains	USD
✓	columns	contains	[1]
	name		content

Figure 4. Attribute Selection

```
<bank>
 <exchange-rate>0.9843</exchange-rate>
 <bank-name>OENB</bank-name>
 <date>5. September 2002</date>
<bank>
```

Afterwards the designer configures a *Scheduler* for the component to specify when the Web pages are searched for new information. In similar manner, the source components, including the navigation sequence, the wrapper and the scheduler for the other two banks are created.

As a second step, the designer needs to integrate the three sources of information into a homogeneous structure. For this, an *Integrator* component is added to which the sources are connected. In Fig. and , the three disks represent the three bank sources and the arcs the XML information flow, in this case to an integrator component depicted as triangle. In the *Integrator* UI, the operator chooses (or constructs from scratch) the desired output XML structure and maps the elements of each input XML schema to it. The designer is also offered the possibility of a content format unification. In this case, just a conceptual merge of the inputs is specified in a visual manner.

Furthermore, the designer connects the output of the integrator to a *Transformer* component (the join symbol in Figure). Here, it is sufficient to select the bank with the highest exchange rate. In general, *Transformer* components are used to join XML documents, and can be used for individual querying for different subscribed customers.

2.4 Information Syndication

At this point, the system already has all the desired information. The final step is the delivery. This step breaks down into two subtasks. At first, the internal XML data has to be rendered to the desired output format. The end customer wishes to receive notifications in plain text: “The best EUR/USD exchange rate of [date] is [value] in [bank]”. This is done by a visual presentation designer of the *Deliverer* component (see UI’s in Fig.). Secondly, the service operator determines and configures a device type where to address the message to. In our case, SMS delivery is chosen. Similarly, another deliverer component connected to the Transformer to deliver messages to email is created. A designer may also configure schedulers and conditions that decide when the results are pushed to the end user.

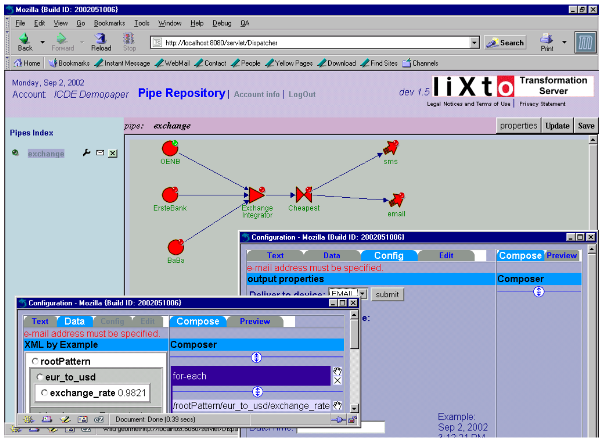

Mozilla (Build ID: 2002051006)
File Edit View Go Bookmarks Tools Window Help Debug QA
Back Forward Reload Stop Search Print
http://localhost:8080/servlet/Dispatcher
Monday, Sep 2, 2002
Account: ICDE Demopaper
Pipe Repository | Account info | LogOut
dev 1.5
lixto
Transformation Server
Pipes Index
exchange
pipe: exchange
properties Update Save
OENB
ErsteBank
BaBa
Exchange Integrator
Cheapest
sms
email
Configuration - Mozilla (Build ID: 2002051006)
Text Data Config Edit Compose Preview
e-mail address must be specified
output properties
Composer
XML by Example
rootPattern
eur_to_usd
exchange_rate 0.9821
for-each
/rootPattern/eur_to_usd/exchange_rate
submit
Example: Sep 2, 2002

Figure 5. Lixto Transformation Server

Finally, the created information pipe has to be activated and runs independently of the operator, delivering the required information to the user in frequent intervals. Due to the robustness of Lixto wrappers, the application needs little maintainance, except if the Web pages completely change their structure.

3 Conclusion and Further Scenarios

Lixto is an easily accessible Web technology based on a solid theoretical framework and a visual approach that allows application designers to define continuously running information agents fetching data from the Web. Many functions that will be tangible only in the future “Semantic Web” are turning into reality already today with *Lixto*. Besides the described sample application the actual system demonstration will cover complex applications including online press clipping, mobile applications, and portal integration. Moreover, we will show Lixto’s advantages in respect to other wrapper tools and screen-scrapers such as its high flexibility, robustness, expressiveness, and usability.

References

[1] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *Proc. of VLDB*, 2001.

[2] G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for Web Information Extraction. In *Proc. of PODS – Best paper award*, 2002.

[3] M. Herzog and G. Gottlob. InfoPipes: A flexible framework for M-Commerce applications. In *Proc. of TES*, 2001.