Creador OPEN: sParser2 Component

With this document you'll learn:

- ✓ What is sParser component
- ✓ How to initialize
- ✓ Filters and Selectors
- ✓ Functions for Selectors
- ✓ Examples

What is sParser2 component ?

Its a Coldfusion component, fork of the sParser component of Creador OPEN framework, that allows to extract information from texts and website sources, similar to jQuery Selectors used with Javascript. The only difference from the original sParser is that this one is Adobe Coldfusion compatible and its internal fields are created with a "i_" prefix.

How to initialize it

To use it in a cfm page, you must init the component as follows:

```
<cfset object=CreateObject("component","sparser2")>
<cfset parser=object.init("http://www.yourwebsite.com")>
```

With that, you are ready to start using selectors and filters.

Selectors and Filters

Selectors allow you to "select" in a fast and elegant way, elements or groups of elements from a source text (which can be either xml or html based), and extract or modify them using methods and functions.

You use selectors with the command:
parser.**selector("selector statement")**, and can use it as follows:

selector("*")
Selects all tags of the source content.

selector("tag")
Select all instances of the specified tag, of the source content.

selector(".class")
Select all tags that have the given CSS class.

selector("#id")
Select all tags that have the given ID attribute.

selector("tag.class")
Select all instances of the specified tag, that have the given CSS class.

selector("tag#id")
Select all instances of the specified tag, that have the given ID attribute.

selector("tag#id.class")
Select all instances of the specified tag, that have the given ID attribute and the given CSS class.

selector(":filter")
Applies the given filter to all tags in the source content

Filters

The filters are a way to refine a selector statement, and can "filter" its results.

selector("tag:even")
Selects all even instances of the specified tag (or statement).

selector("tag:odd")
Selects all odd instances of the specified tag (or statement).

selector("tag:eq(n)")
Selects the **nth** position of the specified tag (or statement).

selector("tag:neq(n)")
Selects all instances of the specfied tag except the one at the **nth** position.

selector("tag:gt(n)")
Selects all instances of the specified tag (or statement) that are after the **nth** position defined.

selector("tag:lt(n)")
Selects all instances of the specified tag (or statement) that are before the **nth** position defined.

selector("tag:not(selector statement)")
Selects all instances of the specified tag that do not match the given statement.

selector("tag:first")
Selects the first instance of the specified tag.

selector("tag:first-child")
Selects the first child of each instance of the specified tag.

selector("tag:contains(x)")
Selects all instances of the specified tag that contain the text in **x**.

selector("tag:header")
Selects all instances of the specified tag that are header tags.

selector("tag:button")
Selects all instances of the specified tag that are button tags.

selector("tag:checkbox")
Selects all instances of the specified tag that are of type checkbox.

selector("tag:checked")
Selects all instances of the specified tag that are checked.

selector("tag:selected")
Selects all instances of the specified tag that are selected.

selector("tag:disabled")
Selects all instances of the specified tag that are disabled.

selector("tag:enabled")
Selects all instances of the specified tag that are enabled.

selector("tag:empty")
Selects all instances of the specified tag that contain no child elements.

selector("tag:input")
Select all instances of the specified tag that are of type input (input, select, textarea, button).

selector("tag:last-child")
Selects the last child of each instance of the specified tag or statement.

selector("tag:parent")
Selects all instances of the specified tag that are the parent of another element.

selector("tag:only-child")
Selects all instances of the specified tag that are the only child of their parents.

Attribute Filters

Aditionally you can specify attribute filters:

selector("tag[attribute]")
Selects all instances of the specified tag that have the specified attribute.

selector("tag[attribute=value]")
Selects all instances of the specified tag that contain the given attribute with its value equal to the given string.

selector("tag[attribute!=value]")
Selects all instances of the specified tag that contain the given attribute with its value different than the given string.

selector("tag[attribute^=value]")
Selects all instances of the specified tag that contain the given attribute with its value starting with the given string.

selector("tag[attribute$=value]")
Selects all instances of the specified tag that contain the given attribute with its value ending with the given string.

selector("tag[attribute*=value]")
Selects all instances of the specified tag that contain the given attribute with its value containing the given string.

selector("tag[attribute~=value]")
Selects all instances of the specified tag that contain the given attribute with its value containing the given string, delimited with spaces.

selector("tag[attribute|=value]")
Selects all instances of the specified tag that contain the given attribute with its value containing the given string, or starting with the given value followed by a dash.

selector("tag[filter attribute1] [filter attribute2]")
You can specify more than one attribute filter; all must be true for the tag to be included.

Also you can specify cascaded statements.

selector("grandparent_selector parent_selector child_selector")
Selects all tags that match the last child_selector that are inside the parent_selector, and also inside the grandparent_selector items.

Functions for selectors

The resulting items of a selector can be extracted and modified by using the following functions and methods:

selector("x").**query()**
Returns an SQL type query with each tag attribute as a column fieldname, plus the following:

- **i_tag_name**
  Name of the found tag.
- **i_content**
  Contents of the tag instance.
- **i_depth**
  Depth level for the tag's instance.
- **i_begin**
  Char position to tag instance from begining of source content.
- **i_end**
  Char position until end of tag instance from begining of source content.

selector("x").**query(x)**
Returns a query variable like query(), but filtered with an aditional SQL query defined in x. Helper method to avoid using another query of queries.

selector("x").**parent()**
Returns the parent of every found tag.

selector("x").**parent(x)**
Returns the parent of every found tag that also matches the selector statement defined in **x**.

selector("x").**parents()**
Returns all the parent tags of each found tag.

selector("x").**parents(x)**
Returns the parent tags of each found tag that also matches the selector statement defined in **x**.

selector("x").**closest(x)**
Returns the closest tags defined in selector **x** to the found tags of the main selector.

selector("x").**filter(":filter")**
Aplies the given filter to the found tags.

selector("x").**html()**
Returns the html code from the first found tag instance.

selector("x").**html(x)**
Replaces the html code of all found tag instances with the given value in **x**.

selector("x").**replaceWith(x)**
Replaces the contents of all found tag instances with the given value in **x**.

selector("x").**text()**
Returns the combined text of all found tag instances, including their children.

selector("x").**text(x)**
Replaces the text contents of all found tag instances, including their children.

Addionally...

The following method can be applied to the parser object:

parser.**getTitle()**
Returns the page title (only if source is html).

parser.**getEncoding()**
Returns the page charset encoding.

parser.**format()**
Indents the code of the source.

parser.**getContent()**
Returns the current content of the parser. If it has being modified by another method, this way you'll get it modified.

Examples

The following are some selector statement examples:

selector("img[src$=.png]")
Returns all images ending with extension .png.

selector("p.red:even")
Returns all even p tags of class red.

Complete example:

Get the current value of "UF" from the Central Bank Of Chile.

Description:
Returns the text content of the second td tag that is inside the tr tag of class odd, which is in turn is inside tag div of id "ind-dia" (in Coldfusion you must escape the reserved hash # character, doubling it, because it is used for identifing variables).

In code:

```
<cfset object=CreateObject("component","sparser2")>
<cfset parser=object.init("http://www.bcentral.cl/index.asp")>
<cfset uf=parser.selector("div##ind-dia tr.odd td:eq(2)").text()>
<cfoutput>The current UF value of today is: #uf#</cfoutput>
```