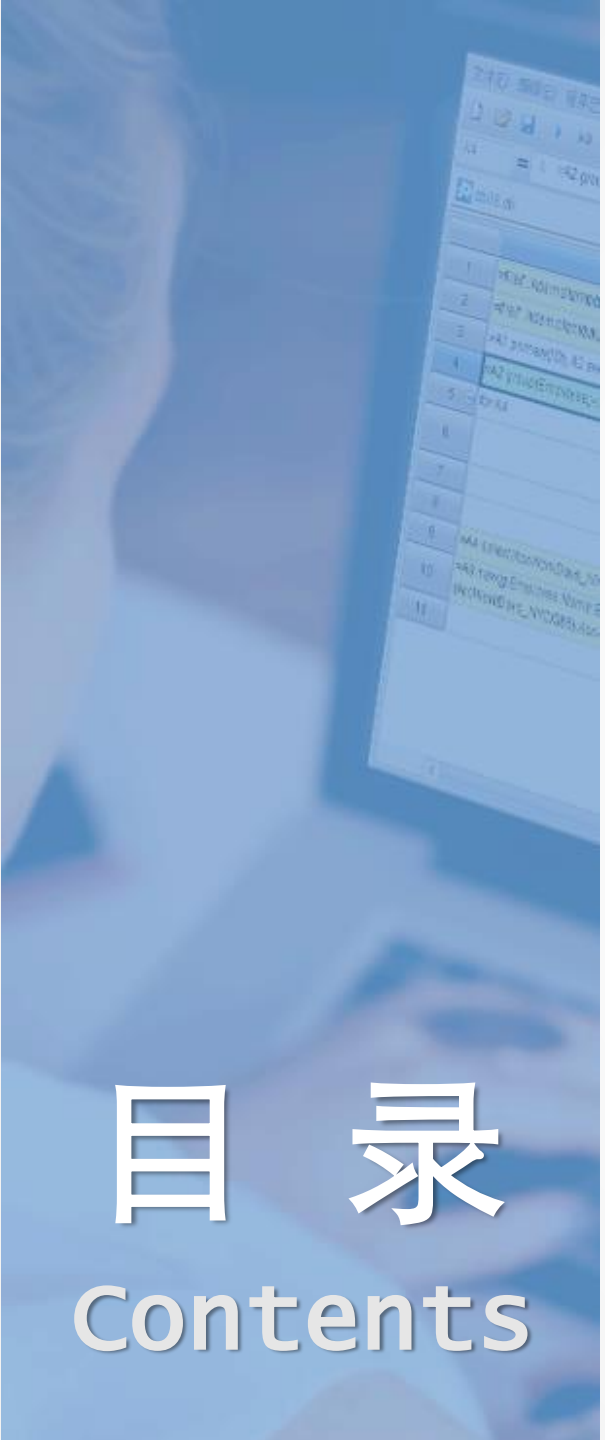
集算器

创新大数据计算引擎

XML数据解析与计算

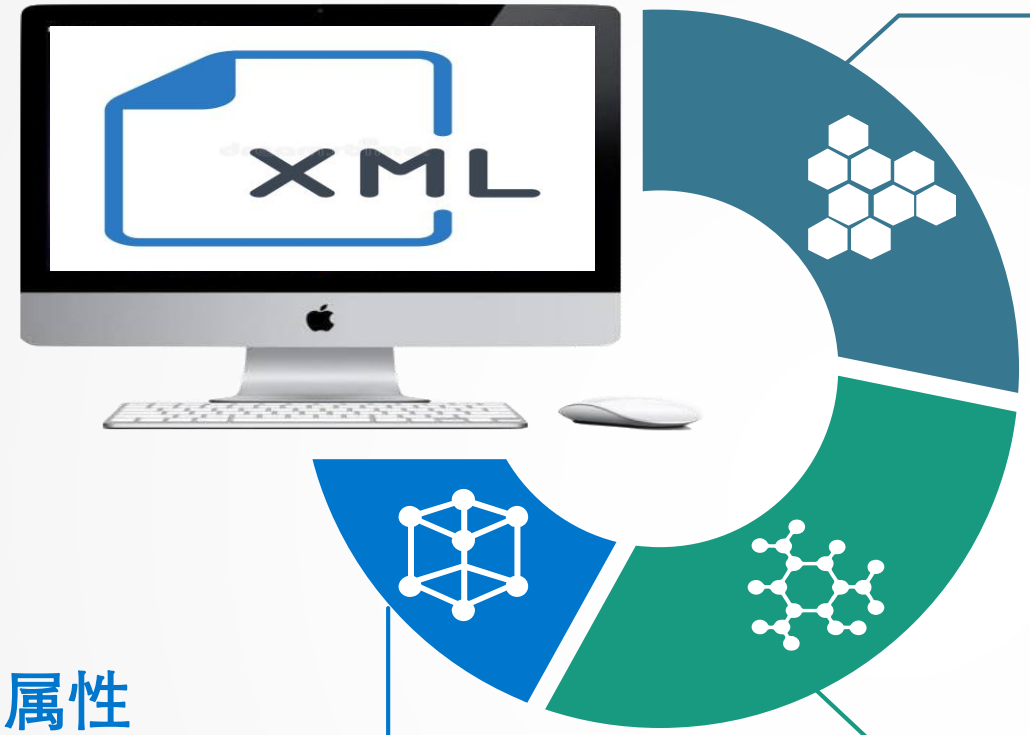
润乾软件出品



目 录

Contents

XML类型与特征



仅有元素内容

```
<?xml version="1.0" encoding="utf-8"?>
<BookStore>
  <Book>
    <title>Basic Mathematics</title>
    <author>Roy</author>
    <author>Jon</author>
    <copies>5</copies>
    <price>100</price>
  </Book>
</BookStore>
```

元素结构不同

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <item>
    <table1>
      <row>
        <column1>item 1 table 1 row 1 col 1</column1>
        <column2>item 1 table 1 row 1 col 2</column2>
      </row>
    </table1>
    <table2>
      <row>
        <columnX>item 1 table 2 row 1 col 1</columnX>
        <columnY>item 1 table 2 row 1 col 2</columnY>
        <columnZ>item 1 table 2 row 1 col 3</columnZ>
      </row>
    </table2>
  </item>
</list>
```

有元素有属性

```
<?xml version="1.0" encoding="utf-8"?>
<library>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author name="Giada De Laurentiis" country="it"/>
    <year>2005</year>
    <info>Hello Italian!</info>
  </book>
</library>
```

仅有元素内容

XML作为数据源是常见需求，但现有Java技术做起来比较麻烦：业务灵活性差、API接口烦而多、代码臃肿等。所以集算器提供了xml()函数，可以很方便的应对各类XML

Index	EID	NAME	SURNAME
1	1	Rebecca	Moore
2	2	Ashley	Wilson
3	3	Rachel	Johnson
4	4	Emily	Smith
5	5	Ashley	Smith

A1:查询数据表

	A
1	=demo.query("select * from EMPLOYEE")
2	=xml(A1)
3	=xml(A2)
4	=xml(A2,"xml/row")

A2:序表解析为xml串

Value
<?xml version="1.0" encoding="GBK"?> <xml> <row> <EID>1</EID> <N...

明细

```
<?xml version="1.0" encoding="GBK"?>
<xml>
  <row>
    <EID>1</EID>
    <NAME>Rebecca</NAME>
    <SURNAME>Moore</SURNAME>
  </row>
  <row>
    <EID>2</EID>
    <NAME>Ashley</NAME>
    <SURNAME>Wilson</SURNAME>
  </row>
  <row>
    <EID>3</EID>
    <NAME>Rachel</NAME>
    <SURNAME>Johnson</SURNAME>
  </row>
  <row>
    <EID>4</EID>
    <NAME>Emily</NAME>
    <SURNAME>Smith</SURNAME>
  </row>
  <row>
    <EID>5</EID>
    <NAME>Ashley</NAME>
    <SURNAME>Smith</SURNAME>
  </row>
</xml>
```

A3:xml串解析为序表
取根节点<xml>内容

xml
[[1,Rebecca,Moore],[2,Ashley,Wilson],[3,Rachel,Johnson], ...]

<xml>子节点为<row>

row
[[1,Rebecca,Moore],[2,Ashley,Wilson],[3,Rachel,Johnson], ...]

<row>节点下的元素值

Index	EID	NAME	SURNAME
1	1	Rebecca	Moore
2	2	Ashley	Wilson
3	3	Rachel	Johnson
4	4	Emily	Smith
5	5	Ashley	Smith

A4:直接取出<row>层的内容，返回序表

Index	EID	NAME	SURNAME
1	1	Rebecca	Moore
2	2	Ashley	Wilson
3	3	Rachel	Johnson
4	4	Emily	Smith
5	5	Ashley	Smith

仅有元素内容 – 多属性合并与格式化

- 多个Book组成BookStore list
- 每个Book会有多个author需合并为一列，copies可能存在非法字符需格式化为数值型

result

Index	title	author	copies	price
1	History	Tom	10	80
2	Basic Mathematics	Roy&Jon	5	100
3	Java	Harry&Potter	6	100

格式化为整型

Index	title	author	copies	price
1	History	Tom	10;	80
2	Basic Mathematics	[Roy,Jon]	5	100
3	Java	[Harry,Potter]	6	100

多个值用&连接

A2:Xml结构化后

	A	B
1	=file("/workspace/BookStore.xml")	/打开xml文件
2	=xml(A1.read(),"BookStore/Book")	/解析xml串为记录
3	=A2.new(title:title,if(ifa(author),author.concat("&"),author):author,if(ifstring(copies),int(replace(copies,";","")),copies):copies,price)	/生成序表

```
<?xml version="1.0" encoding="utf-8"?>
<BookStore>
  <Book>
    <title>History</title>
    <author>Tom</author>
    <copies>10;</copies>
    <price>80</price>
  </Book>
  <Book>
    <title>Basic Mathematics</title>
    <author>Roy</author>
    <author>Jon</author>
    <copies>5</copies>
    <price>100</price>
  </Book>
  <Book>
    <title>Java</title>
    <author>Harry</author>
    <author>Potter</author>
    <copies>6</copies>
    <price>100</price>
  </Book>
</BookStore>
```

有元素有属性

xml()函数带选项@s，可以将形如<K F=V F=V …>D</K>的XML串解析为以K,F,…为字段的记录，K取值为D，D是多层XML内容时解析为排列，<K …./K>时D解析为null，<K…></K>时D解析为空串

Index	bookstore
1	[[[Harry Potter,en],[J K. Rowling,it],[2005],…],[[Learning XML,en],[Eri…

A2:解析为多层序列

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author country="it">J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author country="uk">Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

子节点为<book>,属性:<category>

Index	book	category
1	[[Harry Potter,en],[J K. Rowli…	CHILDREN
2	[[Learning XML,en],[Erik T. …	WEB

<book>节点下元素与属性

Index	Member
1	[Harry Potter,en]
2	[J K. Rowling,it]
3	[2005]
4	[29.99]

title	lang
Harry Potter	en

author	country
J K. Rowling	it

year
2005

price
29.99

A1:读文件解析为xml串

Value
<?xml version="1.0" encoding="utf-8"?> <bookstore> <book category="…

Index	category	title	lang	author	country	year	price
1	CHILDREN	Harry Potter	en	J K. Rowling	it	2005	29.99
2	WEB	Learning XML	en	Erik T. Ray	uk	2003	39.95

A4:生成新序表，取出相应的元素值与属性值

	A
1	=file("/root/workspace/book.xml").read()
2	=xml@s(A1)
3	=xml@s(A1).bookstore
4	=A3.new(category,book(1).title:title,...)

有元素有属性 – 对位合并与过滤

- 每个book会有多个author，其属性name和country需对位合并在一列
- 结构化后可过滤、分组

1、结构化xml，其中author为list，属性name与country的值需分别用逗号连接

result1

Index	category	year	title	lang	info	name	country
1	COOKING	2005	Everyday Italian	en	Hello Italian!	Giada De Laurentiis	it
2	CHILDREN	2006	Harry Potter	en	Hello Potter!	J K. Rowling	uk
3	WEB	2005	XQuery Kick Start	en	Hello XQue...	James McGovern,Per Bothner	us,us
4	WEB	2003	Learning XML	en	Hello XML!	Erik T. Ray	us

2、结构化xml，其中author为list，属性name与country的值需对位合并为一列

result2

Index	title	category	year	author	info
1	Everyday Italian	COOKING	2005	Giada De Laurentiis[it]	Hello Italian!
2	Harry Potter	CHILDREN	2006	J K. Rowling[uk]	Hello Potter!
3	XQuery Kick Start	WEB	2005	James McGovern[us],Per Bothner[us]	Hello XQuery!
4	Learning XML	WEB	2003	Erik T. Ray[us]	Hello XML!

3、在2的基础上，只查询2005年的书籍信息

result3

Index	title	category	year	author	info
1	Everyday Italian	COOKING	2005	Giada De Laurentiis[it]	Hello Italian!
2	XQuery Kick Start	WEB	2005	James McGovern[us],Per Bothner[us]	Hello XQuery!

```xml
<?xml version="1.0"?>
<library>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author name="Giada De Laurentiis" country="it"/>
        <year>2005</year>
        <info>Hello Italian!</info>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author name="J K. Rowling" country="uk"/>
        <year>2006</year>
        <info>Hello Potter!</info>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author name="James McGovern" country="us"/>
        <author name="Per Bothner" country="us"/>
        <year>2005</year>
        <info>Hello XQuery!</info>
    </book>
    <book category="WEB">
        <title lang="en">Learning XML</title>
        <author name="Erik T. Ray" country="us"/>
        <year>2003</year>
        <info>Hello XML!</info>
    </book>
</library>
```

有元素有属性 – 对位合并与过滤(示例)

集算器可直接解析并计算XML，其敏捷语法体系仅需很少代码即可完成上述需求。

	A	B
1	=file("/workspace/book1.xml")	/打开xml文件
2	=xml@s(A1.read(),"library/book").library	/解析xml串为字段的记录，并获取节点值
3	=A2.new(category,book.field("year").ifn():year,book.field("title").ifn():title,book.field("lang").ifn():lang,book.field("info").ifn():info,book.field("name").select(~).concat@c():name,book.field("country").select(~).concat(","):country)	/生成新序表，获取序列中每个字段的字段值并做非空判断，其中list需连接为字符串
4	=A3.new(title,category,year,(lang,name.array().(~+"[")++country.array().(~+"]")).concat@c():author,info)	/生成新序表，其中list列需做对位相加后再连接为字符串
5	=A4.select(year==2005)	/在A4基础上，按条件过滤

A2：分解过程

Index	book	category
1	[[Everyday Italian,en],[,Giada De Laur...	COOKING
2	[[Harry Potter,en],[,J K. Rowling,uk],[2...	CHILDREN
3	[[XQuery Kick Start,en],[,James McGov...	WEB
4	[[Learning XML,en],[,Erik T. Ray,us],[2...	WEB

Index	Member
1	[Everyday Italian,en]
2	[,Giada De Laurentiis,it]
3	[2005]
4	[Hello Italian!]

title	lang
Everyday Italian	en

author	name	country
(null)	Giada De Laurentiis	it

year
2005

info
Hello Italian!

元素结构不同

xml(x,s)函数，其中s表示要取出的层标识，多层用/分隔，空表示从根开始取，对节点下有不同结构的元素时，可以利用s精准获取某层元素

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <book>
    <title>The Spanish Cook Book</title>
    <author>Miguel Ortiz</author>
    <date>2005-11-23</date>
  </book>
  <audio>
    <title>The Bluest Blues</title>
    <artist>Barry Sadley</artist>
    <year>2006</year>
    <price>39.95</price>
  </audio>
</list>
```

title	author	date
The Spanish Cook Book	Miguel Ortiz	2005-11-23

A2:取出<book>层的内容，返回序表

title	artist	year	price
The Bluest Blues	Barry Sadley	2006	39.95

A3:取出<audio>层的内容，返回序表

Value
<?xml version="1.0" encoding="utf-8"?> <list> <book> <title>The Spanish Cook Book</title> ...

A1:读文件解析为xml串

	A
1	=file("/root/workspace/book.xml").read()
2	=xml(A1,"list/book")
3	=xml(A1,"list/audio")

元素结构不同 – 子节点包含不同元素

- 多个item组成list
- 每个item有固定量的tables，且table不同，每个table有不固定量的rows

table1

ItemID	column1	column2
1	item 1 table 1 row 1 col 1	item 1 table 1 row 1 col 2
1	item 1 table 1 row 2 col 1	item 1 table 1 row 2 col 2
2	item 2 table 1 row 1 col 1	item 2 table 1 row 1 col 2
2	item 2 table 1 row 2 col 1	item 2 table 1 row 2 col 2

table2

ItemID	columnX	columnY	columnZ
1	item 1 table 2 row 1 col 1	item 1 table 2 row 1 col 2	item 1 table 2 row 1 col 3
1	item 1 table 2 row 2 col 1	item 1 table 2 row 2 col 2	item 1 table 2 row 2 col 3
2	item 2 table 2 row 1 col 1	item 2 table 2 row 1 col 2	item 2 table 2 row 1 col 3
2	item 2 table 2 row 2 col 1	item 2 table 2 row 2 col 2	item 2 table 2 row 2 col 3

```xml
<?xml version="1.0" encoding="utf-8"?>
<list>
    <item>
        <table1>
            <row>
                <column1>item 1 table 1 row 1 col 1</column1>
                <column2>item 1 table 1 row 1 col 2</column2>
            </row>
            <row>
                <column1>item 1 table 1 row 2 col 1</column1>
                <column2>item 1 table 1 row 2 col 2</column2>
            </row>
        </table1>
        <table2>
            <row>
                <columnX>item 1 table 2 row 1 col 1</columnX>
                <columnY>item 1 table 2 row 1 col 2</columnY>
                <columnZ>item 1 table 2 row 1 col 3</columnZ>
            </row>
            <row>
                <columnX>item 1 table 2 row 2 col 1</columnX>
                <columnY>item 1 table 2 row 2 col 2</columnY>
                <columnZ>item 1 table 2 row 2 col 3</columnZ>
            </row>
        </table2>
    </item>
    <item>
    </item>
</list>
```

List...

元素结构不同 – 子节点包含不同元素(示例)

在函数中，指定层级标识，可精准获取此层元素值

ItemID	row
1	[[item 1 table 1 row 1 col 1,item 1 table 1 row 1 c...
2	[[item 2 table 1 row 1 col 1,item 2 table 1 row 1 c...

ItemID	column1	column2
1	item 1 table 1 row 1 col 1	item 1 table 1 row 1 col 2
1	item 1 table 1 row 2 col 1	item 1 table 1 row 2 col 2
2	item 2 table 1 row 1 col 1	item 2 table 1 row 1 col 2
2	item 2 table 1 row 2 col 1	item 2 table 1 row 2 col 2

A3:分解过程

	A	B
1	=file("/workspace/items.xml")	/打开xml文件
2	=xml(A1.read(),"list/item/table1")	/解析xml串为字段的记录，并获取节点值
3	=A2.new(#:ItemID,row)	/生成序号，节点值集合
4	=A3.news(row;ItemID,row.column1:column1,row.column2:column2)	/展开集合，生成新序表
5	=xml(A1.read(),"list/item/table2")	/解析xml串为字段的记录，并获取节点值
6	=A5.new(#:ItemID,row)	/生成序号，节点值集合
7	=A6.news(row;ItemID,row.columnX:columnX,row.columnY:columnY,row.columnZ:columnZ)	/展开集合，生成新序表

A6:分解过程

ItemID	row
1	[[item 1 table 2 row 1 col 1,item 1 table 2 row 1 col 2,ite...
2	[[item 2 table 2 row 1 col 1,item 2 table 2 row 1 col 2,ite...

ItemID	columnX	columnY	columnZ
1	item 1 table 2 row 1 col 1	item 1 table 2 row 1 col 2	item 1 table 2 row 1 col 3
1	item 1 table 2 row 2 col 1	item 1 table 2 row 2 col 2	item 1 table 2 row 2 col 3
2	item 2 table 2 row 1 col 1	item 2 table 2 row 1 col 2	item 2 table 2 row 1 col 3
2	item 2 table 2 row 2 col 1	item 2 table 2 row 2 col 2	item 2 table 2 row 2 col 3

综合应用举例 – 数据库与XML关联

cities表来自Mysql数据库，state数据来自xml文件，关联计算后，分组统计每个州的人口数

cities

Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	2	Los Angeles	3798981.0	5
3	3	Chicago	2886251.0	1
4	4	Houston	2009834.0	1
5	5	Philadelphia	1492231.0	2
6	6	Phoenix	1371960.0	1

JOIN

result

Index	STATE	POPULATION
1	Alabama	6268045.0
2	Alaska	9576547.0

state.xml

```
<?xml version="1.0" encoding="utf-8"?>
<data>
  <state>
    <STATEID>1</STATEID>
    <NAME>"Alabama"</NAME>
    <ABBR>"AL"</ABBR>
  </state>
  <state>
    <STATEID>2</STATEID>
    <NAME>"Alaska"</NAME>
    <ABBR>"AK"</ABBR>
  </state>
</data>
```

综合应用举例 – 数据库与XML关联(示例)

集算器可直接读取XML与Mysql数据进行混合计算；提供了一致的计算接口，各种数据源都可以用统一的风格来计算

	A	B
1	=Mysql.query("select * from cities where STATEID<=2")	/查询cities表
2	=xml(file("/workspace/state.xml").read(),"data/state")	/解析xml串为字段的记录，并获取节点值
3	=A2.new(STATEID,NAME,ABBR).keys(STATEID)	/生成序表，并设定主键
4	>A1.switch(STATEID,A3:STATEID)	/cities与state关联
5	=A1.groups(STATEID.NAME:STATE;sum(POPULATION):POPULATION)	/分组汇总

A3:Xml结构化后

Index	STATEID	NAME	ABBR
1	1	Alabama	AL
2	2	Alaska	AK

Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	3	Chicago	2886251.0	1
3	4	Houston	2009834.0	1
4	5	Philadelphia	1492231.0	2
5	6	Phoenix	1371960.0	1

A4:执行关联后

Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	3	Chicago	2886251.0	1
3	4	Houston	2009834.0	1
4	5	Philadelphia	1492231.0	2
5	6	Phoenix	1371960.0	1

STATEID	NAME	ABBR
2	Alaska	AK

STATEID	NAME	ABBR
1	Alabama	AL

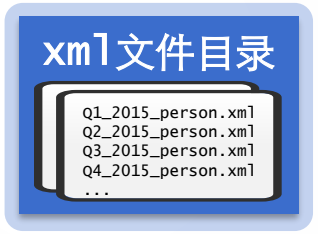
综合应用举例 – 批量解析

- 目录下有多个XML文件，每个XML具有相同的结构
- 批量解析并结构化

result

Index	Overlay_Filen...	Overlay_Name	Overlay_X...	Overlay_Y...	Text_Top	Text_Left	Text_Bott...	Text_Right	Vector_Gr...	CanRotate...	CanRotate...	CanRotate...	CanFlip_X	CanFlip_X...	CanFlip_X...
1	arrow_1.png	Arrow	100	100	40	10	40	25	0	1	1	1	0	0	0
2	arrow_2.png	2-Sided Arrow	100	100	45	25	45	25	0	1	0	0	0	0	0

Index	Overlay
1	[arrow_1.png,Arrow,100, ...]
2	[arrow_2.png,2-Sided Arrow,100, ...]



Index	Member
1	/root/workspace/xml/sample/n1.xml
2	/root/workspace/xml/sample/n2.xml

A2:分别读取每个xml

A1:当前目录下的xml

person.xml

```
<?xml version="1.0"?>
<Overlay>
    <Overlay_Filename>arrow_1.png</Overlay_Filename>
    <Overlay_Name>Arrow</Overlay_Name>
    <Overlay_XGrow>100</Overlay_XGrow>
    <Overlay_YGrow>100</Overlay_YGrow>
    <Text_Top>40</Text_Top>
    <Text_Left>10</Text_Left>
    <Text_Bottom>40</Text_Bottom>
    <Text_Right>25</Text_Right>
    <Vector_Grow>0</Vector_Grow>
    <CanRotate_90>1</CanRotate_90>
    <CanRotate_180>1</CanRotate_180>
    <CanRotate_270>1</CanRotate_270>
    <CanFlip_X>0</CanFlip_X>
    <CanFlip_X90>0</CanFlip_X90>
    <CanFlip_X180>0</CanFlip_X180>
    <CanFlip_X270>0</CanFlip_X270>
    <Fill_Color>1</Fill_Color>
    <Border_Color>1</Border_Color>
</Overlay>
```

	A	B
1	=directory@p("/workspace/tmp/*.xml")	/列出满足通配符路径的文件名
2	=A1.(xml(file(~).read()))	/分别解析每个xml串为记录
3	=A2.conj(~.array())	/将每个序列合并

综合应用举例 – 结构化webService

- 根据传入参数，调用外部WebService，返回该地区的天气情况
- xml结果集结构化

result

Index	str1	str2	str3	str4	str5	str6	str7	str8	str9
1	Henan (Province).	Xinyang(City).	464000	57297.jpg	2019/12/20 16:00:31	0°C/10°C	December 20 is cloudy to overcast	East to north is less than category 3	.....

"http://www.webxml.com.cn/WebServices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=%E4%BF%A1%E9%98%B3":"UTF-8"

weather.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSche
  <string>Henan (Province).</string>
  <string>Xinyang(City).</string>
  <string>464000</string>
  <string>57297.jpg</string>
  <string>2019/12/20 16:00:31</string>
  <string>0°C/10°C</string>
  <string>December 20 is cloudy to overcast</string>
  <string>East to north is less than category 3</string>
  <string>......</string>
</ArrayOfString>
```

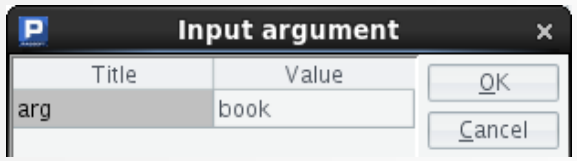
List...

argCity=信阳，A1:拼接的url

	A	B
1	=wsdl=concat("\"http://www.webxml.com.cn/WebServices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=",urlencode(argCity,"UTF-8"),"\":\"UTF-8\"")	/结合传入参数argCity,拼出完整wsdl url
2	=httpfile(${wsdl})	/将url结果转为文件流
3	=xml(file(A2).read(),"ArrayOfString/string")	/解析xml
4	=create(${A1.(concat("str",#)).concat@c()})	/创建空序表
5	>A4.record(A3)	/记录填入序表

综合应用举例 – 根据参数获取不同数据

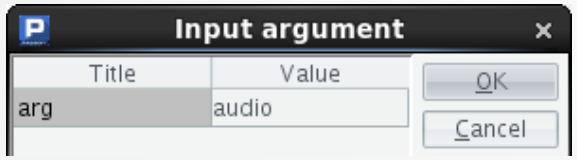
- 一个XML包含多种标签结构，每种结构具有同等列数的标签属性
- 根据参数不同，获取相应的数据呈现不同的报表



arg=book，提取标签为book的数据

report1

Index	category	title	lang	name	country	year
1	COOKING	The Spanish Cook Book	es	Miguel Ortiz	es	2005
2	CHILDREN	Everyone is Super Special	en	Sally Bush	us	2005



arg=audio，提取标签为book的数据

report2

Index	category	title	lang	name	country	year
1	MUSIC	We All Sing Perty	en	Mary Rogers	us	2006
2	MUSIC	The Bluest Blues	en	Barry Sadley	us	2006

big.xml

```
<?xml version="1.0" encoding="utf-8"?>
<library>
  <book category="COOKING">
    <title lang="es">The Spanish Cook Book</title>
    <author name="Miguel Ortiz" country="es"/>
    <year>2005</year>
  </book>
  <book category="CHILDREN">
    <title lang="en">Everyone is Super Special</title>
    <author name="Sally Bush" country="us"/>
    <year>2005</year>
  </book>
  <audio format="CD" category="MUSIC">
    <title lang="en">We All Sing Perty</title>
    <artist name="Mary Rogers" country="us"/>
    <year>2006</year>
  </audio>
  <audio format="CD" category="MUSIC">
    <title lang="en">The Bluest Blues</title>
    <artist name="Barry Sadley" country="us"/>
    <year>2006</year>
  </audio>
</library>
```

List...

综合应用举例–根据参数获取不同数据(示例)

利用集算器解析XML后，其敏捷语法体系仅需很少代码就能完成逻辑判断，特有的宏机制极大地提高了代码复用程度

	A	B	C
1	=file("/workspace/big.xml")		/打开xml文件
2	=xml@s(A1.read())	=${arg}=null	/A2:解析xml,B2:定义宏的变量arg,默认为null
3	=A2.library		/获取library节点值
4	for A3	if(A4.fname(1)==arg)	/A4:循环节点,B4:根据参数,判断第一个字段名
5			=${arg}=if(${arg}==null,create(category,${A4.${arg}.conj(~.fname()).concat@c()}).record(A4.${arg}.conj(~.array()).insert(1,A4.category)),${arg}.record(A4.${arg}.conj(~.array()).insert(1,A4.category)))
6	=${arg}=${arg}.new(category,title,lang,name,country,year)		

A3:获取library节点值为序列

Index	Member
1	[[The Spanish Cook Book,es],[,Miguel Ortiz,es],[2005], ...]
2	[[Everyone is Super Special,en],[,Sally Bush,us],[2005], ...]
3	[[We All Sing Perty,en],[,Mary Rogers,us],[2006], ...]
4	[[The Bluest Blues,en],[,Barry Sadley,us],[2006], ...]

C5:首次循环变量为空时，创建空序表所包含的列，然后向空序表里插入一条记录，之后依次插入直到结束

Index	category	title	lang	author	name	country	year
1	COOKING	The Spanish Cook Book	es	(null)	Miguel Ortiz	es	2005
2	CHILDREN	Everyone is Super Special	en	(null)	Sally Bush	us	2005

A6:生成新序表，返回报表需要的通用列

Index	category	title	lang	name	country	year
1	COOKING	The Spanish Cook Book	es	Miguel Ortiz	es	2005
2	CHILDREN	Everyone is Super Special	en	Sally Bush	us	2005

创新技术

推动应用进步！


各设备故障分析
1.53%
0.95%
2.40%
全年故障综合分析
近7天故障情况
近7天故障次数前5的设备
服务器CPU使用率
今日
昨日
今天硬盘情况