

# MyCocktail

## Romulus Mashup Builder



# MyCocktail Handbook

<http://www.ict-romulus.eu/web/mycocktail>

Author: Informática Gesfor

MyCocktail is part of ICT Romulus Project <http://www.ict-romulus.eu>

## Table of Contents

<b>1</b>	<b>INSTALLATION.....</b>	<b>3</b>
1.1	Requirements.....	3
1.2	Deploy the WAR.....	3
1.3	Configure MyCocktail.....	3
1.4	Database.....	3
1.5	Run MyCocktail.....	4
<b>2</b>	<b>MYCOCKTAIL MASHUP BUILDER.....</b>	<b>5</b>
2.1	General Overview of the Mashup Builder.....	5
2.2	Action panel.....	7
2.2.1	Input parameters.....	9
2.2.2	Services.....	10
2.2.2.1	Importation of a service from WADL.....	15
2.2.3	Operators.....	18
2.2.4	Renderers.....	27
<b>3</b>	<b>PAGE EDITOR.....</b>	<b>38</b>
3.1	Toolbars.....	39
3.1.1	FCKEditor Toolbar.....	39
3.1.2	Page Editor Toolbar.....	39
3.2	Design area.....	42
<b>4</b>	<b>WEBSITE AND ONLINE DEMO.....</b>	<b>43</b>
<b>5</b>	<b>REFERENCES.....</b>	<b>43</b>

# 1 Installation

## 1.1 Requirements

MyCocktail requires:

- Servlet container (Apache Tomcat, Glassfish, JBoss, etc.)
  - Compatible Web browser: Mozilla Firefox, Opera, Chrome, or Safari. Internet Explorer (not fully compatible).
  - Flash plug-in installed in the web browser.

## 1.2 Deploy the WAR

The MyCocktail distribution contains the WAR MyCocktail.war, it should be deployed in a Servlet Container.

## 1.3 Configure MyCocktail

Once the project is deployed, the folder which are contained in the WAR are unzipped. We have to enter in the deployed folder and open the file in "afrous-config.js" in folder "js/afrous" has to be configured.

```
var afrous;
if (!afrous) afrous = {};
(function() {
afrous.packages = {};

/***** SERVER CONFIGURATION *****/
afrous.baseURL = "http://<DOMAIN>[:<PORT>]/MyCocktail"
afrous.packages.scriptBaseURL = afrous.baseURL+"/js/afrous";
/***** SERVER CONFIGURATION *****/

})();
```

<DOMAIN> has to be substituted by the domain where is published de application and <PORT> by the port if it is necessary. For instance: <http://www.ict-romulus.eu/MyCocktail> or <http://localhost:8080/MyCocktail>.

## 1.4 Database

Enter in the "database" folder (in MyCocktail distribution) and launch this command:

./start-db.sh (Linux or MAC) start-db.bat (Windows)

To stop the database enter in the "database" folder and launch this command:

./stop-db.sh (Linux or MAC) stop-db.bat (Windows)

## **1.5 Run MyCocktail**

MyCocktail is published by the servlet container where you have made the deploy usually you can access to it these URLs:

- MyCocktail Mashup Builder: <http://<DOMAIN>/MyCocktail>
- MyCocktail Page Editor: <http://<DOMAIN>/MyCocktail/editor.html>

## 2 MyCocktail Mashup Builder

### 2.1 General Overview of the Mashup Builder

The main window of the application is shown in Illustration 1.

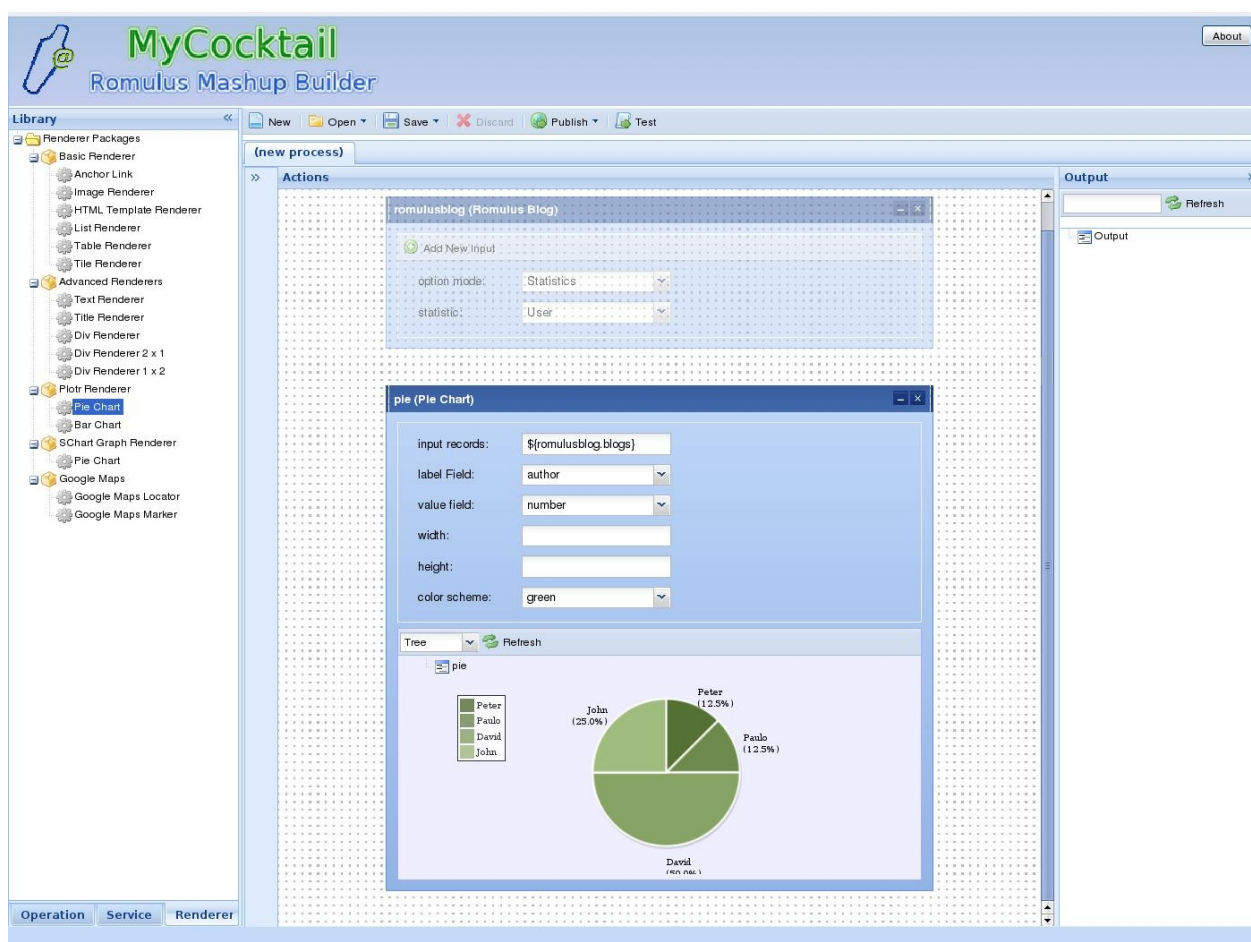
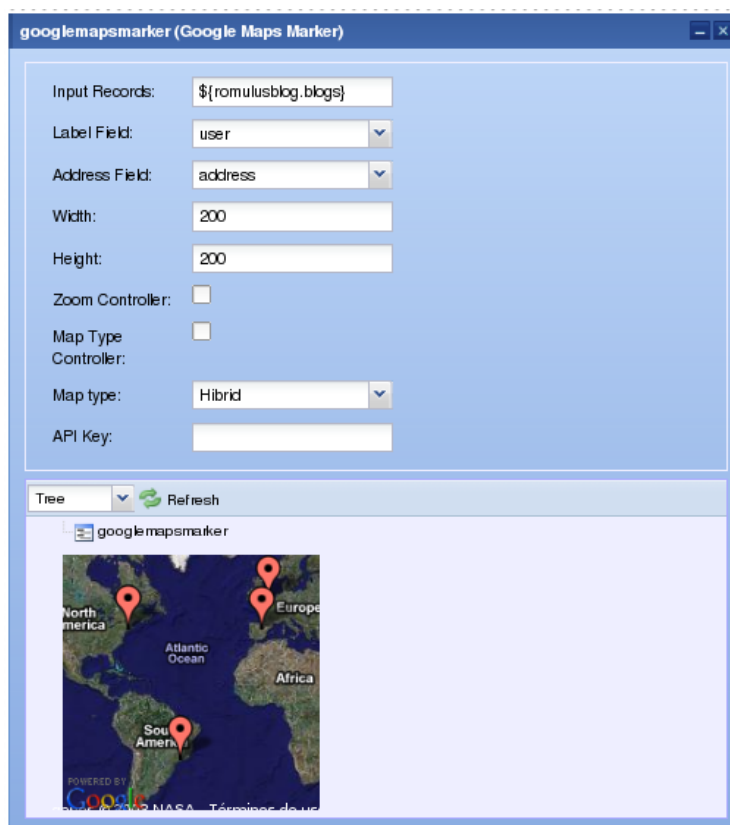


Illustration 1: Graphic user interface of the application

The interface is divided in three parts:

- **Library panel:** in this panel appears the operators, services and renderers that can be used to build the mashup:
  - **Operators:** The information obtained can be manipulated with the operators. For example, it's possible to sort, filter or group the information by the parameter chosen.
  - **Service:** Several RESTful services that can be invoked: del.icio.us (tags, posts, etc.), Yahoo Web Search, Google AJAX Search, Flickr Public Photo Feed, Twitter, Amazon, etc.
  - **Renderers:** the information can be represented in different renders:
    - **HTML renderers:** this components generate HTML elements (image, link or

- table tags). One renderer can be displayed into others, the output of a render can be used like input of another render.
- **Statistic renderers:** two kinds of statics diagram can be generated: pie char and bar char.
  - **Google Maps renderers:** they geolocalise the information in a map.
  - **Smile renderers:** integration of Smile Widgets [SmiURL] into MyCocktail, currently the timeline widget [TimURL].
  - **Widget renderers:** this renderers insert Flash components to represent the information.
- **Actions panel:** the operators can be drag and drop to it and is where they are combined in order to collaborate to form a mashup. All the operators, services and renderers are represented in the panel as a form with some fields, this is the input of the operator. Submitting the form, the result of the operator is shown in the below part of the form.



*Illustration 2: Dialog with the input fields and output representation of the Google Maps Renderer*

- **Output panel:** the mashup should be dragged and dropped to this panel to be exported. An important capability is that program allows to represents more than one mashup in a web page. It is possible with the div renderer that divides the page in different areas and the mashups can be put in this areas.

## 2.2 Action panel

This is the panel where the operators, services and renderers can be dragged and dropped to it. All of them have the same structure when are dropped to this panel. They are represented in the panel as a form with some fields, this is the input of the operator. Submitting the form, the result of the operator is shown in the below part of the form (output displayer).

In the next figure is shown the form that the user has to complete to display the data into a map of Google Maps.

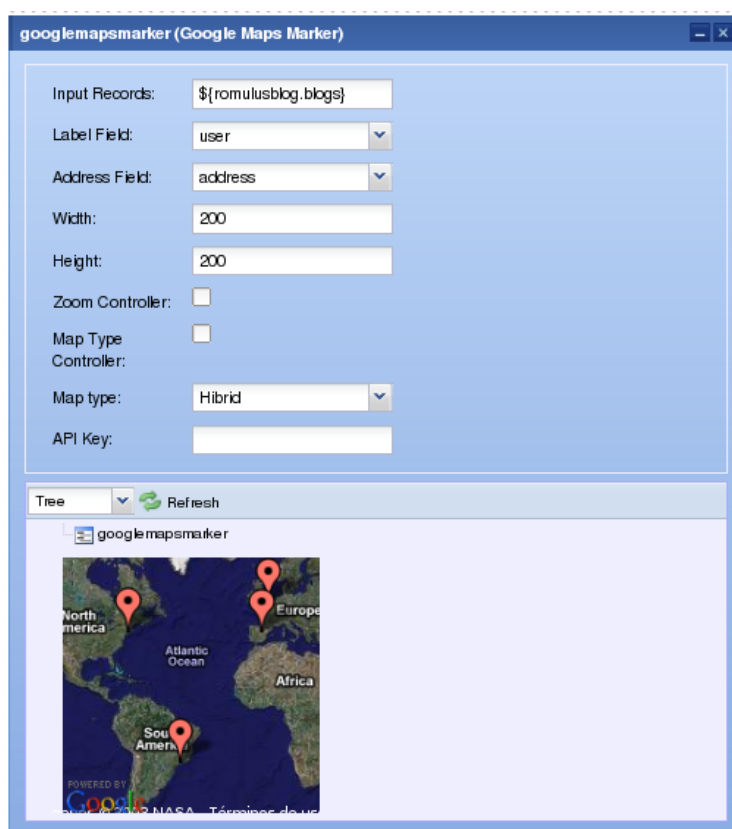


Illustration 3: Panel configuration of the Google Maps renderer

The dialog panel also has a element that represent the output, in this case is called "googlemapsmarker". This output element has the property that can be dragged and dropped in any input field of other operator, service or renderer depending of the case.

The output of this element is combined with the input of other ones and thus the different elements pass the information to one to another making possible to build the mashup.

This panel has a tool bar, which performs actions over the process. A process is the configuration of a mashup in which is developed the flow of the mashup from the data information providers (services) to the renders.

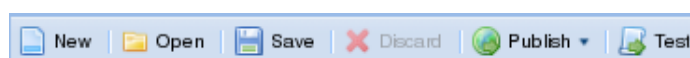
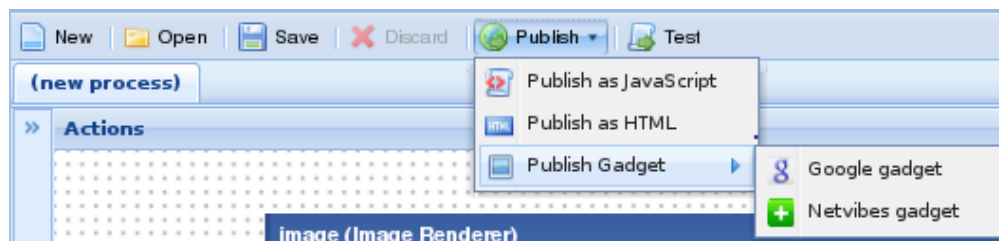


Illustration 4: Tool bar of the Action panel.

The tool bar has the following buttons:

- New: pressing this button a new process is created.
- Open: this option opens a process from local disk.
- Save: for saving a process to local disk.

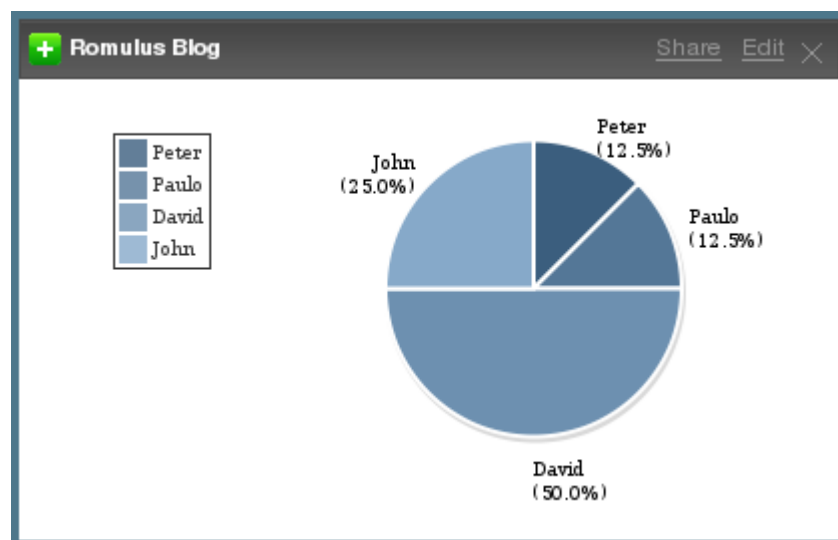


*Illustration 5: Publish menu*

- Publish: in this menu appears the exportation next exportation options.
  - Publish as JavaScript
  - Publish as HTML
  - Publish as Google Gadget
  - Publish as Netvibes Gadget

The tool provides exportation to Google Gadget and Netvibes Gadget, the full code needed for the mashup is provided by the application.

This is the result when the mashup is displayed in Netvibes Web Page.



*Illustration 6: A mashup developed with the application in Netvibes site*

- Test: the mashup is directly tested in a external window of the web browser when the user press this button.



## 2.2.1 Input parameters

MyCocktail allows to take input parameters to interact with the mashup, these parameters will be retrieved from URL when the mashup is exported. This can be useful if the user want to change the data source of the mashups without change the way in which the mashup is displayed. For instance, these parameter can be used to filter the data source, to make a search or to configure some aspect of the mashup.

Below the toolbar is the name of the actual process, if it is not saved the name is “(new process)” and below there is a panel with the list of input parameters defined in the process. The input parameters are parameters which can be used in the forms of the operators and when the mashup is exported this parameters are retrieved from the URL of the page in which they are inserted. The addition of a parameter is possible through the button “Add New Param” (see Illustration 7).

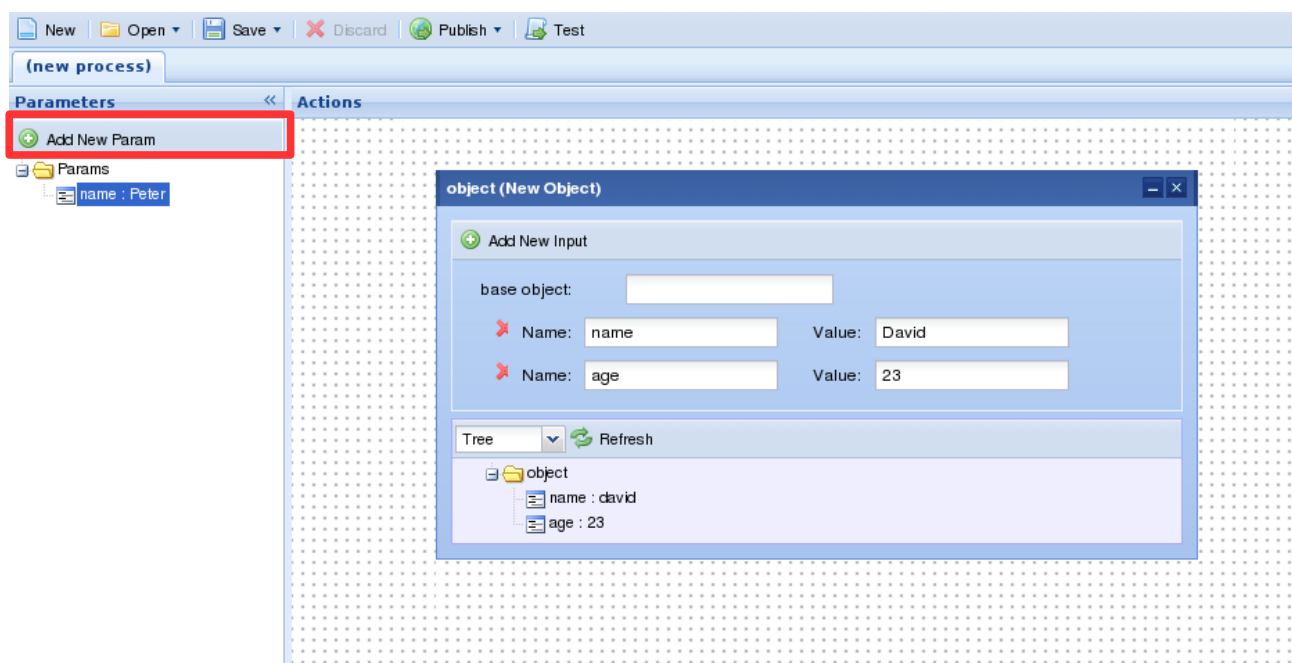


Illustration 7: MyCocktail – “Add New Param” button

Once the user has pressed this button a dialog appears to define the parameter name, the data type and its default value.

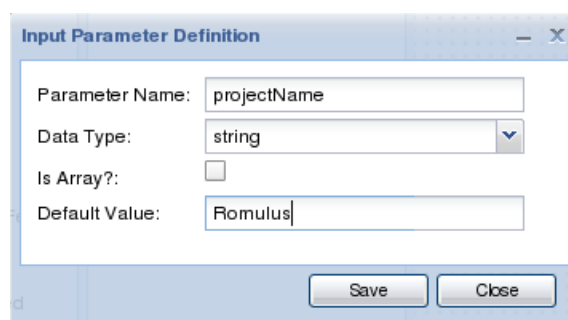
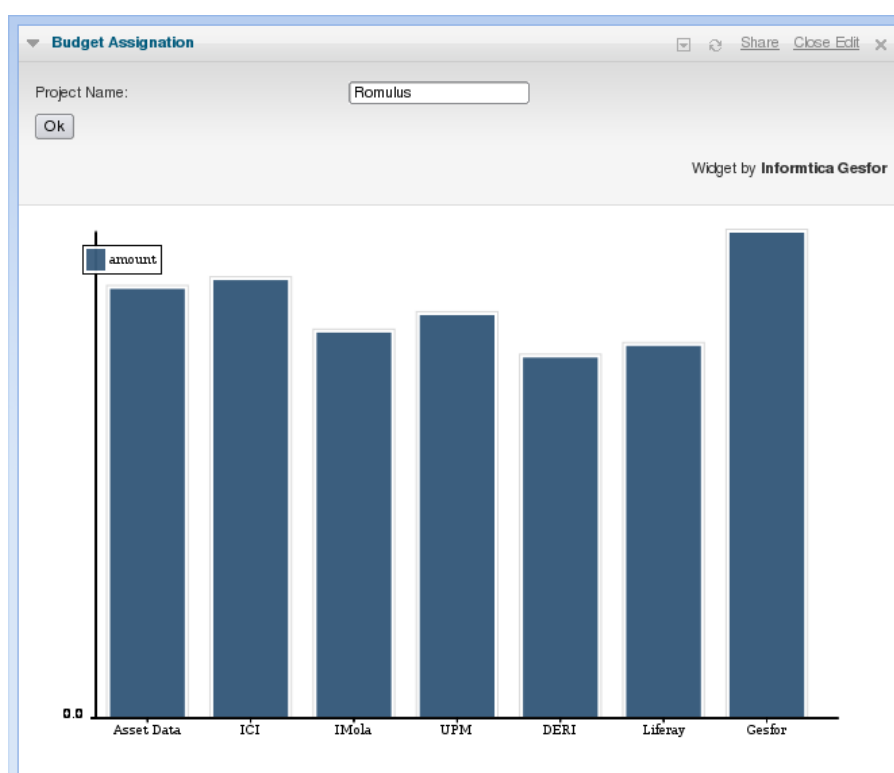


Illustration 8: Panel to introduce an input parameter

As said before, when the mashup is inserted in a page, the parameters are taken from the URL. Taking account the example of the last illustration, to pass an argument to the mashup should be done in this way:

http://mypage?projectName=MyProject: if the page where is inserted the mashup is "http://mypage", the value for projectName parameter will be "MyProject" and if any URL parameter with this name is specified the parameter takes the default value.

When the mashup is exported as Netvibes Gadget the value of the parameter should be filled from the parameters of the widget. In the next sample (Illustration 9) is shown a mashup created with MyCocktail in Netvibes web site, pressing "Edit" in the top right of the widget appears a small form to fill the parameters of the widget. This configuration has been set by MyCocktail when the mashup was exported taking into account the input parameters of the mashup.



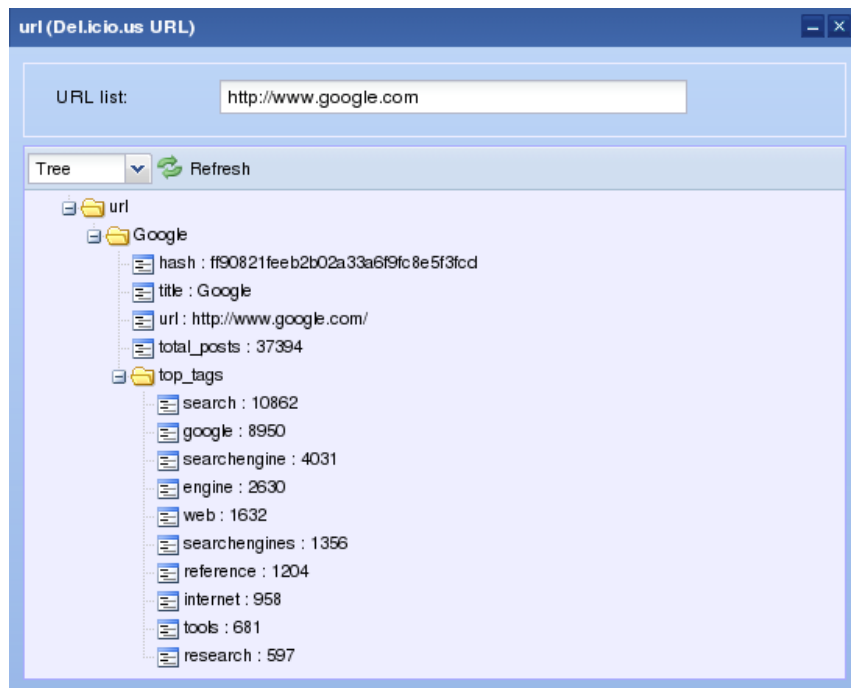
*Illustration 9: Mashup created with MyCocktail in Netvibes web site*

## 2.2.2 Services

The list of services provided by Mashup Builder are:

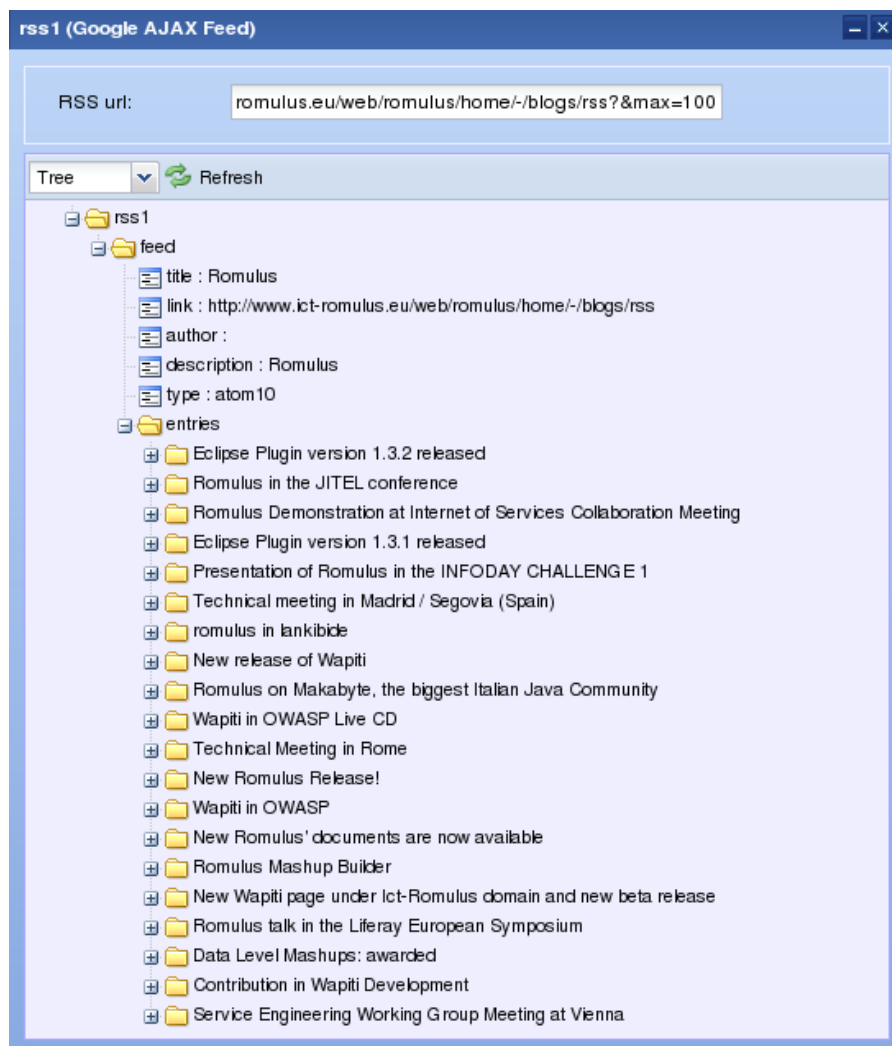
- Amazon
  - Amazon ECS Item Search: searches items in Amazon.com using Amazon E-Commerce Service.
- Delicious
  - Del.icio.us Posts: returns del.icio.us recent bookmark list of the user.
  - Del.icio.us Tags: returns del.icio.us tags of the user.

- Del.icio.us URL: returns posts information of given URL.



*Illustration 10: Panel of Del.icio.us URL service*

- Del.icio.us Network: returns del.icio.us network user id list.
- Del.icio.us Fans: returns del.icio.us fans user id list.
- Flickr
  - Public Photo Feed: public photos feed in Flickr.
- Google
  - Google AJAX Search: returns search results for given keyword query.
  - Google AJAX Feed: gets RSS/Atom feed of the URL.



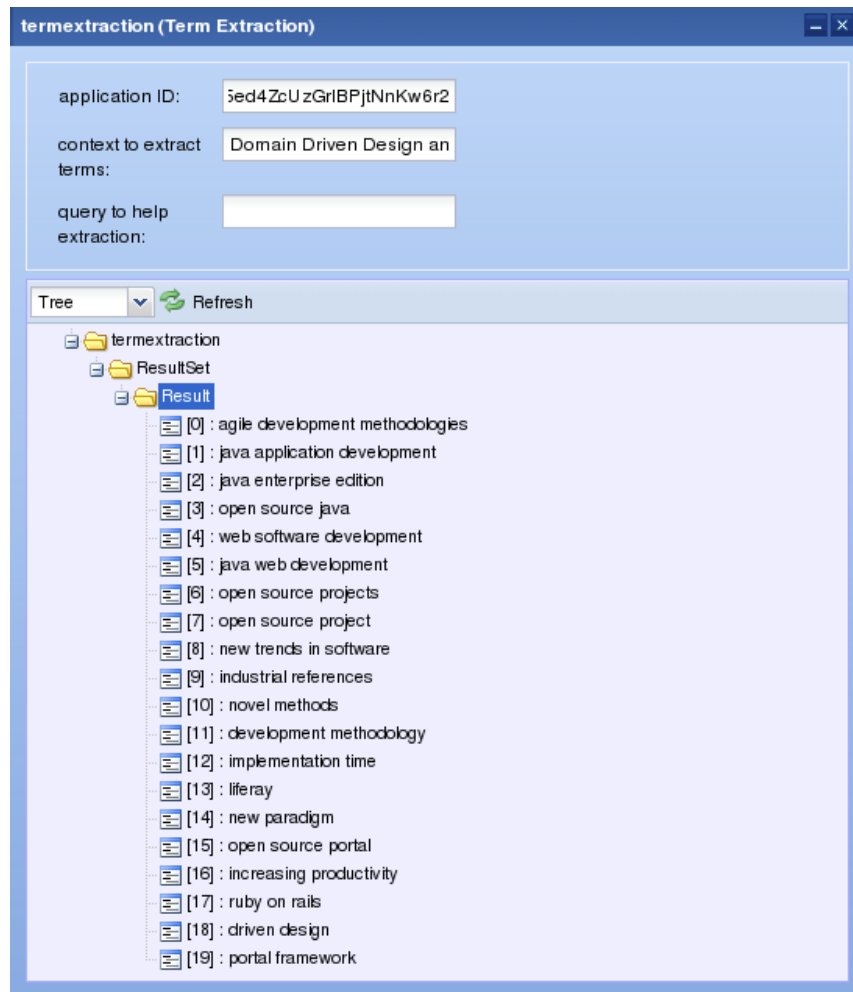
*Illustration 11: Panel of Google AJAX Feed service*

- Google AJAX Lookup Feed: Lookups RSS/Atom URL from given URL.
- Google Translate: Translates text between the given languages.



*Illustration 12: Panel of Google Translate service*

- Twitter
  - Public Timeline: returns the 20 most recent statuses from non-protected users who have set a custom user icon.
  - Friends Timeline: returns the 20 most recent statuses posted in the last 24 hours from the authenticating user.
  - User Profile: returns user profile in twitter.
  - User Timeline: returns the 20 most recent statuses posted in the last 24 hours from the authenticating user.
  - Friends: returns up to 100 of the authenticating friends of the user who have most recently updated, each with current status
  - Followers: returns the authenticating user's followers, each with current status on line.
  - Favorites: returns the 20 most recent favourite statuses for the authenticating user or user specified by the ID parameter in the requested format.
- Yahoo
  - Yahoo Web Search: returns search result from a given keyword query.
  - Fetch RSS: fetches RSS data from given URL using RSS2JSON Feed.



*Illustration 13: Panel of Yahoo Term Extraction service*

- Yahoo Term Extraction: The Term Extraction Web Service provides a list of significant words or phrases extracted from a larger content.
- Youtube
  - Youtube Videos Search: gets video feeds from Youtube.

A custom dialog has been developed to test the application extensibility, this is an example that integrates the data got from a Romulus application through a REST service. The JSON returned represents how many blogs have each user. This JSON is going to be used as example in the next sections to demonstrate the use of the operators. The application represents JSON with a tree structure to make it more readable.

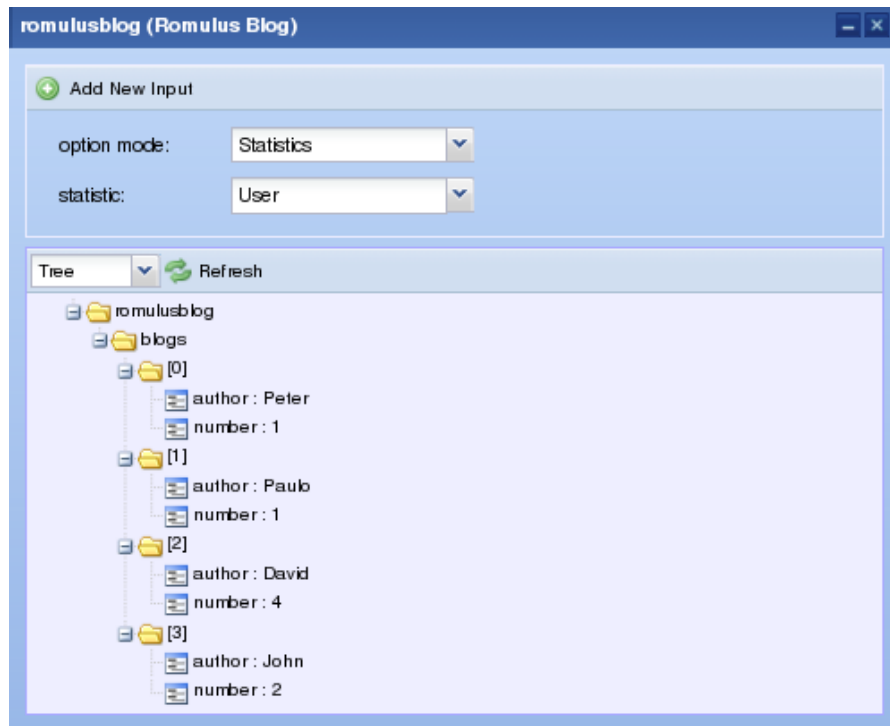


Illustration 14: JSON returned by a service represented in a tree structure.

### 2.2.2.1 Importation of a service from WADL

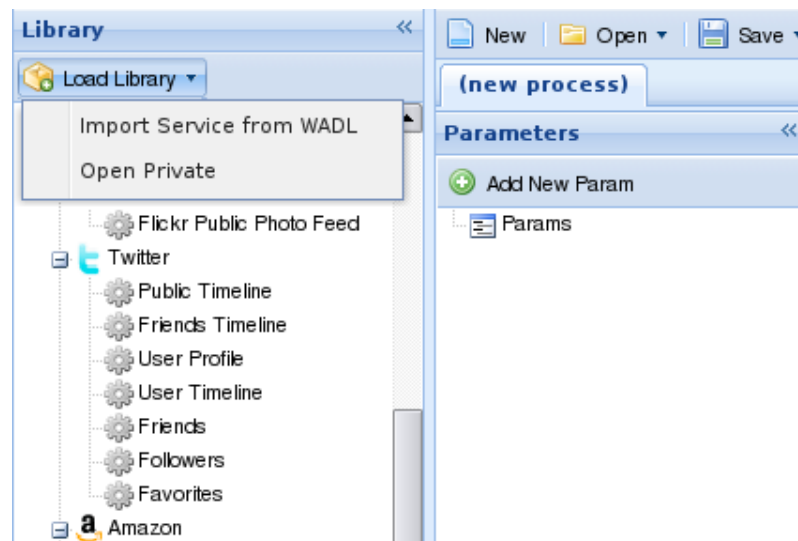
WADL (Web Application Description Language) [WADLURL] is an XML-based format language that provides a machine-readable description of HTTP-based web services, (usually REST Services).

This is a sample of WADL, it is the definition for the Google Translator Service. The WADL has to be published online to be imported in MyCocktail.

```

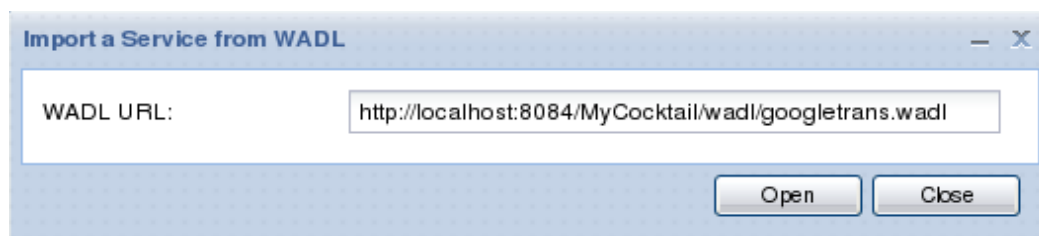
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
  <resources base="http://ajax.googleapis.com/ajax/services/">
    <resource path="language/translate">
      <method name="GET" id="translate">
        <request>
          <param type="xs:string" style="template" name="v" fixed="1.0" />
          <param type="xs:string" style="query" name="q"/>
          <param type="xs:string" style="query" name="langpair"/>
        </request>
        <response>
          <representation mediaType="application/json"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
  
```

MyCocktail is able to read a subset of WADL, does not allow all the WADL possibilities because it is focused in the most common format of REST Services like Google, Yahoo or Flickr ones. The option for the WADL importation is available in the top left of MyCocktail window (see Illustration 15).



*Illustration 15: Import Service from WADL menu*

The next dialogue appears when the user selects the “Import Service from WADL” in the menu. Filling the field with the URL where is the WADL file, MyCocktail will retrieve it and transform into a MyCocktail service.



*Illustration 16: Dialog to import a WADL Service*

Once the WADL is processed, a new service appears in the Services panel, the number of services generated by MyCocktail depends of the services defined into the WADL.



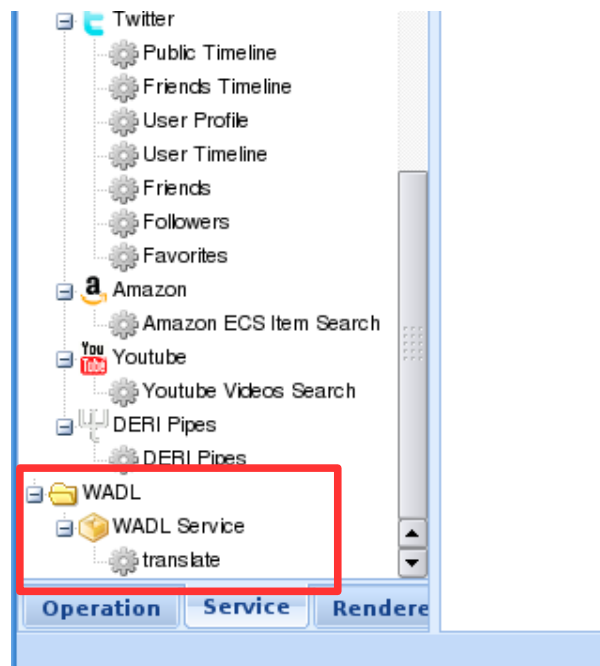


Illustration 17: The generated service in the Services panel

This is the panel generated automatically from the WADL, and the response retrieved from the service (Illustration 18). This panel appears when the service is dragged and dropped from the Services Panel to the Action Panel.



Illustration 18: Panel for the service generated from WADL

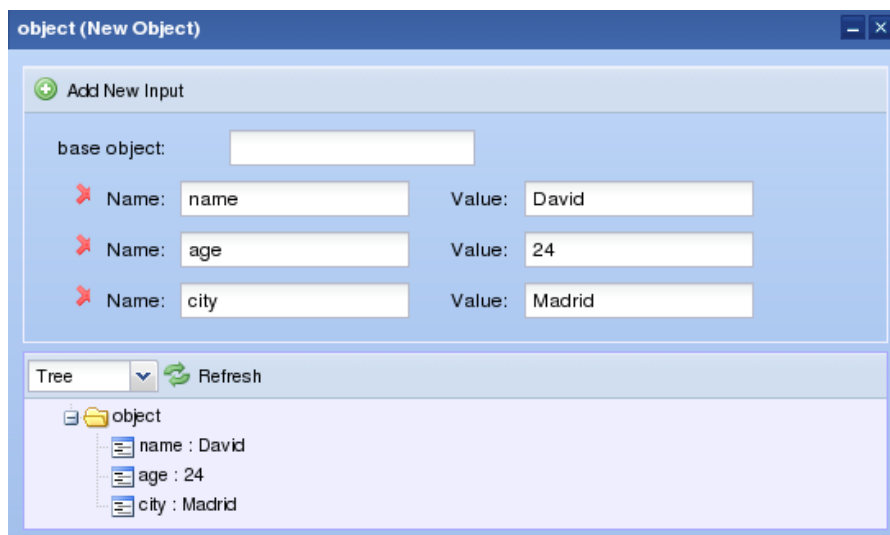
## 2.2.3 Operators

The response of the server invocation usually is a JSON (JavaScript Object Notation) which is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays that are called objects. This associative arrays can be manipulated with the operators available.

### Basic Operators

#### New Object

This operator creates a new object, extending from a base object. The base object field is not mandatory and more properties can be also added to the base object clicking on the button “Add New Input”. The “Name” field is the name of the property and the “Value” field is the value of this property.



*Illustration 19: Panel of the New Object operator*

#### Branch

This operator evaluates the condition given to redirect the flow of the application.

#### Cast

Force-casting given value to specified type.

#### Key/Value Pair List

It creates an array from an object, which contains key/value pair of the object properties.



*Illustration 20: Panel of the Key/Value Pair List operator*

## Array Operators

This kind of operators makes operations with arrays, the elements of this arrays typically are objects (associative arrays).

### New array

This functionality creates a new array extending from a existing one.

### Iterate

This operator iterate a array and open a new process for each element of the array, the new process development is similar to a normal process. It collects each element output and emits a new output array.

### Filter

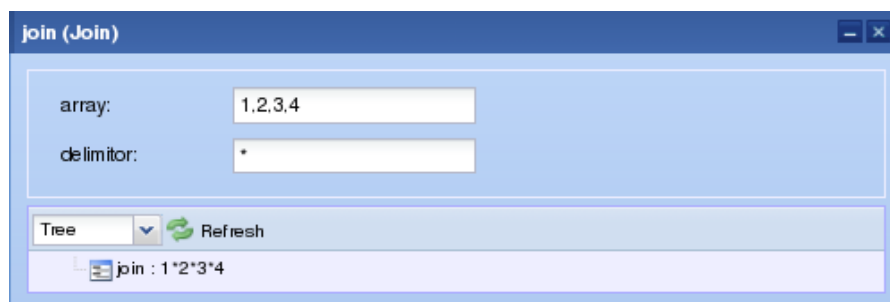
This option filters the array elements which the given conditions. To add a new condition, the user has to click in the above button “Add New Condition” and if there is more than one condition, they can be combined with an AND or OR operator (condition mode).



*Illustration 21: Panel of the filter operator*

## Join

This operator puts all elements of the array into a string, the elements are separated by given delimiter.



*Illustration 22: Panel of the join operator*

## Slice

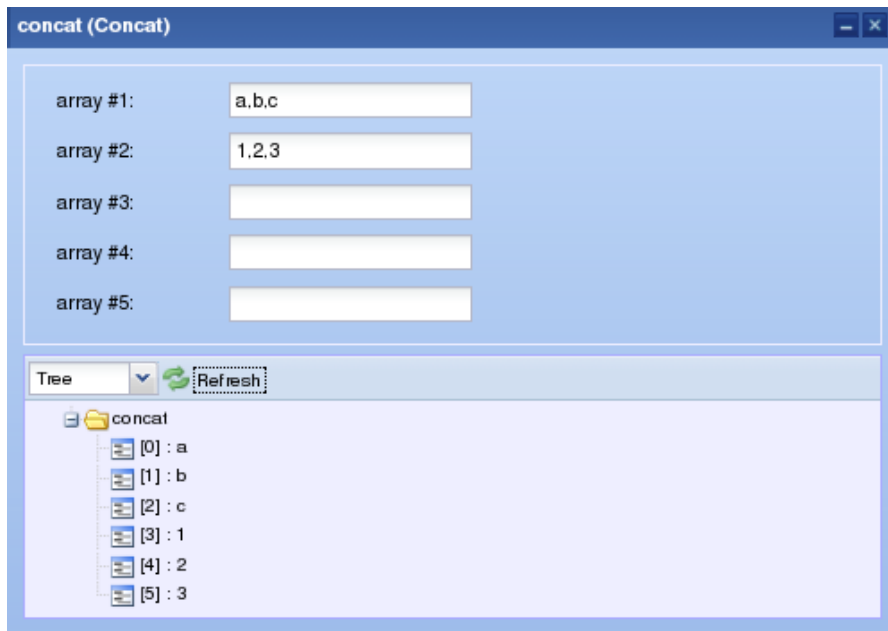
It gets a selected subset of a given array.



*Illustration 23.: Panel of the slice operator*

## Concat

This operator joins more than two arrays in one.



*Illustration 24: Panel of the concat operator*

## Flatten

This functionality turns multidimensional arrays into linear ones.

## Pluck

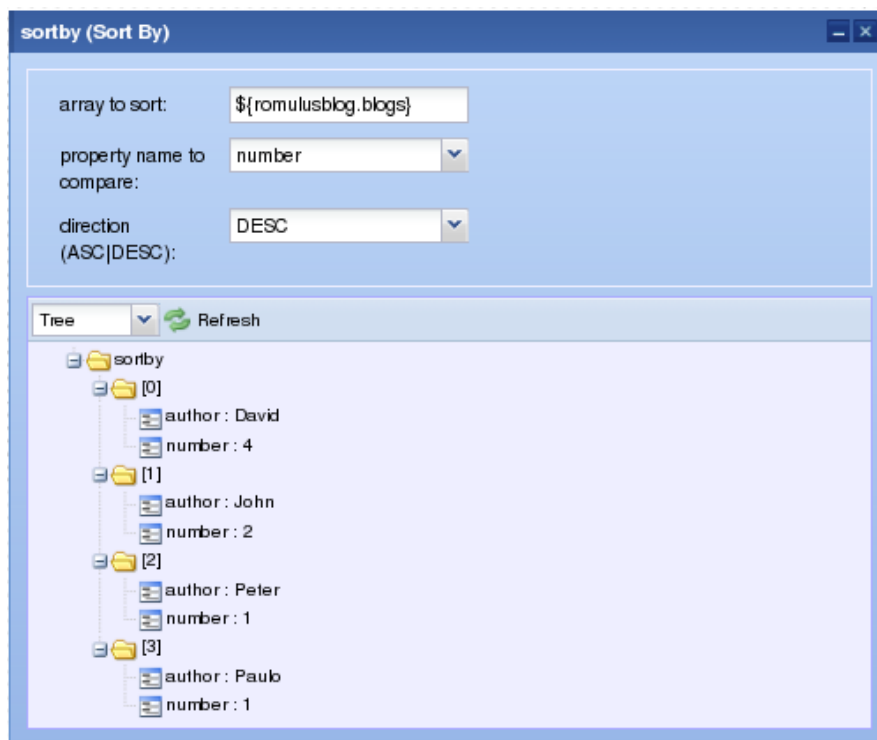
This operator retrieves the value to the specified property in each element of the given array and returns the results in a new array.



*Illustration 25: Panel of the pluck operator*

## Sort By

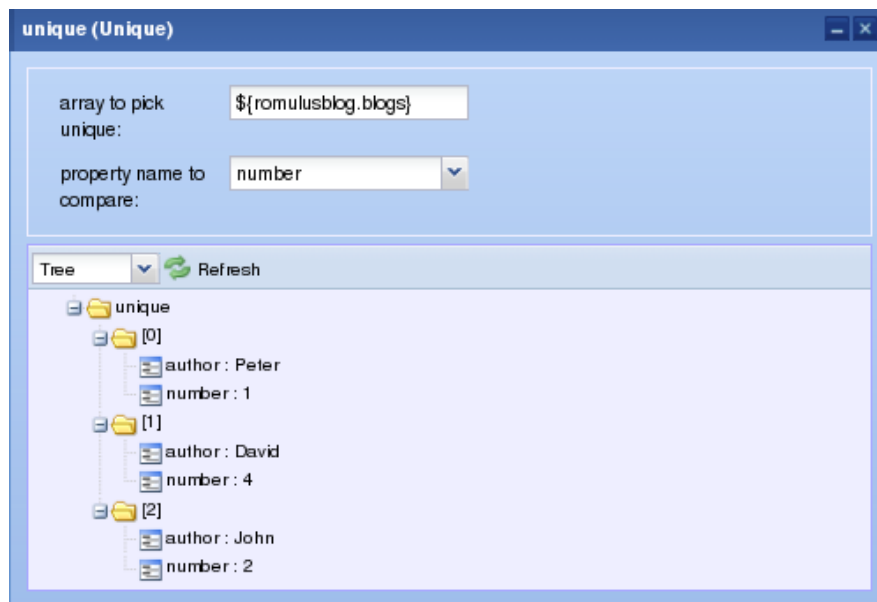
It allows to sort an array chosen the array to sort, the property name of the objects to compare and the direction of the sort (ascendant or descendent).



*Illustration 26: Panel of the sort by operator*

## Unique

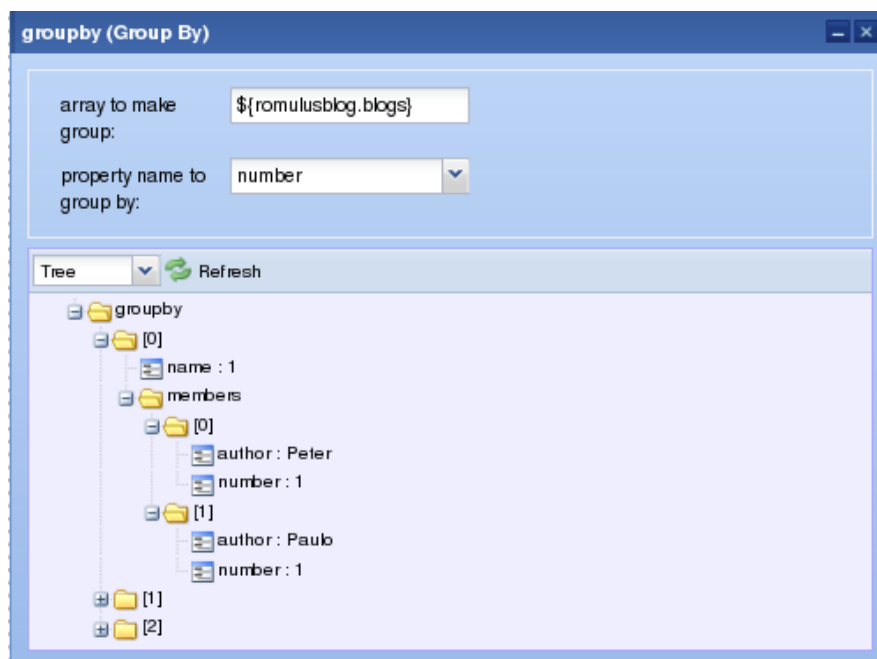
Pick unique elements from given array. All duplicate elements are removed, only first one left.



*Illustration 27: Panel of the unique operator*

## Group By

This option divides an array into sub array groups. Grouping is based on property value.

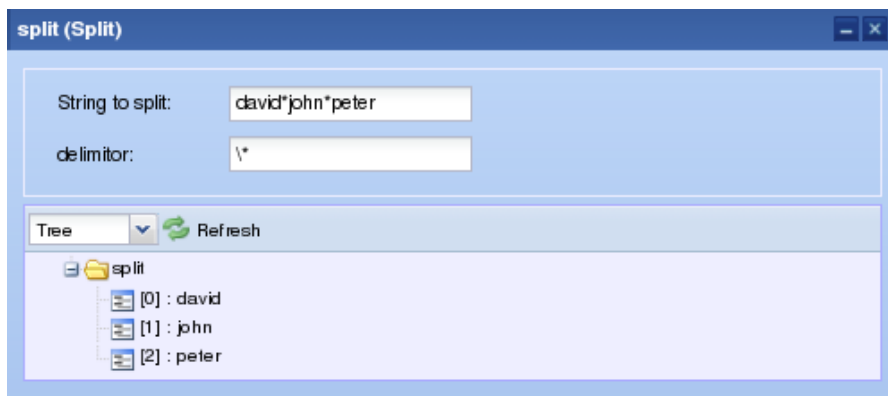


*Illustration 28: Panel of the group by operator*

## String Operators

### Split

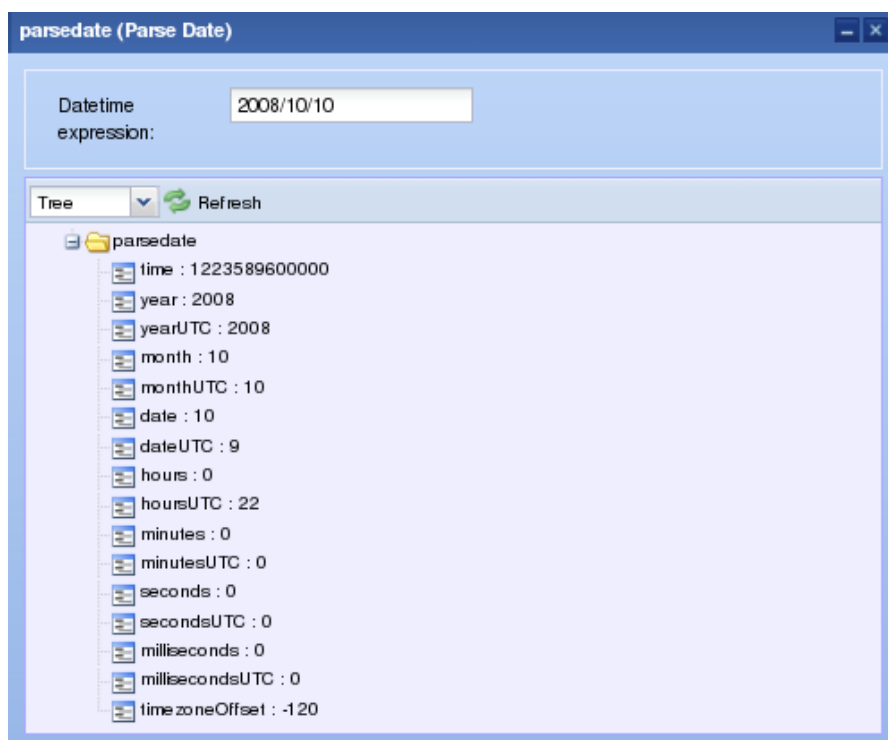
This functionality splits the given string into array, cutting the string by the given delimiter



*Illustration 29: Panel of the split operator*

### Parse date

This operator parses a date from Unix epoch or W3C-DTF formatted string.

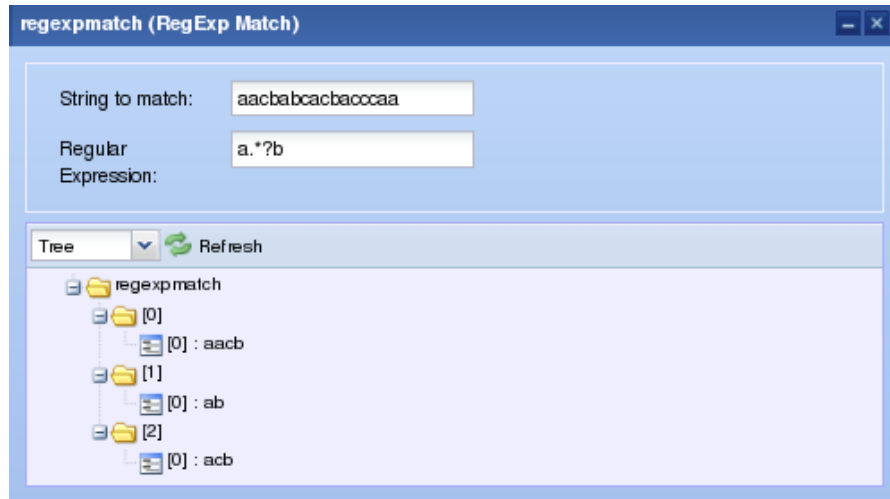


*Illustration 30: Panel of the parse date operator*



## RegExp match

It matches entire string with given regular expression.

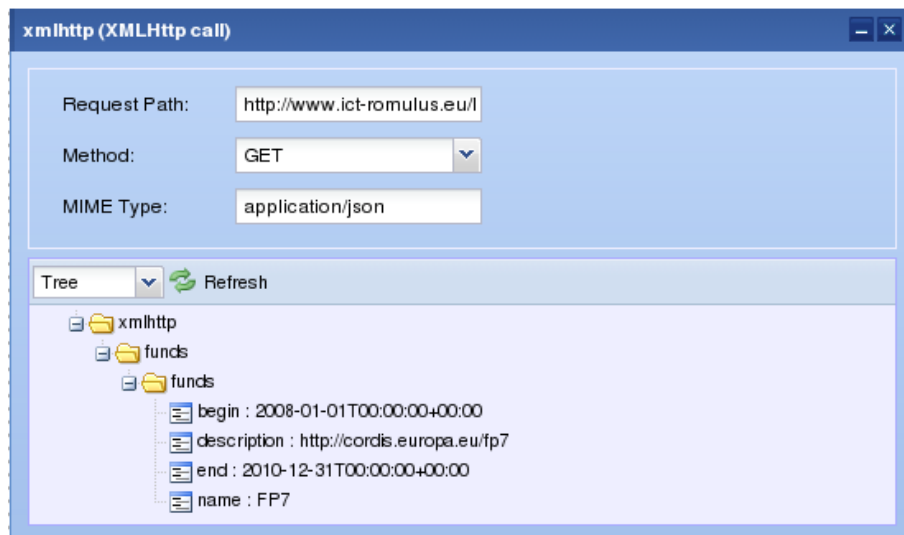


*Illustration 31: Panel of the RegExp match operator*

## Ajax Operators

### XMLHttpRequest

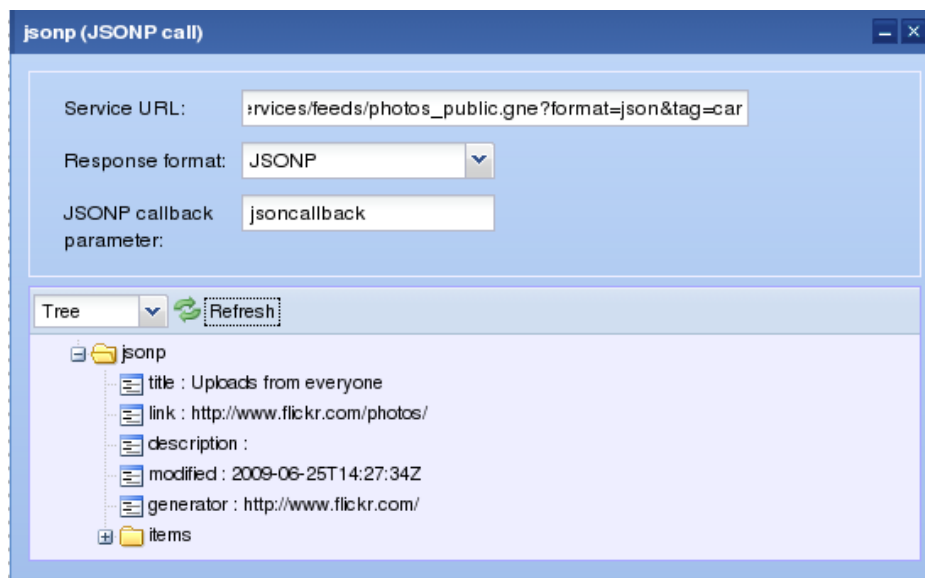
This operator makes an Ajax call to a URL given. If the MIME Type of the response is specified and accepted by MyCocktail, the information is ordered in folders and items.



*Illustration 32: Panel of the XMLHttpRequest operator*

## JSONP call

It makes an invocation to a REST service in order to get a JavaScript response which has an invocation to a function with a JSON passed in the parameter of the function. All the services are implemented using this operator. They provides a smarter way to complete the parameters of the RESTful service meanwhile in the JSONP call operator the user has to build the URL with the parameters manually.



*Illustration 33: Panel of the JSONP operator*

## 2.2.4 Renderers

### Anchor Link

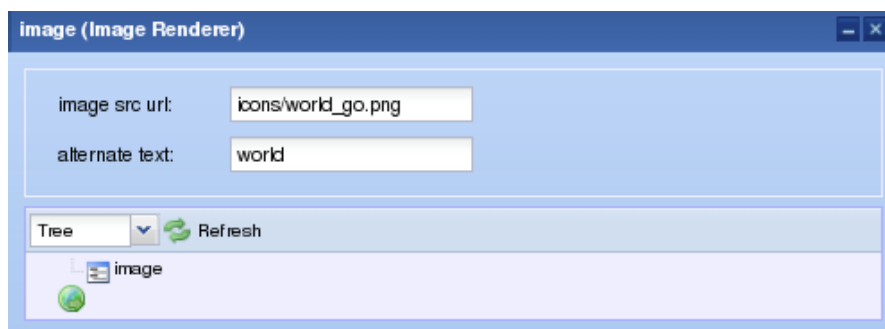
This option renders an anchor link.



*Illustration 34: Panel of the Link renderer*

### Image Renderer

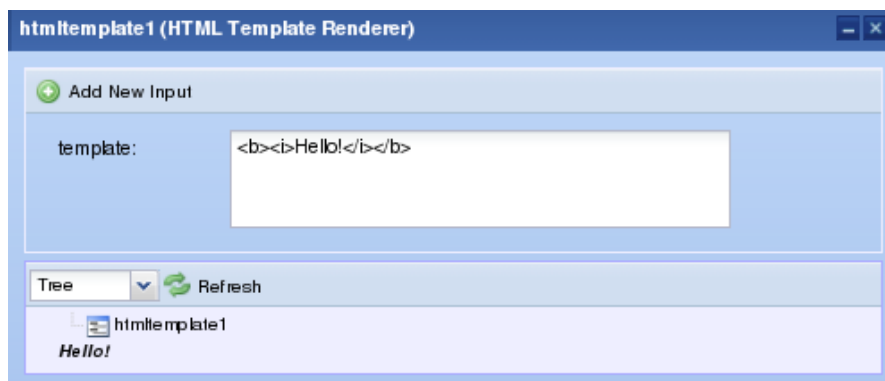
It renders a image with the given URL.



*Illustration 35: Panel of the image renderer*

### HTML Template Renderer

This functionality generates HTML from template. Input properties can be referred and embedded by following notation `<br/>#{<i>propname</i>}`,



*Illustration 36: Panel of the filter renderer*

## List Renderer

It renders an array data in bullet list format (unordered list).



*Illustration 37: Panel of the list renderer*

## Table Render

This option renders a multidimensional array (a matrix) like a table.



*Illustration 38: Panel of the table renderer*

## Tile Renderer

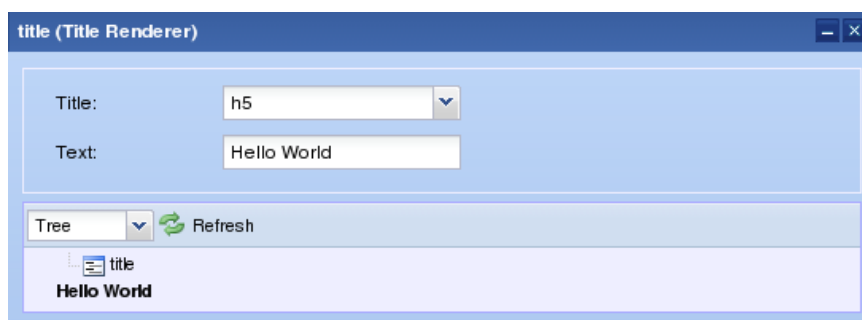
This functionality renders an array data in tile.



*Illustration 39: Panel of the tile renderer*

## Title Renderer

This options renders a HTML Title with the given text.



*Illustration 40: Panel of the Title renderer*

## Text Renderer

It creates a text with the font characteristics given.



*Illustration 41: Panel of the Text renderer*

## Div Renderer

It creates a div that contains another divs, each div can be filled with another render or with text. Each div has the same height and width.

## Pie Chart

It renders a pie chart from the given inputs records. This records must be an array of objects. The label field is the member of the object which will be the label, the render allows to choose between the variables of the object. The value field is the variable of the objects which the render uses to calculate the percent for the chart. Also the width, weight and colour can be configured.

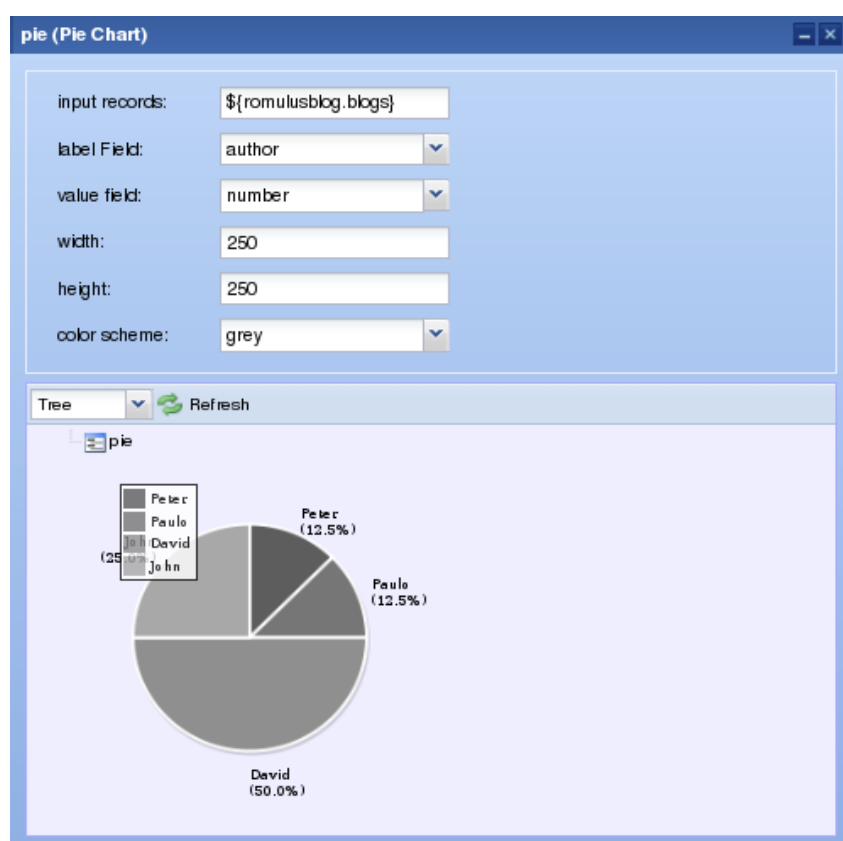
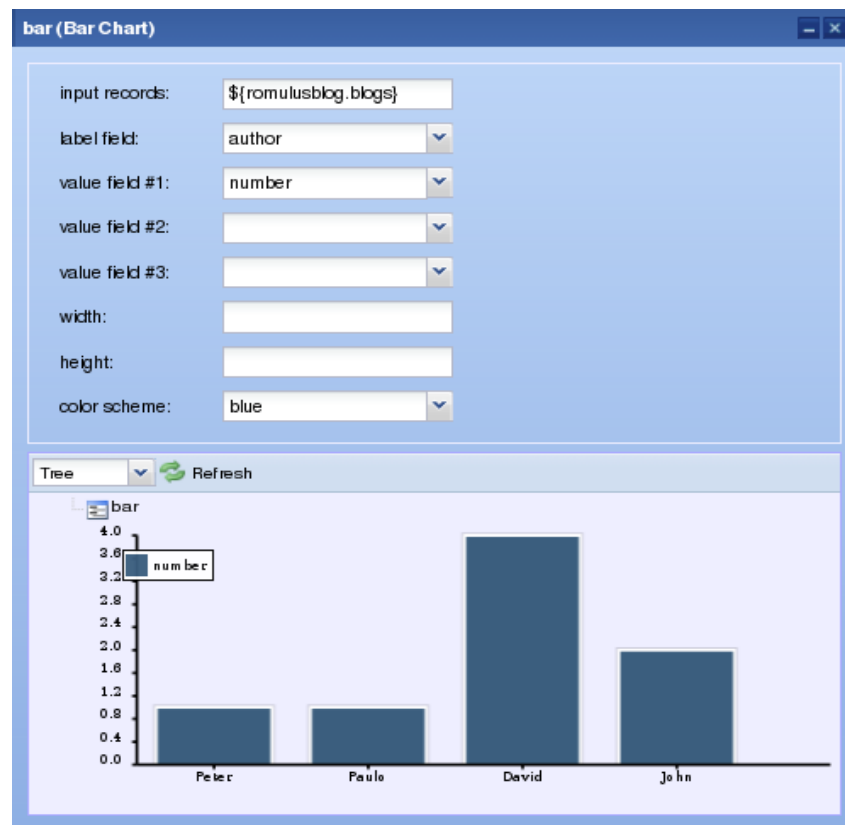


Illustration 42: Panel of the pie chart renderer

## Bar Chart

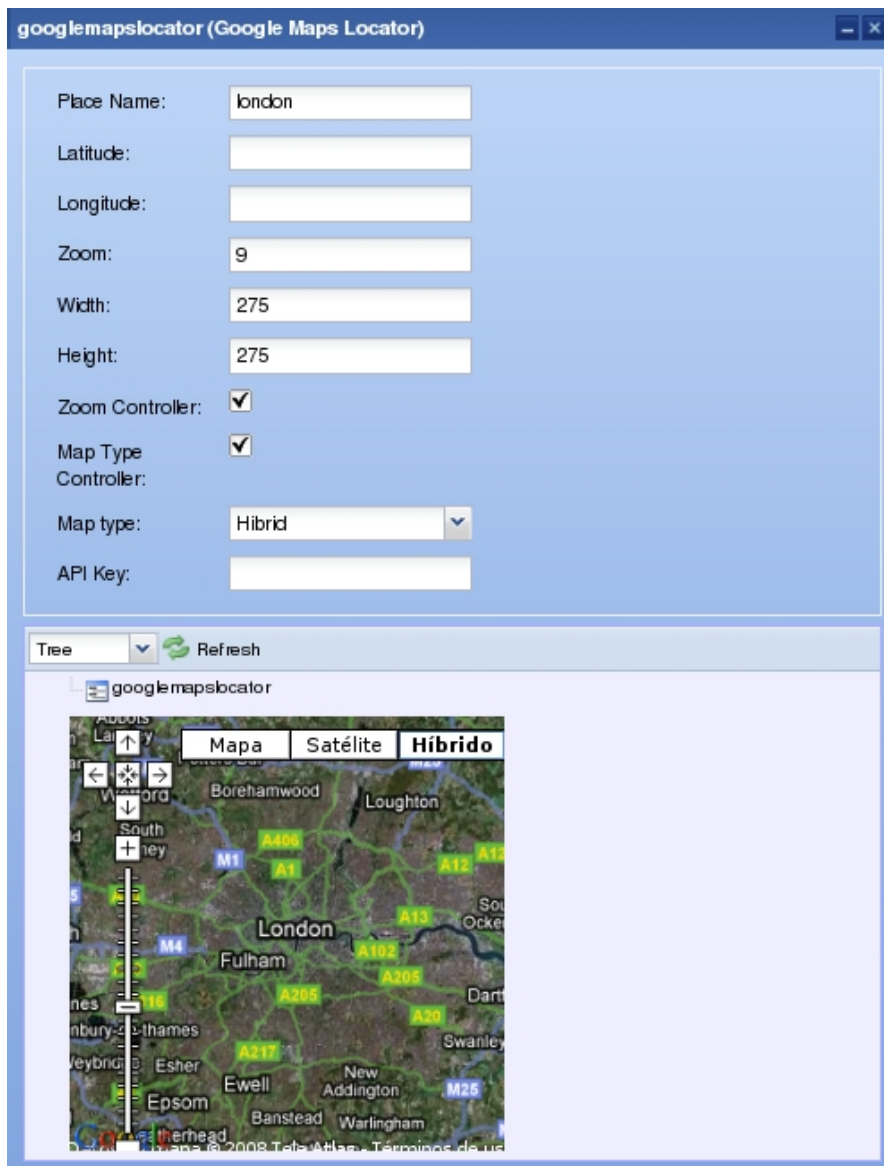
This render is similar to the previous one, the pie chart, but this can represent three value variables in the chart.



*Illustration 43: Panel of the bar chart renderer*

## Google Maps Locator

This render centres a map of Google Maps in the point or place indicated in “Place Name” field. To centre a map in a point the latitude and longitude fields must be filled. The zoom, width and height of the map can be specified. If the zoom and map type controller fields are checked this controllers will be displayed on the map. The API key must be specified if the mashup will be published in a public server. This key can be requested to Google for this public server and it is free.



The screenshot shows a web application window titled "googlemapslocator (Google Maps Locator)". It contains a form with the following fields and controls:

- Place Name:** A text input field containing "london".
- Latitude:** An empty text input field.
- Longitude:** An empty text input field.
- Zoom:** A text input field containing "9".
- Width:** A text input field containing "275".
- Height:** A text input field containing "275".
- Zoom Controller:** A checkbox that is checked.
- Map Type Controller:** A checkbox that is checked.
- Map type:** A dropdown menu showing "Híbrido".
- API Key:** An empty text input field.

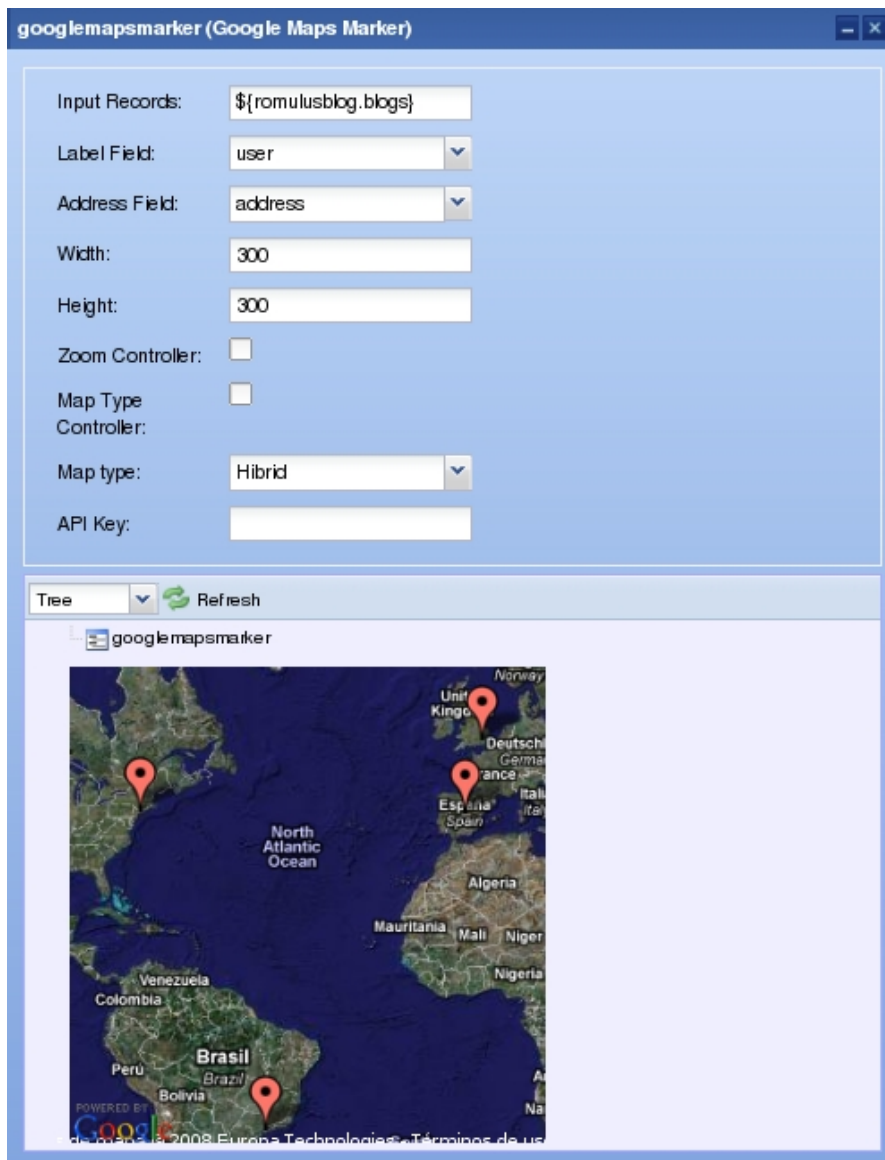
Below the form is a section with a "Tree" dropdown and a "Refresh" button. The main area displays a Google Map of London, centered on the city. The map has a toolbar with navigation controls and a tab bar at the top showing "Mapa", "Satélite", and "Híbrido" (which is selected). The map shows various landmarks, roads, and labels like "London", "Fulham", "Epsom", and "Banstead".

Illustration 44: Panel of the Google Maps locator renderer



## Google Maps Marker

This option draws marks on a map of Google Maps and centres it to make visible all the marks shown. It is a little bit different to the previous one render, the input records must be an array that contains objects. In the "Address Field" must be selected the variable of the object that contains an address (for instance, London, 5<sup>th</sup> Avenue New York, etc.). In the Label Field can be selected a object variable which will be the text displayed when the user makes click in the marker. The rest of the fields are the same that in the previous render.



*Illustration 45: Panel of the Google Maps Marker renderer*

## Timeline renderer

This renderer draws the duration of events along the time with a line knowing the start and end date of these events. The “Input records” field must be an array that contains objects. The most significant fields are “Label Field”, “Initial date” and “Final date”, the form has combos to select these fields, the selectors are filled with the properties of the objects contained in the array.

- “Label Field”, the variable of the object that contains the text to show above the line.
- “Initial date”, the start date of the event and
- “Final date”, the end date of the event.

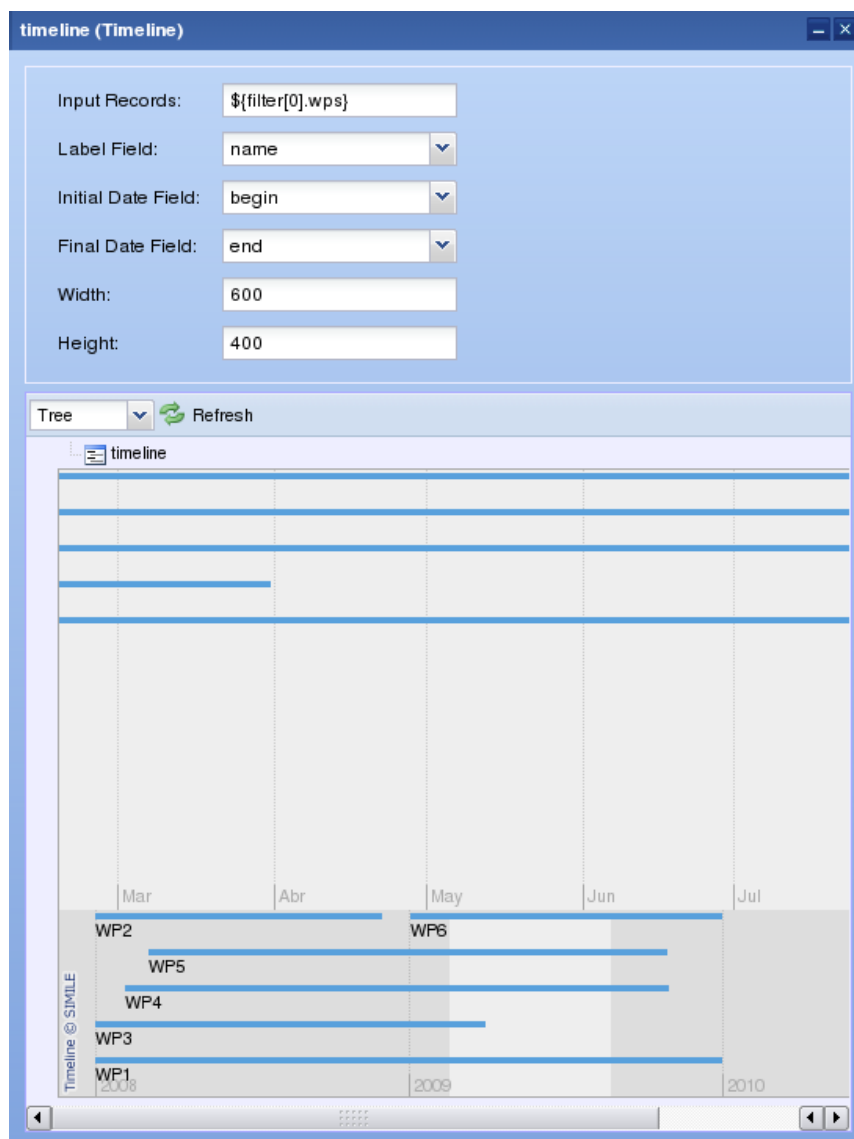
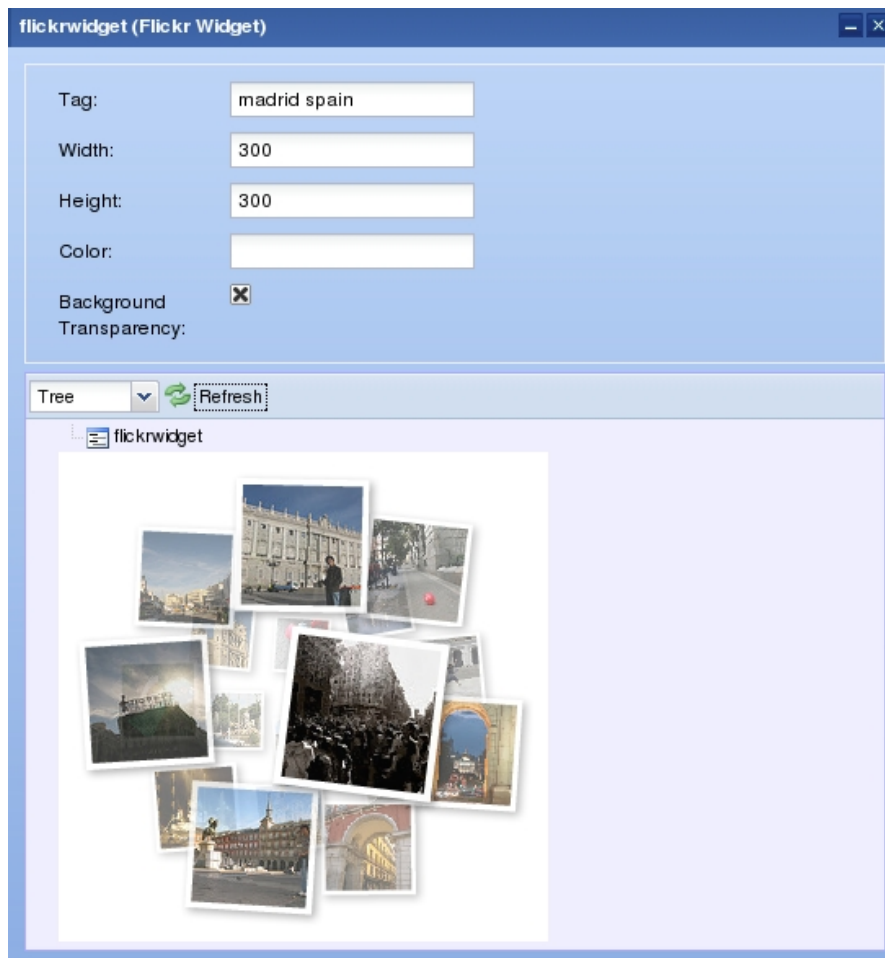


Illustration 46: Timeline renderer

## Flickr Widget

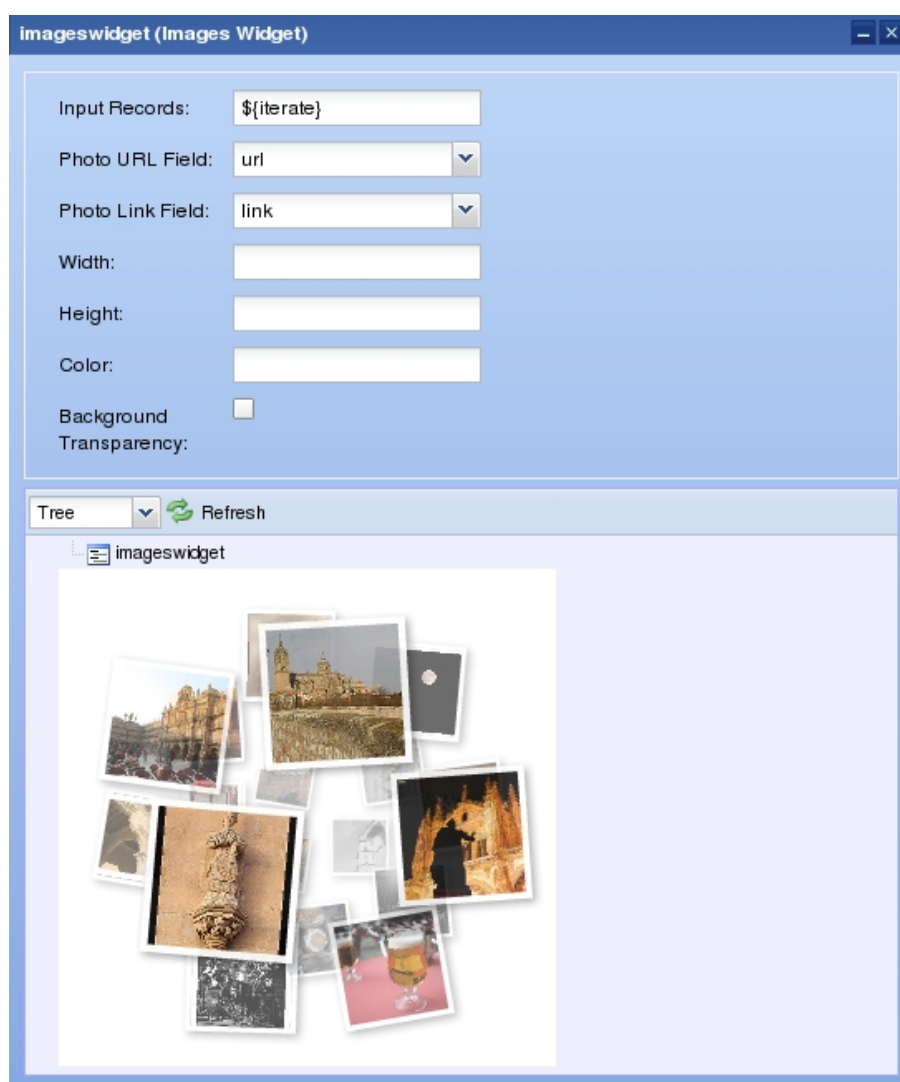
This renderer displays a Flash component which shows Flickr photos composing a sphere. This render has a simple form, in the field Tag should be filled the tag to search the photos, and also can be specified the width, height, background colour or background transparency.



*Illustration 47: Flickr Widget*

## Images Widget

This renderer displays a Flash component which shows photos composing a sphere. It is similar to the previous widgets but this time user can choose the source for the images and they are not retrieved from Flickr automatically. The “Input records” field has to be an array that contains objects, this objects have to contain the URL where is published the images to display. In the fields “Photo URL Field” and “Photo Link Field” have to be selected the properties of the object which contain the URL to the photo and the URL to the page which will be shown when the user click on the image.



*Illustration 48: Images Widget*

## Tag Cloud Widget

It displays a Flash component with a tag cloud, the words change their position when the mouse is over them. The "Input records" field must be an array that contains objects, in the fields "Tag URL Field", "Tag Name Field" and "Tag Size Field" must be selected in the selector a property of the object contained in the array passed into "Input records" field. The most important fields are:

- Tag URL Field, the URL which will be open when the user press in the word
- Tag Name Field, the text to show in the word
- Tag Size Field, the weight of the word with respect to the rest of word. If this field is not specified the weight will be assigned taking into account the order of the objects in the array, being the first one the one with most weight.
- Base URL, if this field is filled, the URL of the words will be the composition of this field and the Tag URL Field.

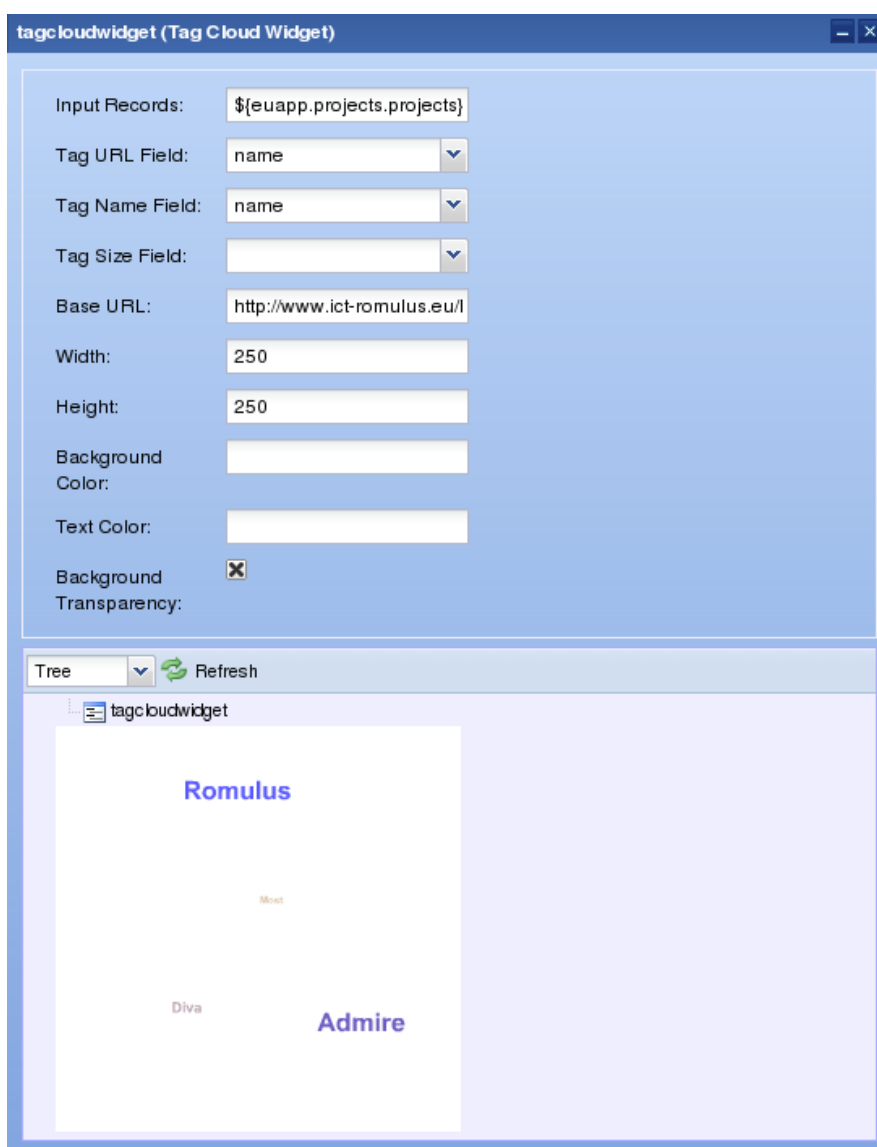


Illustration 49: Tag Cloud Widget

### 3 Page Editor

The Page Editor is a functionality of MyCocktail which allows to design a web page through a GUI allowing to integrate mashups created with MyCocktail. It also allows to introduce text and formats it with FCKEditor, a web text editor.

This is the interface of Page Editor, it is divided in two main parts: the toolbars in the top of the window and the design area below the toolbars.

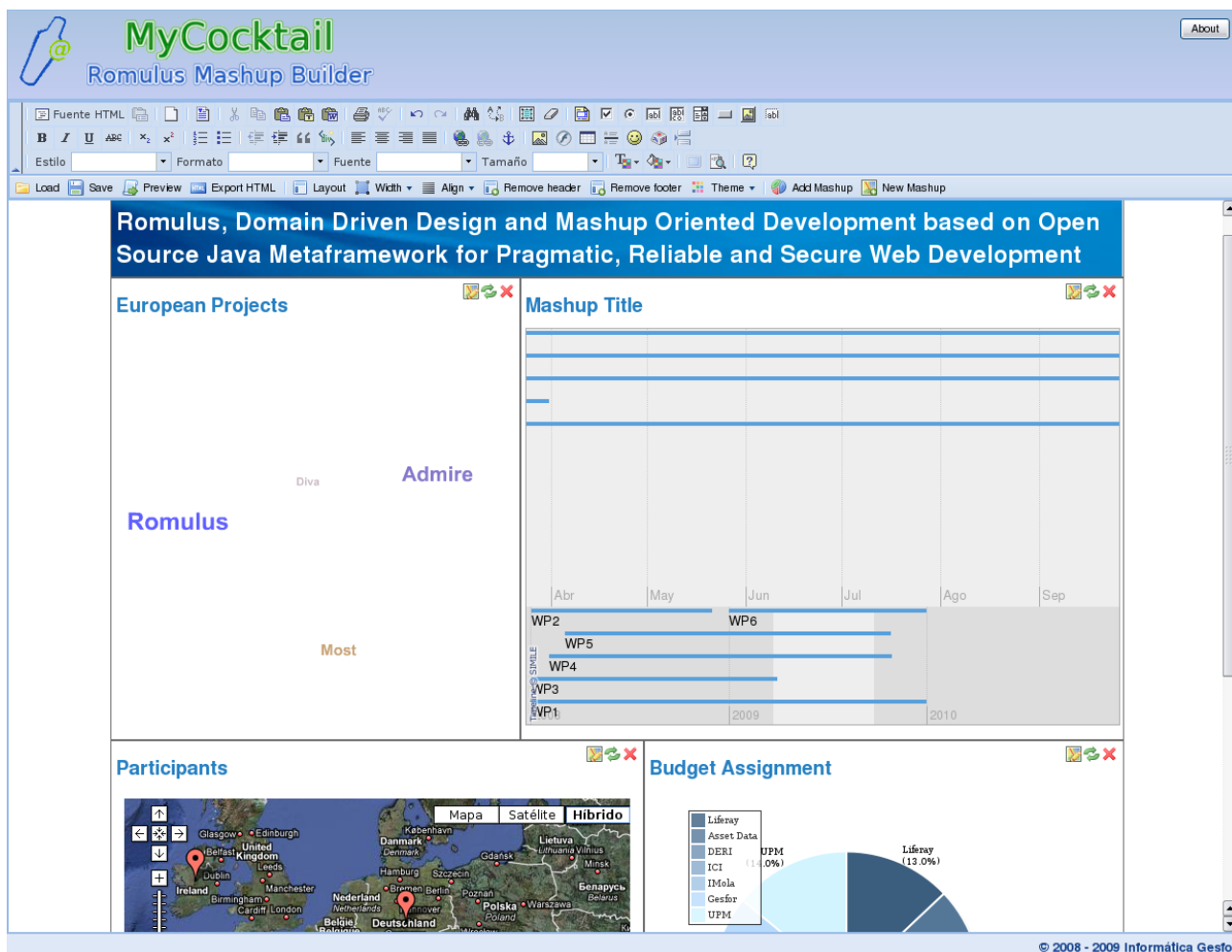


Illustration 50: MyCocktail Page Editor

## 3.1 Toolbars

In the top of the window are two toolbars: the FCKEditor Toolbar and the Page Editor Toolbar.

### 3.1.1 FCKEditor Toolbar

This toolbar belongs to FCKEditor [FckGURL], this editor has been integrated in MyCocktail.

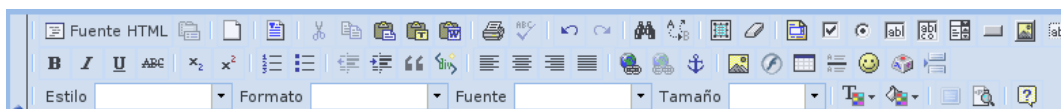


Illustration 51: FCKEditor Toolbar

### 3.1.2 Page Editor Toolbar



Illustration 52: Page Editor Toolbar

This tool bar has the following functionalities:

- Save: to save a page the page in JSON format, the text should be copied and pasted to a file to save it in local disk.

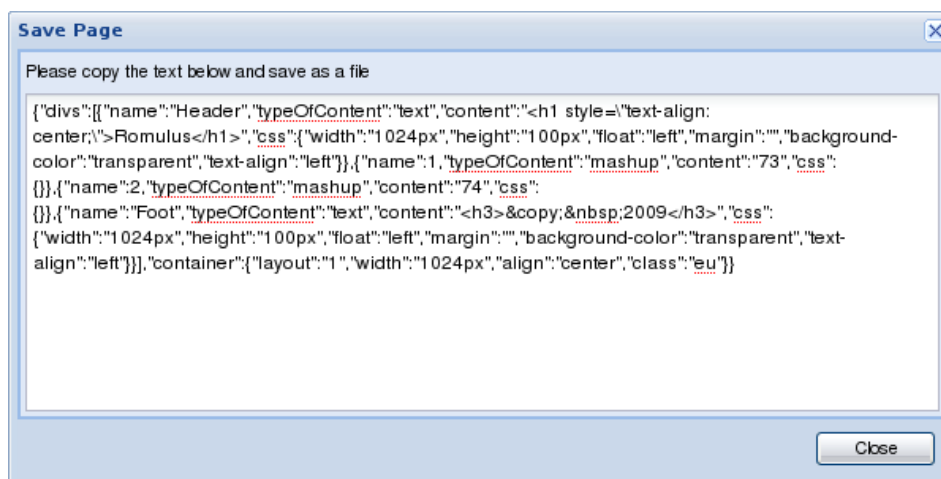


Illustration 53: Save dialogue

- Load: this option loads in the editor a page with the provided configuration, this configuration is provided when the page is saved.
- Preview: to see a preview in another browser window

- Export HTML: this option exports the page in HTML format.

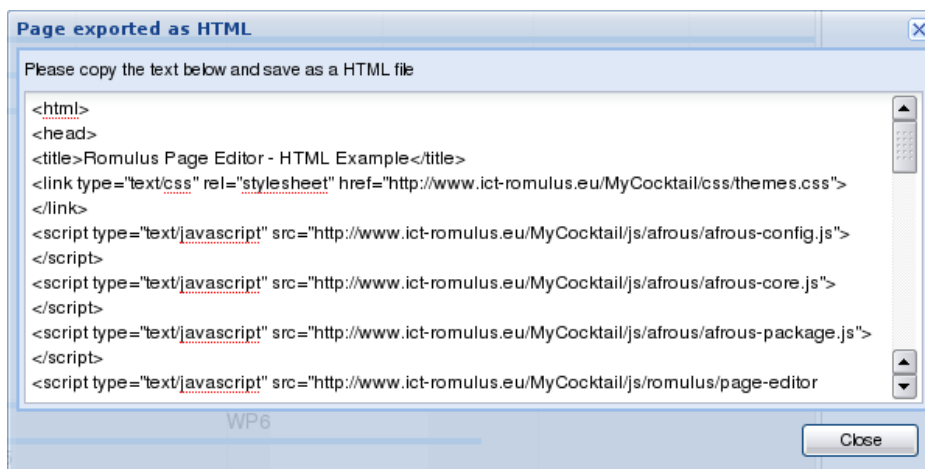


Illustration 54: Dialogue with the HTML to export the page

- Width: It allows to choose a width for the page (800 pixels, 1024 pixels or 100%)
- Align: If the page does not fill all the width of the window with this option can be selected the align of the page (left, centre and right).
- Layout: this option displays the dialogue shown in the next illustration, it allows the user to choose a layout for his page.

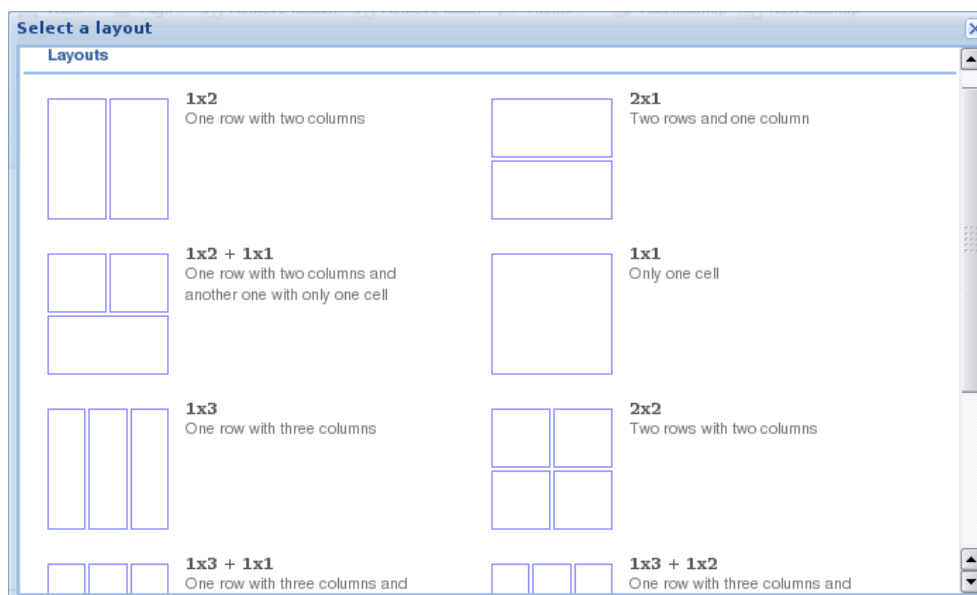


Illustration 55: Layout Selector

- Add/Remove header: To add or remove the page header
- Add/Remove footer: To add or remove the page foot

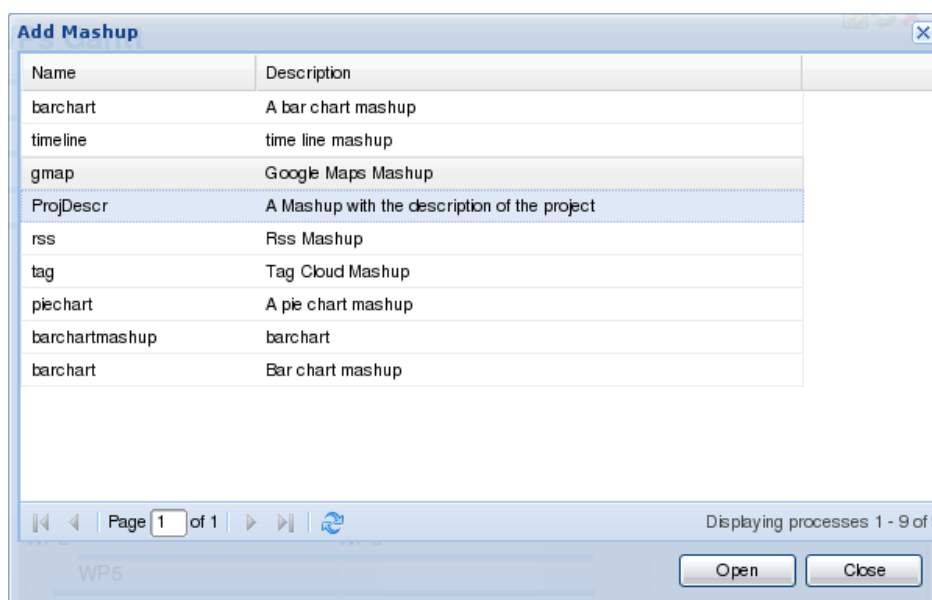


- Theme: A theme can be selected to decorate the page.



*Illustration 56: Theme Selector*

- Add Mashup: This option allows to insert a mashup in the selected area. The mashup has to be previously created and saved with MyCocktail Mashup Builder.



*Illustration 57: Dialogue to insert mashups*

- New Mashup: This button redirects to MyCocktail Mashup Builder.

### 3.2 Design area

This zone is where the user can design the page, this area is a rough representation of the resulting page. The layout of the page can be changed using the Layout button in the Editor Toolbar. The design area is composed by a header, a footer and a variable number of areas between the header and the footer which depend to the chosen layout (Illustration 58).

The header and the footer is optional, the user can add or remove it with the button in the Editor Toolbar (see 3.1.2 Page Editor Toolbar). In all the areas the user can insert text using FCKEditor (Area 1 in Illustration 58) or mashups previously created with MyCocktail (Area 2 in Illustration 58).

The text insertion is too easy, the user has to write the text and then formats it with the FCKEditor toolbar (see 3.1.1 FCKEditor Toolbar3.1.1). The text by default has the format defined in the selected theme by the user, although the text can be changed manually using the formatting options of FCKEditor toolbar.

In the areas mashups can be also inserted using the “Add Mashup” button (see 3.1.2 Page Editor Toolbar).

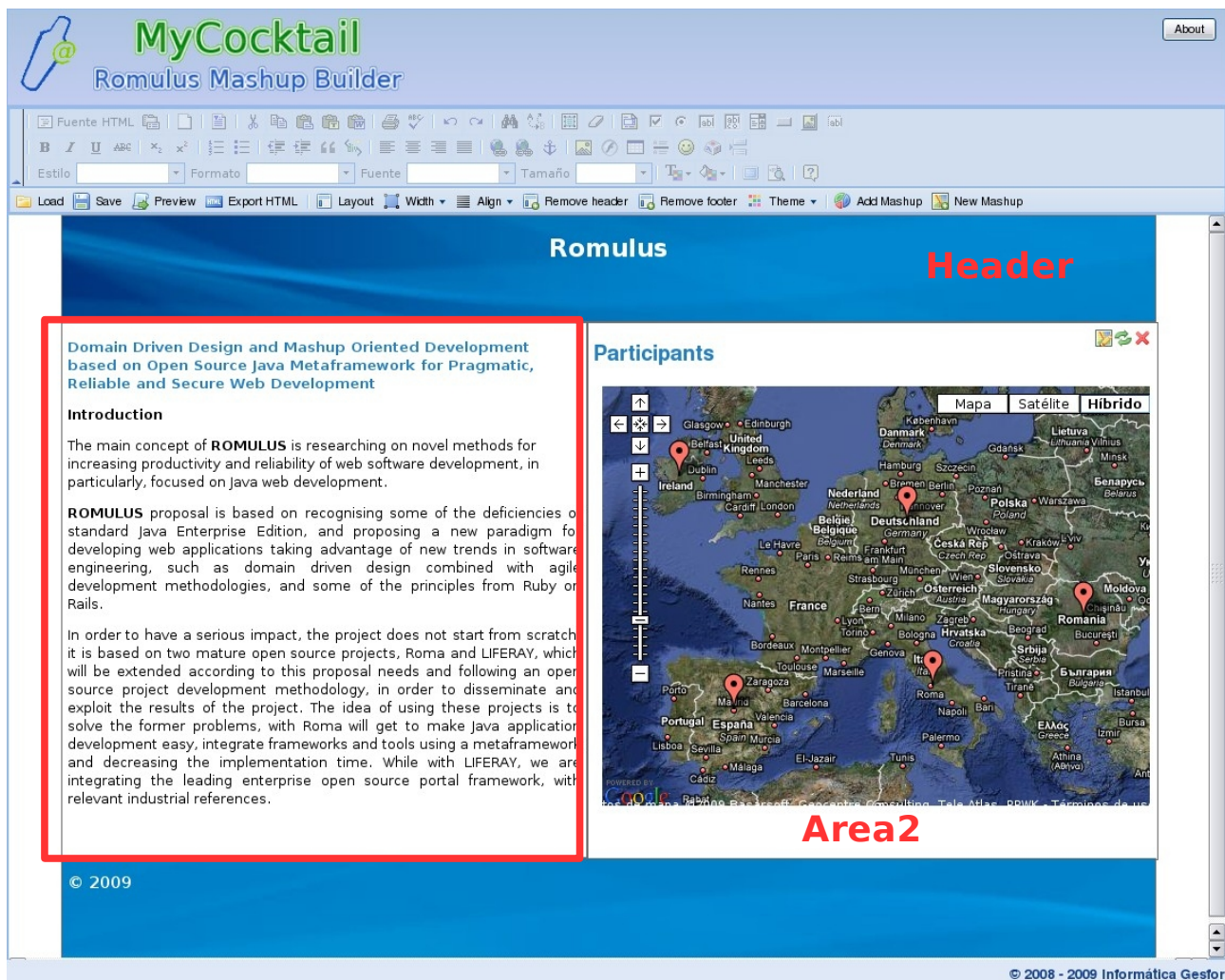





Illustration 58: MyCocktail Page Editor

If a mashup is inserted in the area, a toolbar appears in the top right of the area.



*Illustration 59: Mashup toolbar in the areas of the design area*

The buttons perform the following functionalities:

-  Edit the mashup in the mashup builder.
-  Refresh the mashup, the editor reloads the mashup, the mashup reflects the changes if the mashup or the data used by the mashup has been updated.
-  Delete the mashup.

## 4 Website and online demo

In the MyCocktail web site you can be tuned about new releases of MyCocktail. The website contains videos and demonstrations, you can also try MyCocktail online.

MyCocktail web site: <http://www.ict-romulus.eu/web/mycocktail>

MyCocktail Mashup Builder online: <http://www.ict-romulus.eu/MyCocktail>

MyCocktail Page Editor online: <http://www.ict-romulus.eu/MyCocktail/editor.html>

MyCocktail Demo: <http://www.ict-romulus.eu/MyCocktail/demo.html>

## 5 References

[FckGURL] FCKEditor Users Guide, [http://docs.fckeditor.net/FCKeditor\\_2.x/Users\\_Guide](http://docs.fckeditor.net/FCKeditor_2.x/Users_Guide)

[SmiURL] Smile Widgets, <http://www.simile-widgets.org/>

[TimURL] Timeline Smile Widget, <http://www.simile-widgets.org/timeline/>

[WADLURL] Web Application Description Language definition page, <http://wadl.dev.java.net/>