L00king at the Web, through <XML> glasses

CoopIS'99 – Edinburgh, Scotland

Arnaud Sahuguet
Penn Database Research Group, University of Pennsylvania

Fabien Azavant
École Nationale Supérieure des Télécommunications

Penn Database
Research Group

TELECOM
PARIS

Motivation

- The Web is a formidable medium of communication
  - millions of users (corporations, non-for-profit organization, individuals, the American Congress)
  - low entry cost
  - publishing made easy (text, sound, picture, video)
  - browsers available for free

- But how do you
  - filter hundreds of results from an AltaVista query
  - compare dozens of products from an on-line catalogue
  - “join” information from multiple Web sources

- New Challenges
  - automation
  - interoperability (Web awareness)
  - application-friendliness

Why bother? We already have XML.

- XML today
  - lots of books, (research) articles, extensions, DTDs
  - but not a single real document
- How to play with XML documents
  - Find XML documents on the Web: **good luck!**
  - Use applications with a "save as XML" feature: **maybe for Xmas**
  - Craft your own documents: **if you have nothing else to do!**
- 2 meanings for our title
  - offering XML views, because there is no real XML documents around
  - enriching data on the Web with explicit structure
- Wait a minute!
  - The Web contains zillions of HTML pages.
    HTML and XML are not so different.
  - Wouldn't it be cool to take HTML pages
    and recycle them into XML documents?

Our contribution: Web wrappers

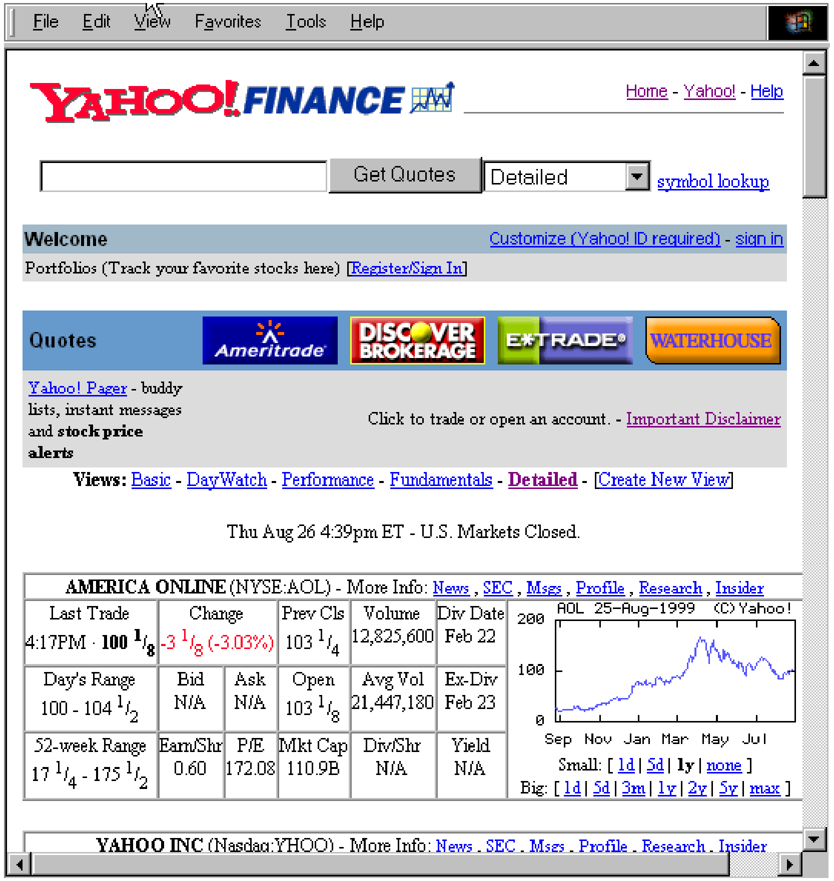
- We want to make the content of Web information sources transparently available to applications, through Web wrappers.
  And we want to export information content in a structured form like XML.

- A Web wrapper has to:
  - retrieve Web information
  - extract Web information
  - structure and export Web information

- What is the challenge here?
  - HTML is involved with layout not structure. The structure is implicit.
  - HTML has no clean syntax. The one from the Web, not the one from the specs.
  - How to offer an expressive and high-level way to extract some specific information from a Web page and map it to XML?

Here comes the World Wide Web Wrapper Factory...

Put the glasses on



YAHOO! FINANCE

Home - Yahoo! - Help

Get Quotes | Detailed | symbol lookup

Welcome Customize (Yahoo! ID required) - sign in

Portfolios (Track your favorite stocks here) [Register/Sign In]

Quotes Ameritrade DISCOVER BROKERAGE E*TRADE WATERHOUSE

Yahoo! Pager - buddy lists, instant messages and **stock price alerts**

Click to trade or open an account. - Important Disclaimer

Views: Basic - DayWatch - Performance - Fundamentals - **Detailed** - [Create New View]

Thu Aug 26 4:39pm ET - U.S. Markets Closed.

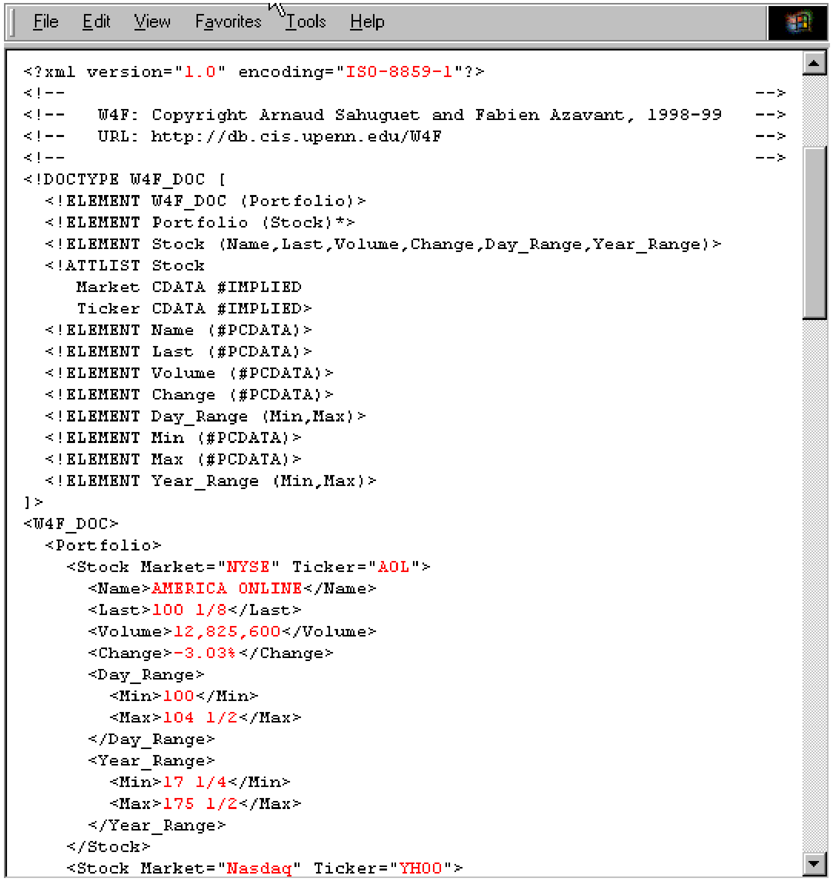
AMERICA ONLINE (NYSE:AOL) - More Info: News , SEC , Msgs , Profile , Research , Insider

Last Trade 4:17PM · **100 $^{1}/_{8}$**	Change -3 $^{1}/_{8}$ (-3.03%)		Prev Cls 103 $^{1}/_{4}$	Volume 12,825,600	Div Date Feb 22
Day's Range 100 - 104 $^{1}/_{2}$	Bid N/A	Ask N/A	Open 103 $^{1}/_{8}$	Avg Vol 21,447,180	Ex-Div Feb 23
52-week Range 17 $^{1}/_{4}$ - 175 $^{1}/_{2}$	Earn/Shr 0.60	P/E 172.08	Mkt Cap 110.9B	Div/Shr N/A	Yield N/A

Small: [1d | 5d | **1y** | none]

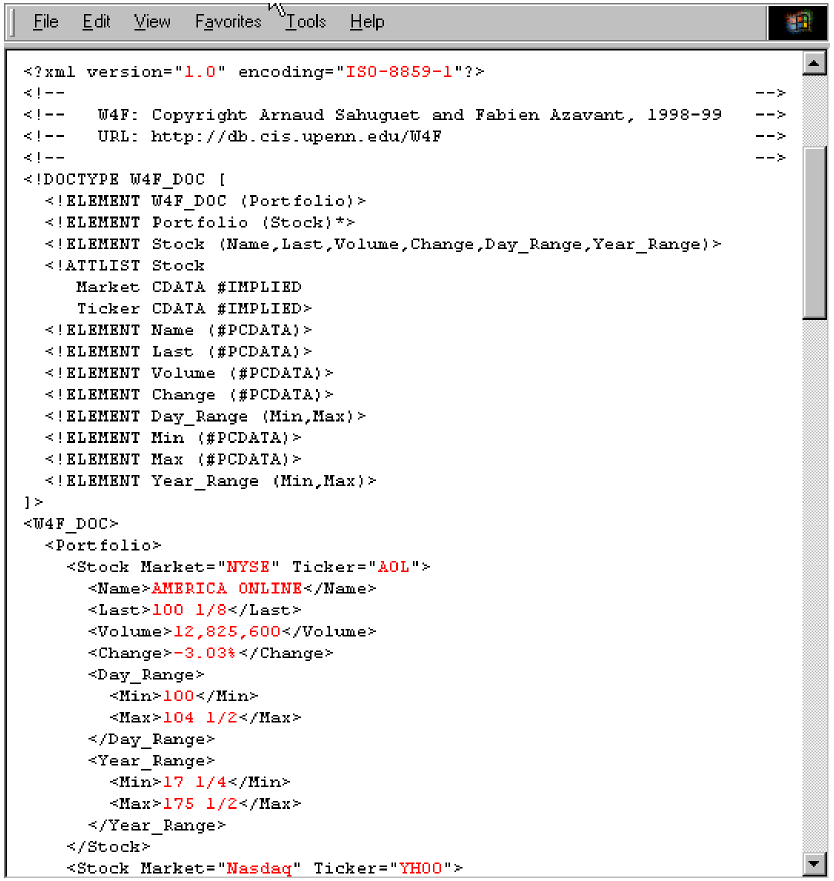
Big: [1d | 5d | 3m | 1y | 2y | 5y | max]

YAHOO INC (Nasdaq:YHOO) - More Info: News . SEC . Msgs . Profile . Research . Insider



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--                                                                -->
<!--   W4F: Copyright Arnaud Sahuguet and Fabien Azavant, 1998-99   -->
<!--   URL: http://db.cis.upenn.edu/W4F                             -->
<!--                                                                -->
<!DOCTYPE W4F_DOC [
  <!ELEMENT W4F_DOC (Portfolio)>
  <!ELEMENT Portfolio (Stock)*>
  <!ELEMENT Stock (Name,Last,Volume,Change,Day_Range,Year_Range)>
  <!ATTLIST Stock
     Market CDATA #IMPLIED
     Ticker CDATA #IMPLIED>
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Last (#PCDATA)>
  <!ELEMENT Volume (#PCDATA)>
  <!ELEMENT Change (#PCDATA)>
  <!ELEMENT Day_Range (Min,Max)>
  <!ELEMENT Min (#PCDATA)>
  <!ELEMENT Max (#PCDATA)>
  <!ELEMENT Year_Range (Min,Max)>
]>
<W4F_DOC>
  <Portfolio>
    <Stock Market="NYSE" Ticker="AOL">
      <Name>AMERICA ONLINE</Name>
      <Last>100 1/8</Last>
      <Volume>12,825,600</Volume>
      <Change>-3.03%</Change>
      <Day_Range>
        <Min>100</Min>
        <Max>104 1/2</Max>
      </Day_Range>
      <Year_Range>
        <Min>17 1/4</Min>
        <Max>175 1/2</Max>
      </Year_Range>
    </Stock>
    <Stock Market="Nasdaq" Ticker="YHOO">
```

Put the glasses on



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--                                                                -->
<!--   W4F: Copyright Arnaud Sahuguet and Fabien Azavant, 1998-99  -->
<!--   URL: http://db.cis.upenn.edu/W4F                             -->
<!--                                                                -->
<!DOCTYPE W4F_DOC [
  <!ELEMENT W4F_DOC (Portfolio)>
  <!ELEMENT Portfolio (Stock)*>
  <!ELEMENT Stock (Name,Last,Volume,Change,Day_Range,Year_Range)>
  <!ATTLIST Stock
     Market CDATA #IMPLIED
     Ticker CDATA #IMPLIED>
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Last (#PCDATA)>
  <!ELEMENT Volume (#PCDATA)>
  <!ELEMENT Change (#PCDATA)>
  <!ELEMENT Day_Range (Min,Max)>
  <!ELEMENT Min (#PCDATA)>
  <!ELEMENT Max (#PCDATA)>
  <!ELEMENT Year_Range (Min,Max)>
]>
<W4F_DOC>
  <Portfolio>
    <Stock Market="NYSE" Ticker="AOL">
      <Name>AMERICA ONLINE</Name>
      <Last>100 1/8</Last>
      <Volume>12,825,600</Volume>
      <Change>-3.03%</Change>
      <Day_Range>
        <Min>100</Min>
        <Max>104 1/2</Max>
      </Day_Range>
      <Year_Range>
        <Min>17 1/4</Min>
        <Max>175 1/2</Max>
      </Year_Range>
    </Stock>
    <Stock Market="Nasdaq" Ticker="YHOO">
```

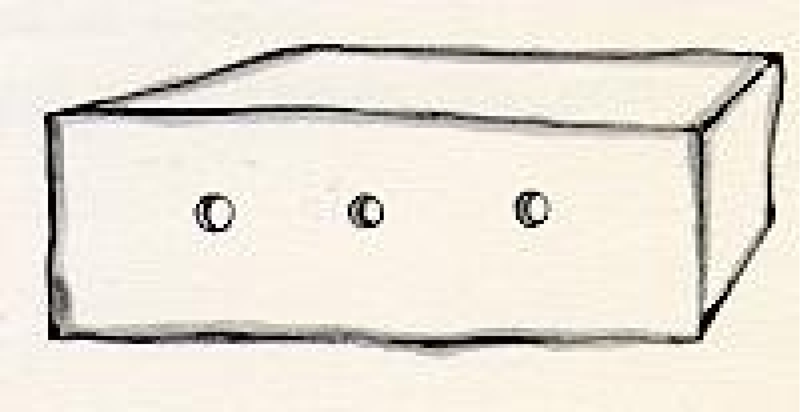
» If you please – draw me a wrapper...«

»If you please - draw me a wrapper...«

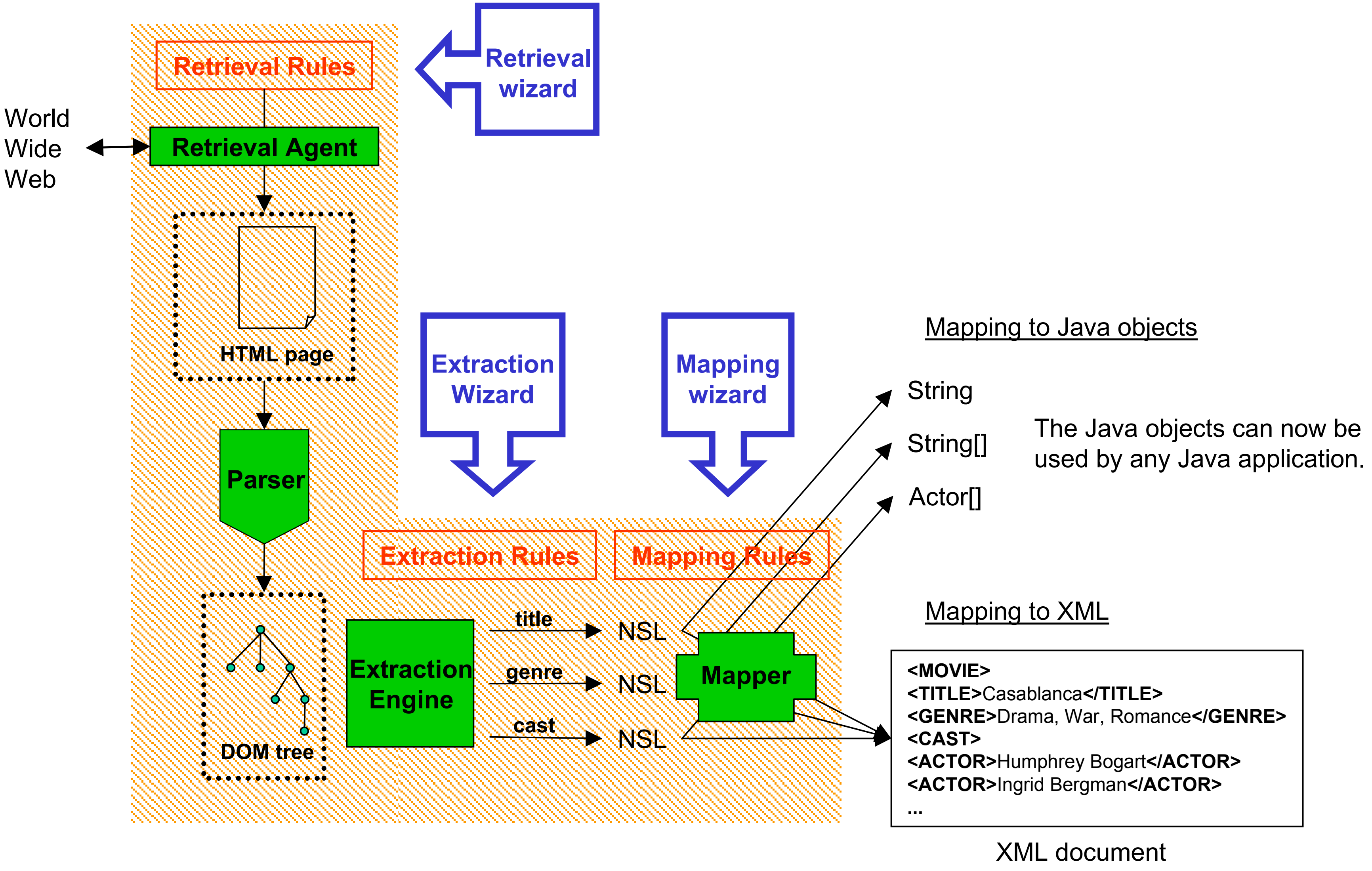
When a mystery is too overpowering, one dare not disobey. Absurd as it might seem to me, a thousand miles from any human habitation and in danger of death, I took out of my pocket a sheet of paper and my fountain-pen. But then I remembered how my studies had been concentrated on geography, history, arithmetic, and grammar, and I told the little chap (a little crossly, too) that I did not know how to draw. He answered me:

»That doesn't matter. Draw me a wrapper...«

The Wrapper is inside the box



W4F wrapper architecture



Retrieval Rules

Retrieval wizard

World Wide Web

Retrieval Agent

HTML page

Parser

DOM tree

Extraction Wizard

Mapping wizard

Extraction Rules

Mapping Rules

Extraction Engine

title

genre

cast

NSL

NSL

NSL

Mapper

Mapping to Java objects

String

String[]

Actor[]

The Java objects can now be used by any Java application.

Mapping to XML

```
<MOVIE>
<TITLE>Casablanca</TITLE>
<GENRE>Drama, War, Romance</GENRE>
<CAST>
<ACTOR>Humphrey Bogart</ACTOR>
<ACTOR>Ingrid Bergman</ACTOR>
...
```

XML document

World Wide Web Wrapper Factory (W4F)

- What W4F is not
  - it is not a query language
  - it is not a mediator system

- W4F is
  - a toolkit to generate wrappers for Web information sources
  - it consists of:
    - an extraction language called HEL (HTML Extraction Language)
    - a mapping language
    - some GUI wizards

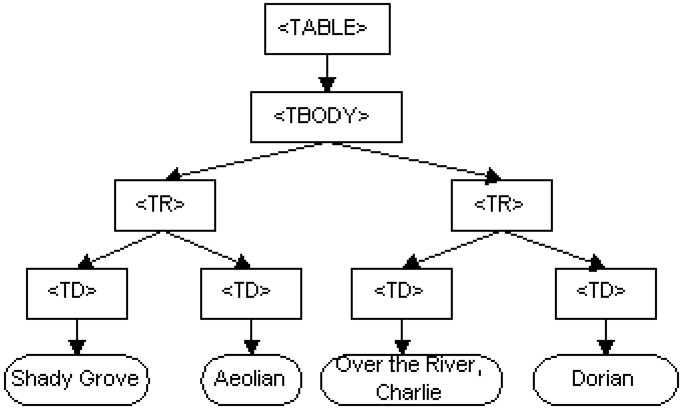
- CAVEAT
  - A given W4F wrapper deals with one type of Web pages.
    To wrap a movie database, one will need a wrapper for movie pages and a wrapper for actor pages for instance.

HTML Extraction Language (HEL)

- Tree-based data-model
  - an HTML page is seen as a labeled tree (DOM$^{\text{Document Object Model}}$)
- Tree navigation via path-expressions (with conditions)
  - extraction rules are described as paths along the tree
  - path expressions always return text values
- Regular expression
  - regular expressions (à la Perl) can be applied on text values to capture finer granularity

```
<TABLE> <TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

HTML



Tree à la DOM

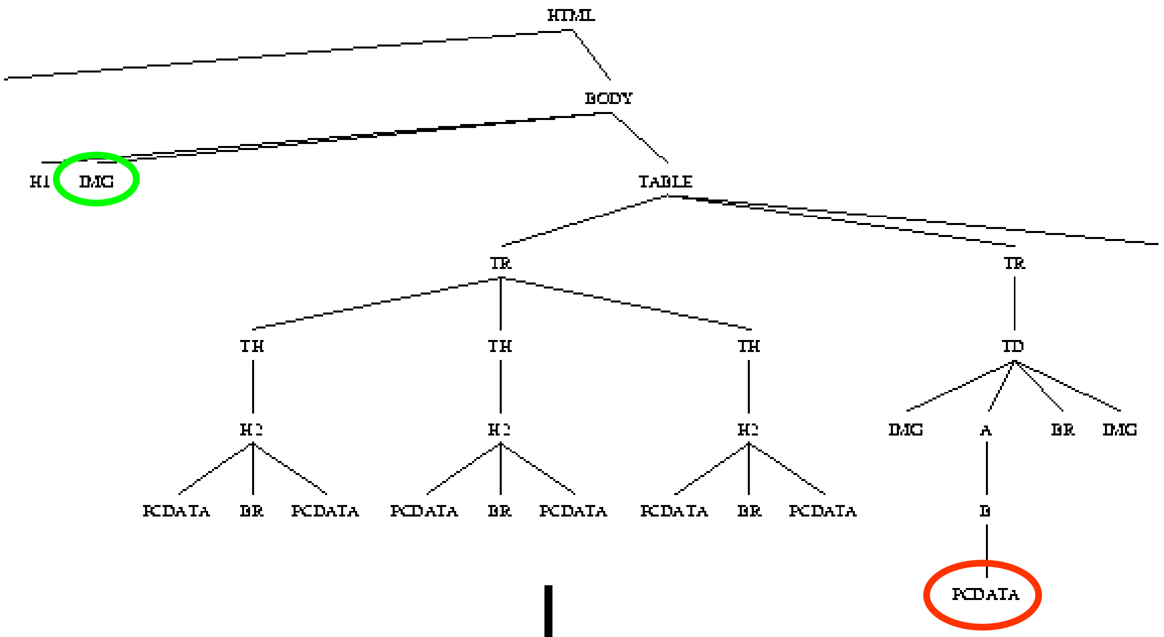
Tree navigation

- Following the document hierarchy: “.”
  - “.” explores the immediate children of a node
  - useful for limited nested structures

- Following the document flow: “->”
  - “->” explores the nodes found along a depth-first search
  - useful to create shortcuts
  - “->” only stops when it reaches the end

- When accessing nodes, index ranges can be used
  - e.g.. html.body->a[*].txt
  - e.g.. html.body.table[0].tr[1-].td[0].txt
  - returns a collection of nodes

2 ways to navigate the tree



HIERARCHICAL NAVIGATION

html.body.img[0].getAttr(src)
html.body.table[0].tr[1].td[0].a[0].b[0].pcdata[0].txt

FLOW NAVIGATION

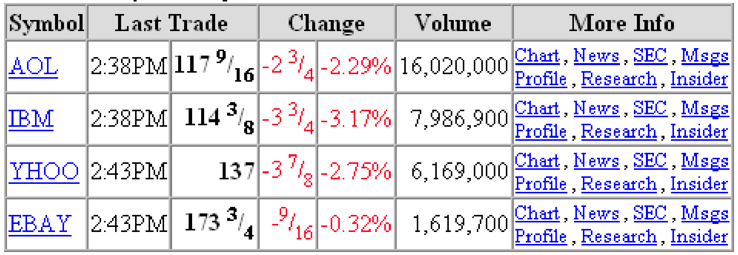
Using "->", there are more than 1 way to get to a node

html->img[0].getAttr(src)
html.h1[0]->img[0].getAttr(src)
html->tr[1]->pcdata[0].txt
html->pcdata[7].txt

Using conditions

- Sometimes, we do not know ahead of time where exactly the information is located. Take the example of the IBM stock.

Let us assume that this table corresponds to table[5] inside the HTML page.



Symbol	Last Trade		Change		Volume	More Info
AOL	2:38PM	117 $^{9}/_{16}$	-2 $^{3}/_{4}$	-2.29%	16,020,000	Chart , News , SEC , Msgs Profile , Research , Insider
IBM	2:38PM	114 $^{3}/_{8}$	-3 $^{3}/_{4}$	-3.17%	7,986,900	Chart , News , SEC , Msgs Profile , Research , Insider
YHOO	2:43PM	137	-3 $^{7}/_{8}$	-2.75%	6,169,000	Chart , News , SEC , Msgs Profile , Research , Insider
EBAY	2:43PM	173 $^{3}/_{4}$	$^{9}/_{16}$	-0.32%	1,619,700	Chart , News , SEC , Msgs Profile , Research , Insider

- You can write the following extraction rule:

```
                html->table[5].tr[i].td[2].txt
      where     html->table[5].tr[i].td[0].txt = “IBM”
```

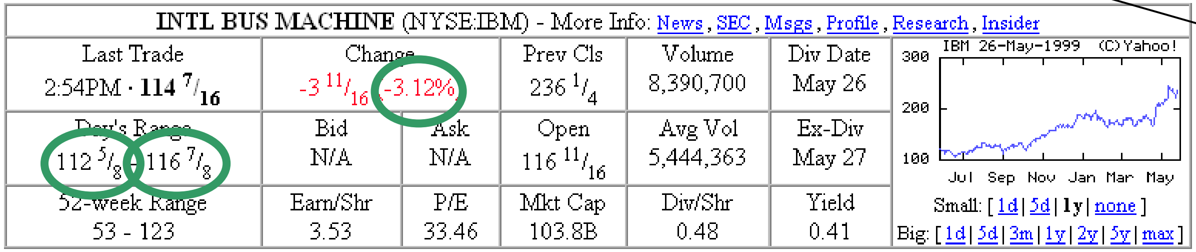
- Conditions involve index ranges only.
- Conditions are resolved against node properties, not nodes themselves.

Using regular expressions

- In some cases, we want to go deeper than the tag structure.

- We want to extract the % change
  - table.tr[1].td[1].txt, **match /[(](.*?)[)]/**
- We want to extract the day’s range for the stock:
  - table.tr[2].td[0].txt, **match/Day’s Range (.*)/, split /-/**

regular expression operators can be used in cascade



INTL BUS MACHINE (NYSE:IBM) - More Info: News , SEC , Msgs , Profile , Research , Insider

Last Trade 2:54PM · **114 $^{7}/_{16}$**	Change -3 $^{11}/_{16}$ (-3.12%)		Prev Cls 236 $^{1}/_{4}$	Volume 8,390,700	Div Date May 26	
Day's Range 112 $^{5}/_{8}$ - 116 $^{7}/_{8}$	Bid N/A	Ask N/A	Open 116 $^{11}/_{16}$	Avg Vol 5,444,363	Ex-Div May 27	
52-week Range 53 - 123	Earn/Shr 3.53	P/E 33.46	Mkt Cap 103.8B	Div/Shr 0.48	Yield 0.41	Small: [1d | 5d | **1y** | none] Big: [1d | 5d | 3m | 1y | 2y | 5y | max]

- Semantics
  - match /(.....)/ returns a string
  - match /(...) (...)/ returns a list of strings
  - split /...../ returns a list of strings

Building Complex Structures

- Atomic values are not enough.

- The fork operator “#” permits to follow a path along various subpaths. Results are put together into a list.

- Following the previous example, we can extract the entire stock information and put it in one structure.

```
html.body.center.table[i:*]
    ( .tr[0].td[0].b[0].txt                                         // name
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /[(](.*?):/           // trading place
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /:(.*?)[)]/           // ticker
    # .tr[1].td[0].b[0].txt                                         // last trade
    # .tr[1].td[3].pcdata[1].txt                                    // volume
    # .tr[1].td[1].txt, match /[(](.*?)[)]/                         // change %
    # .tr[2].td[0].txt, match /Range(.*)/, split /-/                // Day range
    # .tr[3].td[0].txt, match /Range(.*)/, split /-/                // Year range
    )
where html.body.center.table[i].tr[0].td[0].getAttr(colspan) = "7";
```

Mapping the extracted information

- W4F represents the extracted information as Nested String Lists

  -
    ```
    NSL    ::    null
           |     String
           |     list(NSL)
    ```
- Leaf nodes create strings.
- Lists are created by index ranges, forks and regex operators.
- Invalid paths create null.

- NSLs are anonymous and expressive enough to capture complex structures

- NSLs can be manipulated via an API.

- However they are not suitable for high-level application development.

We need a mapping.

W4F Mappings

- W4F offers
  - a default mapping to Java base types for homogenous NSLs
  - a programmatic way to define custom mapping via Java classes
  - declarative specifications for specific target structures
    - K2 mediation system
    - XML

- XML mapping
  - An XML mapping expresses how to create XML elements out of NSLs.
  - An XML mapping is described via declarative rules called *templates* (much more concise to write than DTDs)
  - Templates are nested structures composed of *leaves*, *lists* and *records.*
  - *The structure of XML templates is similar to extraction rules.*
  - From a template, it is straightforward* to infer a DTD.

XML Templates

Leaf Templates

- .Ticker
  - <!ELEMENT Ticker #PCDATA>
  - <Ticker>IBM</Ticker>
- .Ticker (.Symbol^ # `stuff)
  - <!ELEMENT Ticker `stuff>
    <!ATTLIST Symbol CDATA #IMPLIED>
  - <Ticker Symbol="IBM">`stuff</Ticker>
- .Ticker!Symbol
  - <!ELEMENT Ticker EMPTY>
    <!ATTLIST Symbol CDATA #IMPLIED>
  - <Ticker Symbol="IBM"/>

List Templates

- .Portfolio*.templ
  - <!ELEMENT Portfolio templ*>

  -
    ```
    <Portfolio>
      <templ>…</templ>
      <templ>…</templ>
    </Portfolio>
    ```

Record Templates

- .Stock (T1 # … # Tn)
  - <!ELEMENT Stock (T1,…,Tn)>

  -
    ```
    <Stock>
      <T1>…</T1>
    …
      <Tn>…</Tn>
    ```

```
Template   := Leaf | Record | List
Leaf       := . Tag | . Tag ^ | . Tag ! Tag
List       := . Tag Flatten Template
Record     := . Tag ( TemplList )
Flatten    := * | * Flatten
TemplList  := Template | Template # TemplList
Tag        := string
```

The full wrapper

EXTRACTION_RULES

```
html.body.center.table[i:*]
    ( .tr[0].td[0].b[0].txt                                       // name
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /[(](.*?):/         // trading place
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /:(.*?)[)]/         // ticker
    # .tr[1].td[0].b[0].txt                                       // last trade
    # .tr[1].td[3].pcdata[1].txt                                  // volume
    # .tr[1].td[1].txt, match /[(](.*?)[)]/                       // change %
    # .tr[2].td[0].txt, match /Range(.*)/, split /-/              // Day range
    # .tr[3].td[0].txt, match /Range(.*)/, split /-/              // Year range
    )
where html.body.center.table[i].tr[0].td[0].getAttr(colspan) = "7";
```

XML_MAPPING

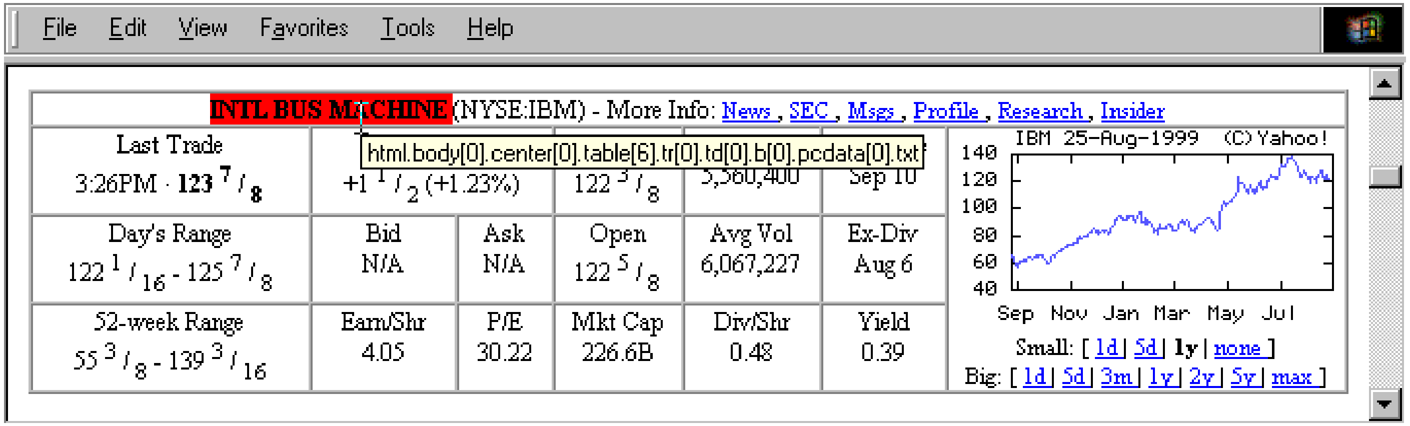
```
.Portfolio*.Stock (
    .Full_Name^
  # .Market^
  # .Ticker^
  # .Last
  # .Volume
  # .Change
  # .Day_Range ( .Min # .Max )
  # .Year_Range ( .Min # .Max )
    );
```

RETRIEVAL_RULES

```
METHOD: GET;
URL: "http://finance.yahoo.com/q?s=AOL+YHOO+IBM+CSCO+LU+EBAY+TXN+EGRP+NOK&d=t";
```

GUI support: Extraction Wizard

- Motivation
  - WYSIWYG
  - simple



INTL BUS MACHINE (NYSE:IBM) - More Info: News , SEC , Msgs , Profile , Research , Insider

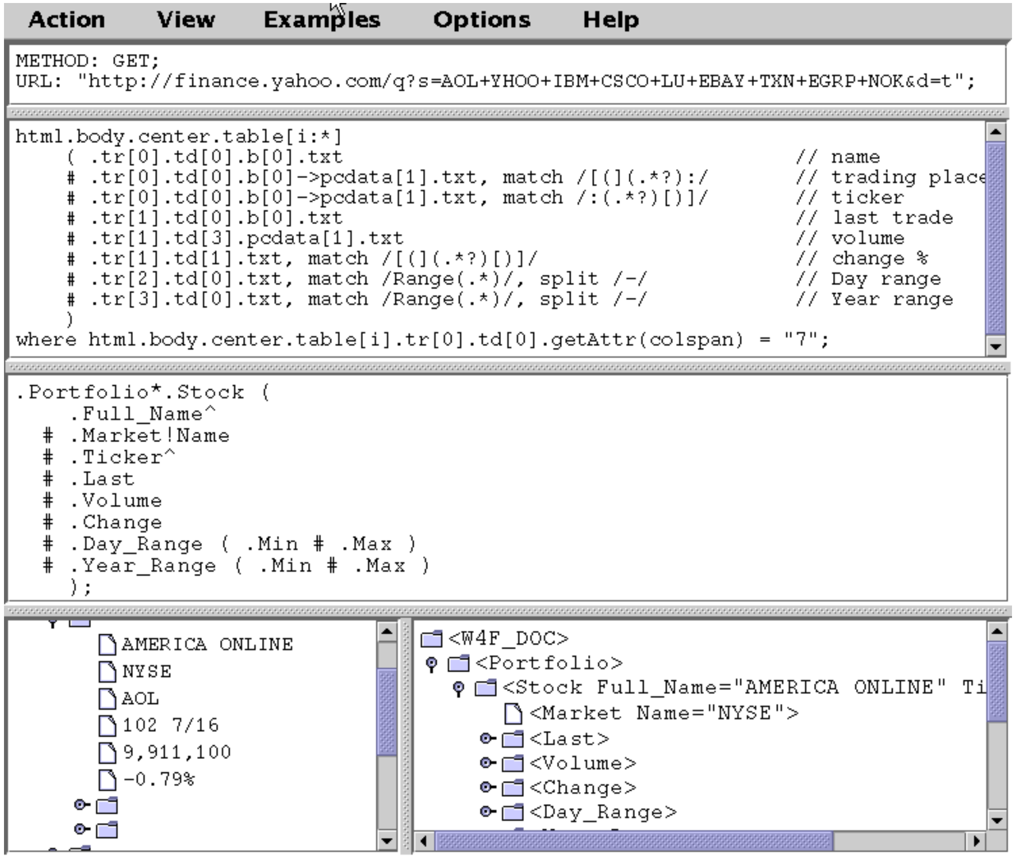
html.body[0].center[0].table[6].tr[0].td[0].b[0].pcdata[0].txt

Last Trade 3:26PM · **123** $^{7}/_{8}$	+1 $^{1}/_{2}$ (+1.23%)		122 $^{3}/_{8}$	5,560,400	Sep 10
Day's Range 122 $^{1}/_{16}$ - 125 $^{7}/_{8}$	Bid N/A	Ask N/A	Open 122 $^{5}/_{8}$	Avg Vol 6,067,227	Ex-Div Aug 6
52-week Range 55 $^{3}/_{8}$ - 139 $^{3}/_{16}$	Earn/Shr 4.05	P/E 30.22	Mkt Cap 226.6B	Div/Shr 0.48	Yield 0.39

Small: [1d | 5d | **1y** | none]
Big: [1d | 5d | 3m | 1y | 2y | 5y | max]

GUI support: Applet Wizard

- Motivation
  - all-in-one GUI
  - for the applet, extraction rules are interpreted (not compiled)



Retrieval

```
METHOD: GET;
URL: "http://finance.yahoo.com/q?s=AOL+YHOO+IBM+CSCO+LU+EBAY+TXN+EGRP+NOK&d=t";
```

Extraction

```
html.body.center.table[i:*]
    ( .tr[0].td[0].b[0].txt                                       // name
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /[(](.*?):/        // trading place
    # .tr[0].td[0].b[0]->pcdata[1].txt, match /:(.*?)[)]/        // ticker
    # .tr[1].td[0].b[0].txt                                       // last trade
    # .tr[1].td[3].pcdata[1].txt                                  // volume
    # .tr[1].td[1].txt, match /[(](.*?)[)]/                       // change %
    # .tr[2].td[0].txt, match /Range(.*)/, split /-/              // Day range
    # .tr[3].td[0].txt, match /Range(.*)/, split /-/              // Year range
    )
where html.body.center.table[i].tr[0].td[0].getAttr(colspan) = "7";
```

XML mapping

```
.Portfolio*.Stock (
    .Full_Name^
  # .Market!Name
  # .Ticker^
  # .Last
  # .Volume
  # .Change
  # .Day_Range ( .Min # .Max )
  # .Year_Range ( .Min # .Max )
    );
```

NSL tree

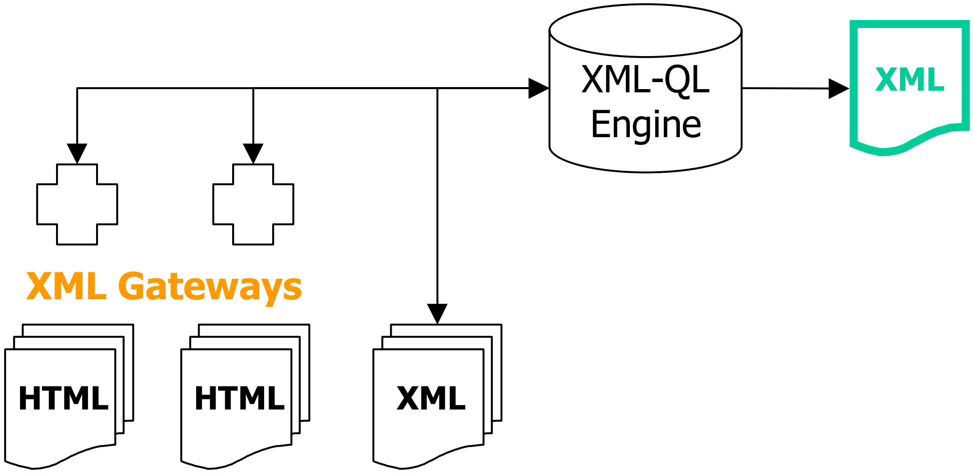
```
AMERICA ONLINE
NYSE
AOL
102 7/16
9,911,100
-0.79%
```

XML document

```
<W4F_DOC>
  <Portfolio>
    <Stock Full_Name="AMERICA ONLINE" Ti
      <Market Name="NYSE">
      <Last>
      <Volume>
      <Change>
      <Day_Range>
```

What can you do with your glasses on?

- XML integration using XML-QL
  - XML documents are constructed on-the-fly by XML Gateways, from HTML pages
  - XML documents are restructured by XML-QL
  - the result is exported as an XML document



XML-QL Integration Example

```
CONSTRUCT <Joint_Work>
        <Movie>$title1</>
        <Year>$year1</>
        </>
WHERE
      <W4F_DOC.Actor NAME=$n1>
         <Filmography.Movie>
          <Title>$title1</>
            <Year>$year1</>
       </>
      </> IN "http://db.cis.upenn.edu/cgi-bin/serveXML?XML=XML&SERVICE=IMDB_Actor&URL=http://us.imdb.com/Name?Bogart,+Humphrey",

      <W4F_DOC.Actor NAME=$n2>
         <Filmography.Movie>
          <Title>$title2</>
            <Year>$year2</>
       </>
      </> IN "http://db.cis.upenn.edu/cgi-bin/serveXML?XML=XML&SERVICE=IMDB_Actor&URL=http://us.imdb.com/Name?Bacall,+Lauren",

      text($title1) = text($title2)
```

- The full example can be found at: http://db.cis.upenn.edu/W4F/Examples/Integration

Experience with W4F

- Wrappers
  - MedLine, Yahoo!, Internet Movie Database, CIA World Factbook, IBM Patent Server, AltaVista, Stock Market Quotes, E-commerce (CDs), etc.
- Web Applications
  - XML gateways, TV-Agent, French White pages, etc.
- Integration
  - W4F wrappers are being used by the K2 mediation system.
  - W4F wrappers can be called from XML-QL.

Now that the extraction of information is granted, applications can focus on value-added services.

W4F Contributions

- Features
  - declarative specification (conciseness)
  - independent layers
  - high-level extraction language (2 navigations, conditions, regex, fork)
  - high-level mappings
  - lightweight ready-to-go Java components (less than 5kb for a wrapper)
  - visual support

- Benefits
  - higher productivity (wrappers are written in minutes)
  - robustness
  - easy maintenance
  - embeddability (small footprint)

Related and Future Work

- Related work
  - Wrapper Generation Project (Univ. Maryland), XWRAP (OGI)
  - JEDI (GMD), Araneus (Roma3)
  - Ariadne (ISI/USC), Wrapper Induction (Kushmerick)
  - WIDL (webMethods)

- Future work
  - extending HEL (document navigation, hyperlinks, etc.)
  - extensions to the mapping language
  - using Machine-Learning to help generate robust extraction rules
  - going beyond extraction
  - engineering (commercial version now available)

- The W4F prototype will be presented at VLDB'99. See you there.

Visit our Web site at http://db.cis.upenn.edu/W4F
and download the software.

Recycled by W4F/HEL

Powered by W4F/HEL