



Ionize for Webdesigner

Ionize 0.93 documentation

Contents

General information	5
License	5
Requirements	6
Basic requirements	6
Additional requirements	6
Terms	6
Ionize philosophy	7
Installation steps	9
Troubles after installation	9
Server internal error	9
Settings saving errors, no thumbnail creation...	9
Quick initialization guide	10
Creating a theme	10
Configure Ionize	10
Customize the website language	10
Manual configuration	10
Writing views	11
Content organization in Ionize	11
Languages	11
Pages	11
Creating one page	11
Ordering pages	11
Articles	11
Media	12
Attach media to a page or article	12
Edit one media meta data	12
Static translations	12
Creating a static translation	13
URL	13
Language detection	13
How will the website be displayed by Ionize ?	13
Content displaying	13
Standard URL	13
Reserved URI	14
Navigation	15
Pages levels	15
Page visibility in the navigation menu	15
Themes & views	16
Create one theme	16
Anatomy of a view	16
Create views	17
Quick steps	17
Tags and template language	18

Anatomy of a tag	18
Tags attributes	18
Single data tag	18
Containers tag / Iterator	18
Forms	20
Setting up a form	20
Anatomy of a form view	20
The form processing controller	21
Tags index	23
Global tags	23
translation	23
current_lang	24
files_path	24
theme	24
google_analytics	24
meta_keywords	25
meta_description	25
meta_title	25
site_title	26
base_url	26
theme_url	26
partial	27
setting	27
Languages tags	28
languages	28
languages > code	28
languages > name	28
Navigation tags	28
navigation	28
navigation > active_class	29
navigation > url	29
navigation > title	29
navigation > subtitle	29
tree_navigation	30
Page tags	30
name	30
title	30
subtitle	31
Articles tags	31
articles	31
articles > article	32
article > title	32
article > subtitle	33
article > id_article	33
article > name	33
article > view	34
article > author	34
article > author_email	34

article > date	34
article > content	35
article > url	35
article > link	35
Medias tags	36
medias	36
medias > id_media	36
medias > title	36
medias > link	36
medias > alt	37
medias > base_path	37
medias > file_name	37
medias > description	37
medias > copyright	37
medias > src	37
medias > size	37
Categories tags	38
categories	38
categories > name	38
categories > title	39
categories > url	39
categories > active_class	39
Archives tags	39
archives	39
archives > name	40
archives > title	40
archives > url	40
archives > active_class	40
archives > nb	40
Pagination tags	40
pagination	41
Forms tags	42
form_error	42
form_error_class	42
validation_errors	42
validation_errors_message	43
set_value	43
set_select	43
set_checkbox	44
set_radio	44

General information

Ionize is a content management system (CMS) based on the PHP CodeIgniter framework. The main purpose of Ionize is to make website creation and content editing very easy for the web designer and for the content editor.

License

This software is released under the Open Source MIT license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Requirements

Basic requirements

- PHP 5.2.0 or higher,
- MySQL,
- FTP access to your server.

Additional requirements

The PHP user also needs to have writing privileges on following folders:

Folder	Description
<code>/application/config/</code>	The config files folder
<code>/files/</code>	User's media folders. Writing rights must also be set to each subfolder
<code>/themes/your_theme/config</code>	The theme config folder
<code>/themes/your_theme/language</code>	Static translations files folder

Terms

Some terms commonly used in this documentation.

Ionize	Means the backend software. Of course Ionize ensure also the display of the website, but to make it simple and understandable, let's call "Ionize" the administration panel.
Website	Means the front-end, or the website as it is displayed to the visitor
View	One page with HTML and Ionize tags.

Ionize philosophy

Website

In Ionize, the website is composed of :

- Content, stored in database (pages, articles)
- One theme, containing display items
- Media files : pictures, videos, music, etc.

Ionize allow the management of one website by installation (one folder). Several themes can be used, but only one will be displayed.

Theme

A theme defines how the content of a website will be displayed.

A theme is a folder containing all the website views, assets and optional widgets used by the website. Themes are located in the **themes** folder.

View

A view is one file used to display data. This could also be called a “template”, but Ionize uses the CodeIgniter “view” terminology.

Common views are articles and page templates.

Their extension is .php, even they don't use PHP but a template tag language.

Tag

A tag is a template language element used to display data easily.

Tags are used in views.

All tags start with the “**ion**” prefix.

Example of tag :

```
<ion:articles num="3" paragraph="1" />
```

This tag will display 3 articles, with content limited to the first paragraph.

Installation steps

1. Copy the ionize zip package content to your server
2. Launch <http://your-domain/install> and follow the installation steps
3. Delete the /install folder after the installation is complete in order to use Ionize

Important : During the install process, you will create the admin account. This account will be mandatory to log onto the admin part of Ionize.

Troubles after installation

Server internal error

Depending on your server configuration, you should edit the .htaccess file and remove the comment (#) before the line :

```
#RewriteBase /
```

Settings saving errors, no thumbnail creation...

Context : When you save settings or add some media to a content, an error message appears.

Solution : The server PHP user must have writing privileges on folders / files:

- /application/config/
- /themes/*
- /files/*

Quick initialization guide

That's it : You just installed Ionize and ask you : " And now, what should I do to create my website? ". This chapter will help you.

Creating a website with Ionize consist of :

1. Creating a theme folder
2. Configure Ionize
3. Start writing your views (or so called page / articles views)

It's very easy and does not need any programming skills. Of course, if you have programming knowledges, you can go further and adapt the Ionize core to your needs, but that's not the topic here (see the Developer Guide to know more about advanced technics)

Creating a theme

Have a look at [Create one theme](#) : Follow the described theme structure.

Set the website theme : **Menu : Settings > Theme**

Configure Ionize

Customize the website language

Menu : Settings > Languages

You can change the default language code and name (language code accept up to 3 chars), or create another language.

The screenshot displays the 'Languages' configuration page. At the top right, there is a green 'Save' button and a 'Hide Options' link. Below this, the 'Existing languages' section contains a table with one entry for 'en' (English). The table has columns for Code, Name, Online, and Default. The 'en' entry has 'en' in the Code field, 'english' in the Name field, a checked 'Online' checkbox, and a blue circular icon in the Default field. To the right of the table is a form to 'Add one language'. This form has input fields for Code and Name, an unchecked 'Online' checkbox, and a green 'Save' button.

Existing languages			
Code	en		✖
Name	english		
Online	<input checked="" type="checkbox"/>		
Default			

▼ Add one language

Code

Name

Online

Manual configuration

Ionize has its own configuration file :

/application/config/ionize.php

These configuration values can be overwritten for each theme with the file :

/themes/your_theme/config/config.php

Writing views

Have a look at : [Create views > quick step](#)

Content organization in Ionize

Ionize organizes content by language, pages and articles.

The entry point is the language, which will be detected when the user enter the website.

If no language was found, the default one will be loaded (**en/english** on a default installation).

Languages

Everything can be translated, from dynamic data stored in database to so called "static" elements directly displayed in templates (such as company footers address).

Pages

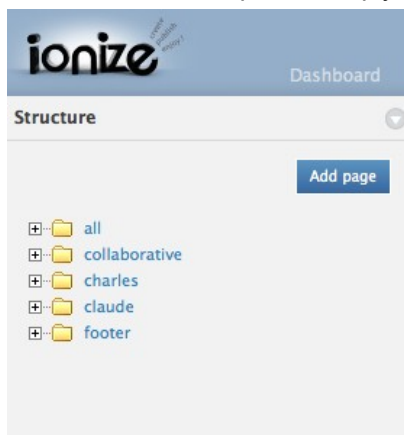
Pages organize the content. They are used to build the navigation.

Pages can be nested, to create a multi-level navigation.

Pages are also containers used to display data or simply links to internal content or external URLs.

Creating one page

In the left Structure panel, simply click "Add page" and enter the informations of your new page.



Ordering pages

Click and hold mouse down on one structure folder to move it up or bottom.

Articles

Articles contains the textual content.

An article can be a simple one, a blog post with publication dates or an images gallery, depending on the way it is displayed.

Media

By media, we mean pictures, videos, files, music, everything which can be linked to an article or inserted in an article as inline content.

The media linked to a content have metadata stored in the database, which have not media directly added to the text of an article.

The main purpose of storing media information in database is to give the user the possibility to create advanced display solution, like pictures galleries or media player directly managed by the views.

Attach media to a page or article

In one article, simply go to the tab “Images” and click on the button “Add media” :

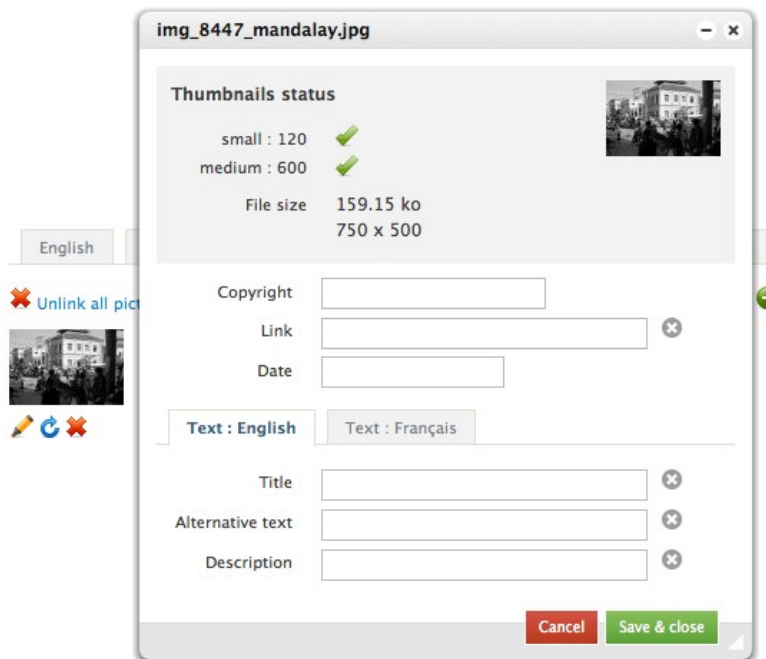


Edit one media meta data

Once one media is attached to a page or a article, you can add some meta data to it.

Once written, these meta data will be linked to the media and if you attach the same media to another article or page, you will find the same data.

In other words, once set, you don't have to set again the media meta data.



Static translations

“Static translations” are textual elements which are not content as pages or articles, but just little texts positioned in a page.

For example, the text “*website created by Partikule*” at the bottom of each website page can of course be written “hardly” in the view, but it could be nice to make it available for each language of the website. So if the user visit the website in french, this text will become “*Site conçu par Partikule*”, which is nicer for french readers.

Creating a static translation

1. Simply add the tag `<ion:translation term="your_term" />` in one of your view.
2. Go to the Ionize menu : Content > Translations
3. You will find your term and be able to write a value for each installed language

URL

Ionize produces user-friendly URLs, containing (or not) the language code.

Example of an URL produced by Ionize :

With language code : http://www.my_domain.com/en/my_page

Without language code : http://www.my_domain.com/my_page

Language detection

Language is detected through the following steps. At each step, if a language is found, detection stops.

1. Does the URL contains the language code ?
2. Is each of the browser (taken by ordering) language present in the website language definition config file ?
3. If no language is found, Ionize uses the default website language

How will the website be displayed by Ionize ?

Content displaying

The content (page / articles) will be displayed through “views”.

A view is a PHP file containing HTML and Ionize tags. Ionize tags are used to display the database stored content. See [Tags and template](#) languages for more about tags.

Views can be linked to pages and articles.

Standard URL

The URL will define what will be displayed.

Ionize displays pages, and articles into each page.

The language code is a part of the URL. It is still possible to create a one language website and not to display the language code in the URL. This choice will be achieve by views, which will not integrate the lang code in the URL used by links.

That means you can have a one language website which will not show any language URL code to the visitor. The visitor will think the website is monolingual.

Example of URL	What will be displayed ?
www.my_domain.com/	The first page of the website, with the detected languages. All produced links in this page will include the lang code in their URL if more than one language is defined or if the webdesigner has chosen to show the lang code.
www.my_domain.com/en/services	The page named services , in english. All articles in this page will be displayed as defined by the view linked to the page named "services".
www.my_domain.com/services/web-design	The article called web-design in the page services . As no language is defined, the detected one or the default website language will be used.

Reserved URI

Some URI display content in a special way. They're called "reserved URI".

They are defined in the config file : **/config/ionize.php**

The designer can, for each functionality, define his own URI. These URI must not be used as page names.

Existing functionalities

Functionality	Function (internal)	URI choosen by designer	Example URL	What will be displayed ?
Pagination	pagination	page	my_domain.tld/my-page/page/xxx	The articles list from pagination strating at the index xxx in the page my-page
Articles by category	category	cat	my_domain.tld/my-page/cat/fun	The articles list published from the fun category in the page my-page
Articles by archives	archives	archive	my_domain.tld/my-page/archive/2009	The articles from 2009

Navigation

Each page is a navigation node by default. That means that each page name can appear in the navigation menu, depending on which kind of menu you implement.

They are 2 tags provided by Ionize to display navigation :

- `navigation` : Provide one level navigation.
- `tree_navigation` : Displays the complete tree of page nodes.

Pages levels

A page has also a level in the tree of your website. This level will help us building our navigation menu.

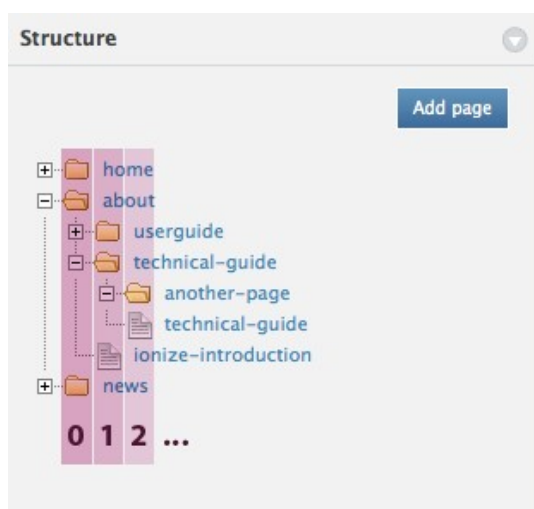


Illustration 1: Pages levels

Page visibility in the navigation menu

By default, a page will be displayed in the navigation menu. For some special usage, you could decide to hide a portion of the menu (not to set it offline, but just to hide)

This is done by unchecking the “appears” property of the page in Ionize.

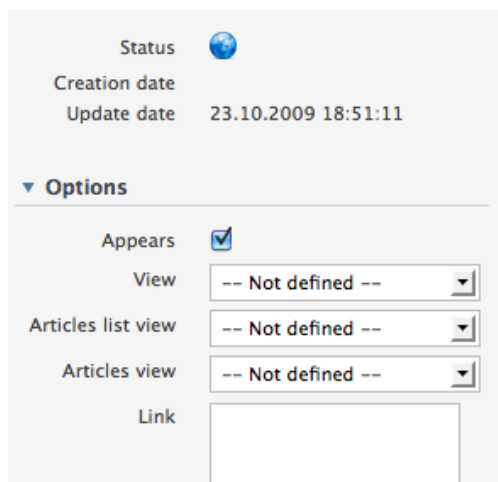


Illustration 2: Pages options : Appears is checked, this page will be present as a node in the navigation menu

Themes & views

All themes are located in the */themes/* folder.

Create one theme

Creating one theme is quite simple. You can copy one of the existing theme folder and start your work, or, if you want to start from scratch, create one folder in the *themes/* folder and include at least the following folders :

Important : Never delete the */themes/admin* folder. This theme is used by the back-end of Ionize. Without this theme, you will not be able to manage your website content.

Folder	Description
<i>my_theme/</i>	Base folder of one theme. In this case, the theme is <i>my_theme</i>
<i>assets/</i>	Theme assets, like CSS and CSS images. Can be organized as you wish or named differently. If you want to use the HTML template functionality of tinyMCE (the integrated text editor), keep this folder named like this.
<i>config/</i>	Theme configuration files. config.php : Overwrites the standard config items defined in <i>/application/config/ionize.php</i> configuration file. views.php : Contains the link between one physical template file and the Ionize displaying name in the admin part. This file is updated by the back-end, so there is no need to edit it.
<i>language/</i>	Contains the languages translation files, grouped by language code folder. language/xx/my_theme_lang.php : One language “static” language elements translation file. Generated by the admin part of Ionize. See “Static languages in views” for more information about these elements.
<i>views/</i>	Contains views of the current theme. You can organize this folder content as you wish. The example contains sub-folders. This is not mandatory and the most important is to declare your views as pages or articles views.
<i>widgets/</i>	All the widgets. See Developer : Widget creation for more information about widgets.

Anatomy of a view

A view displays the website content through tags. We can also say that tags are used in views to display content from the database.

Tags are coded in libraries called “Tag Managers”. You will certainly not need to edit these libraries, but if you wish to know more about tags, like how to create your own tags, have a look at the “Ionize Developer Documentation”.

Pages and articles are displayed according to their defined views. In a view, some tags can use other views to display data.

For example, the “articles” tag has a “view” attribute to force a view to be used instead the view linked through Ionize.

Create views

Quick steps

1. Create the view PHP file in the `/themes/<theme_name>/views/` folder
2. Declare the created view as “page view” or “article view”
This step is mandatory for pages and articles view but has not to be done for partial views (like header or footer for example)
3. Use it in Ionize when editing page or article / Use it in tags.

1. Creating the view file

To create a view, simply create a new PHP file in the `/themes/<your_theme>/views` folder or in a sub folder of this folder.

2. Declare the view

To use this view to display pages or articles, you have to declare this view through Ionize :

1. Log on the admin panel and go to **Settings** → **Theme...**
2. Click on **Current theme views list** : You will see all views from your theme
3. In your view “Logical name” field, put the name you want to see appearing and select one type of view and save.
4. Check in one article / page : the view name will be available in the view list of the item.

▼ Current theme views list : Hand_made



File	Folder	Logical name	Type
article_blog	articles/	<input type="text" value="Blog Article"/>	<input type="text" value="Article"/>
article_blog_list	articles/	<input type="text" value="Blog List Article"/>	<input type="text" value="Article"/>
intro_article	articles/	<input type="text"/>	<input type="text" value="-- No type --"/>
last_news	articles/	<input type="text"/>	<input type="text" value="-- No type --"/>

Illustration 3: View declaration in admin panel

Note : Some views do not have any logical name : It's normal since they are not used by articles or pages but are used directly by tags.

For example : The “**last_news**” views is not declared and does not have to be as it is used by the tag “article” to show the last news in a specific block.

Tags and template language

Tags are used in views to display the content of the website. They can as well display simple data than a string or nested data.

They are 2 kind of tags :

- single data tags
- containers tags

Anatomy of a tag

An tag has :

- A prefix : ion
- A name
- Some attributes, or not, depending on the tag

```
<ion:tag_name attribute1="attribute_value" />
```

Tags attributes

Tags attributes let you go deeper in displaying data by setting some filter, getting data from other pages and such kind of stuff.

Tags attributes can have several values :

- **Boolean (true / false)** : when the attribute is omitted, the value is considered as false
- **String** : A simple string
- **Mixed** : A string containing more than one information. This kind of data depend on the tag and you need to refer to the tag index to understand them.

Single data tag

A single data tag is standalone. That means it doesn't have any children. It self "contains" the data. Calling such a tag is quite simple.

```
<!-- This will display the website keywords or the page keywords -->
<ion:meta_keywords />
```

Some tags can have the enclosed wished HTML tag as parameter.

This permit to use them as conditional tags : If the tag contains a value, this value will be returned enclosed by the given tag, if no value is set, nothing will be returned.

Containers tag / Iterator

A container tag, or iterator, contains child data tags. Some containers needs to be displayed through views while others can be directly outputted to the view page.

```
<!-- Articles : First way of displaying articles content -->
<ion:articles num="3">
  <ion:title />
  <ion:content paragraph="1" />
</ion:article >

<!-- Articles : Container with a specific view
The base folder for the views is my_theme/views/
```

In this example, the wished view is in the "articles" folder, so we need to give the path to this folder.

The tag will get the asked view, set it to each article and render the result.

-->

```
<ion:articles num="3" paragraph="1" view="articles/last_articles" />
```

The above example will display the last three articles with the view "***last_articles***".

As you can see in the view attribute, we didn't put the base folder for the view. The folder `themes/your_theme/views` is considered as the base folder for the tag.

The file ***themes/your_theme/views/articles/last_articles.php*** must exist.

Example of content of the ***last_articles.php*** file :

```
<!-- my_last_article.php content : display each article
      of the "articles" parent object
-->
<div class="result">
  <!-- This can be understood as : "foreach article" -->
  <ion:articles>
    <!-- display the article title and use the URL for the link -->
    <h3><a href="<ion:url />"><ion:title /></a></h3>

    <!-- display the article content
          In this case, as we define the paragraph attribute to "1" in the
          iterator, it will only display the first paragraph of the article.
          It has been possible to write directly the paragraph limitation in the
          "content" tag :
          <ion:content paragraph="1" />
    -->
    <ion:content />

  <!--Close the "foreach article" tag -->
</ion:articles>
</div>
```

Forms

Ionize provides a complete set of tags to write forms in your views. These tags are very close to the CodeIgniter form validation class and form helper.

A form can be build in a page or in a article view.

Important

Because the form data processing can be very different from one form to another (store data in database, send mails or not, etc.) each form is processed through a dedicated controller.

Ionize doesn't provide a form builder, but allows you to manage the form data as you wish.

They are 2 main ways for building forms in views :

1. Use the Ionize Form tags (see [form tags](#) for more information)
 2. Include your PHP form view as partial view in one of your page or article view (see [partial tag](#) for more information).
- So it is possible to use pure PHP and the CodeIgniter form approach to manager your forms.

Setting up a form

1. Create a form dedicated view (can be an article or a page) and store it in your */themes/your_theme/views* folder or any subfolder of your view folder.
2. Create a form controller in the */application/controller/* folder. This controller should extend *Base_controller*
3. Modify your */application/config/routes.php* file and add your route to the form processing controller.

Anatomy of a form view

In this example, we suppose you created your form in an article view using the Ionize tags.

For example using the standard CodeIgniter form approach, refer to the CodeIgniter documentation.

```
<!-- The form error message : displayed when a validation error occurs -->
<ion:validation_errors_message tag="div" class="errorMsg" term="form_contact_error" />

<!-- This form will be processed by /application/controller/form_process.php -->
<form action="<ion:base_url />form_process" method="post" name="myForm">

    <!-- This message will be displayed if a validation error occurs
         during the field validation process
    -->
    <ion:form_error input="username" tag="span" class="errorMsg" />

    <label>Name</label>
```

```

<!-- The value will be feeded again in case of error -->
<input type="text" name="username" value="<ion:set_value input="username" />" />

<ion:form_error input="email" tag="span" class="errorMsg" />
<label for="email" >Email</label>
<input type="text" name="email" value="<ion:set_value input="email" />" />

<input type="submit" value="<ion:translation term="form_send" />" />
</form>

```

The form processing controller

Located in /application/controllers/

```

class Form_accredit_compiegne extends Base_Controller
{
    /**
     * Constructor
     */
    function __construct()
    {
        parent::__construct();

        // FTL Tag library
        require_once APPPATH.'libraries/ftl/parser.php';
        require_once APPPATH.'libraries/ftl/arraycontext.php';

        // Form tag manager
        require_once APPPATH.'libraries/Tagmanager.php';
        require_once APPPATH.'libraries/Tagmanager/Form.php';

        // FTL tags Context
        $this->context = new FTL_ArrayContext();

        $f = new TagManager_Form($this);
        $f->add_globals($this->context);
        $f->add_tags($this->context);
    }

    /**
     * Validates the contact form
     */
    function index()
    {
        $this->load->helper(array('form', 'url'));
        $this->load->library('form_validation');
    }
}

```

```

$this->form_validation->set_rules('name', lang('name'), 'required');

// Delimiters for individual error messages :
// Put to nothing, task done by the FTL tagmanager
$this->form_validation->set_error_delimiters('', '');

// FAILS
if ($this->form_validation->run() == FALSE)
{
    // If request comes from XHR, send just the form again
    if ($this->is_xhr() === true)
    {
        $this->render('forms/my_form', $this->context);
    }

    // Else, ensure the validation data are kept and
    // redirect to the referring page
    else
    {
        // Put the validation_errors string message
        // to the flash session data
        $this->session->set_flashdata('validation_errors', $this-
>form_validation->error_string());

        // Put the form field data array to the flash session data
        $this->session->set_flashdata('field_data', $this-
>form_validation->_field_data);

        redirect($_SERVER['HTTP_REFERER']);
    }
}

// SUCCESS
else
{
    // Redirect to the success message page
    redirect('success_message_page_name');
}
}

```

Tags index

Tags usage :

```
<ion:parent_object_name attribute="my_attribute">
  <ion:object_name />
</ion:parent_object_name>
```

Pre defined tags are available for:

- Website data
- Languages
- Navigation
- Current page data
- Articles
- Medias
- Articles by category, articles by archive period
- Pagination
- Form Validation

Global tags

Website basic data, like website name, meta_keywords, etc.

These tags don't have any attributes.

Example of usage :

```
<ion:site_title />
```

translation

Returns

The static term, in the current language.

Static translation terms are elements which are not content as pages or articles, but just little texts which can be present in a page.

For example, the text “*website created by Partikule*” at the bottom of each website page can of course be written “hardly” in the view, but it could be nice to make it available for each language of the website. So if the user visit the website in french, this text will become “*Site conçu par Partikule*”, which is nicer for french readers.

Attributes

term Mandatory.
The index name of the wished term.

Usage

```
<ion:translation term="your_term" />
```

current_lang

Returns

The current lang code, on 2 or 3 chars.

Attributes

none

Usage

```
<ion:current_lang />
```

files_path

Returns

User's medias base folder, as set in Ionize Advanced settings

Attributes

none

Usage

```
<ion:files_path />
```

theme

Returns

The current theme name. Is also the theme folder.

Attributes

none

Usage

```
<ion:theme />
```

google_analytics

Returns

Google analytics script as defined in the Ionizes Advanced settings

Attributes

none

Usage

```
<ion:google_analytics />
```

meta_keywords

Returns

Website current language Meta Keywords or current page Meta Keywords if they are set a the current visited page.

Attributes

none

Usage

```
<ion:meta_keywords />
```

meta_description

Returns

Website current language Meta Description or current page Meta Description if it is set a the current visited page.

Attributes

none

Usage

```
<ion:meta_description />
```

meta_title

Returns

1. The current displayed **article meta_title** value if not empty
2. else the **page meta_title** if not empty,
3. else the current page **title** value.

This tag is usefull for the HTML <title> tag feeding : it gives the title of the browsers window.

Usage

Example of header usage

```
<html>
<head>
  <title><ion:site_title/> - <ion:meta_title/></title>
</head>
...
```

As this tag also support enclosure, it can like an enclosed conditional tag :

```
<body>
<!-- This will return the tag value enclosed by H2 tag if a value exist or nothing if no
value
-->
<ion:meta_title tag="h2" />
```

site_title

Returns

The website title as set in the Ionize settings

Attributes

none

Usage

```
<ion:site_title />
```

base_url

Returns

The website base URL, with or without the lang URL.

base_url will not try to find if several languages are online and display the lang code in the URL by itself. You need to set the attribute "lang" to true if you want the tag to try to find out if it has or not to display the lang URL.

If you want to force the lang to appear in the URL, even only one language is existing, set "force_lang" to true.

Attributes

lang true / false
Set to true if you want the tag to try to find out if several languages are currently online and prints out the base URL including the lang code.
If only one language is online and you want the lang code to appear in the URL, you need to use the "force_lang" attribute.

force_lang true / false
Set to true, it will add the language code to the URL, even there is only one language defined or online for the website. Can be used without the lang attribute.

Usage

```
<ion:base_url [ lang="true/false" force_lang="true/false" ] />
```

theme_url

Returns

The current theme complete URL, with trailing slash. Usefull for accessing assets (CSS, images)
Examples of return : `http://your_domain.tld/themes/your_theme/`

Attributes

none

Usage

```
<ion:theme_url />
```

partial

Returns

The result of a view used as partial.

This view can contains other tags, but can also be a pure PHP view. As views cannot mix tags and PHP, this is the only way to add PHP to views.

Attributes

path Mandatory.
The path to the view, relative to the *themes/your_theme/views* folder.

php true / false
false if not set.
If set to true, force the load of the view as pure PHP view. If not set, it will return the view parsed as it doesn't contains PHP but tags.

Usage

```
<ion:partial path="default/header" php="false" />
```

setting

Returns

A website settings item, from the setting database table. All main and usable settings have their dedicated tag, so usage of this tag is normally not necessary.

Attributes

item Mandatory. The wished setting name

Usage

```
<ion:setting item="item_name" />
```

Languages tags

These tags are used to build the languages navigation menu.

languages

Languages is a container tag, an iterator.

Childrens

code, name

Attributes

None

Usage

```
<ion:languages>  
  <ion:code /><ion:name />  
</ion:languages>
```

languages > code

Returns

The lang code

Attributes

none

languages > name

Returns

The lang name

Attributes

none

Navigation tags

navigation

Container tag for navigation items.

Children tags : url, active_class, title, subtitle1, subtitle2

Attributes

active_class	Set the CSS class name to use to highlight the current menu item. For example, if “ <i>active</i> ” is used, the children tag active_class will just contains the string active on the current menu item selected by the visitor.
level	The wished level to display. “0” is the top menu. Using level “1” will display the child items from the current level 0 page.
view	Path to the view to use to display the navigation menu.

Usage

```
<ion:navigation level="0" active_class="active">
  <li>
    <a class="<ion:active_class />" href="<ion:url />"><ion:title /></a>
  </li>
</ion:navigation>
```

Usage with use of a nested view :

```
<ion:navigation level="0" active_class="active" view="navigation.php" />
```

In the above example, the used view is “navigation.php”. This file should contains the HTML block :

```
<li>
  <a class="<ion:active_class />" href="<ion:url [lang="true"] />"><ion:title /></a>
</li>
```

navigation > active_class

Returns

The string set in the attribute **active_class** of the navigation tag, only if the menu item name is the same as the current visited page.

navigation > url

Returns

The URL to the corresponding page, with or without lang code.

This tag will try to find if more than one languages are online for the website. If more than one languages are online, it will include the lang code in the URL to the page.

If the page has a special link, this link will be analyzed before trying to get the standard URL.

Attributes

lang If set to true, will force the URL to include the lang code.
If a special link is set, it will add the lang code only on an internal link.

navigation > title

Returns the title of the page.

navigation > subtitle

Returns the subtitle 1 and 2 of the page.

tree_navigation

Returns

A tree based navigation menu, as a unordered HTML list (...).

Caution : This tree navigation menu is based on a helper : /application/helpers/navigation_helper.php
If you wish to create your own, just copy this file to : /themes/your_theme/helpers/
The tag will automatically take your helper instead the default one.

Attributes

level The starting wished display level. "0" is the first level.
depth Depth you want your menu to go.
"2" means 2 levels, including the starting one. So
lang Do the menu item include the lang URL.
Forced for the moment. Means if you set it to true, it will include the lang URL even there is only one language set for the website.
active_class The class name you wish to use for the current menu items.

Usage

```
<ion:tree_navigation level="0" depth="2" helper="navigation:get_nested_navigation"  
lang_url="true" active_class="active" />
```

Page tags

These data displays data from the current displayed page. These tags are used in the same way than the global tags : without any enclosing tag.

name

Returns

The page name

Usage

```
<ion:name />
```

title

Returns

The page title, depending on the current language

Attributes

tag Optional enclosing tag.
If set, this will returns the title enclosed by the given tag, or nothing if no title exists for the page. This avoid having empty title tags.

Usage

```
<!-- This will return the page title -->
<ion:title/>

<!-- This will return the page title, with enclosing tags
returns : <h2>Your title</h2>
or nothing if there is no value set for the title.
-->
<ion:title tag="h2"/>
```

subtitle

This tag is working like the <ion:title /> tag.

Articles tags

The articles tag can be used as an iterator or as a container for the children article tag.

articles

Container / Iterator tag.

Attributes

filter	Logical filtering of the articles on their base data. Base data are : title, type, author
from	On or more page name, separated by coma Returns all articles from the given pages name.
from _categories	Returns articles from given categories names. Categories names must be separated by comma
from _categories _condition	and / or Conditional inclusion / exclusion of categories names.
num	Number of article to return. Not compatible with use of pagination.
order_by	SQL like order by clause Example : 'date DESC' will order the articles by date descending
pagination	true / false Use of pagination. If pagination tag is used to show the pagination links, this attribute must be set to use pagination in this set of articles.
paragraph	Number. Limit the articles content display to the defined number of paragraph.
scope	global / parent The global scope will return articles from all website. Parent scope will return articles from curent parent page (all child pages articles).
view	View path to the view to use. Ionize permit to define the wished view, but if the view is set in one tag, it will force articles to use this view. The path is relative to the themes/your_theme/views/ folder

Usage

With each article displayed through the defined view. In this case, the tag `<ion:article>` will use the Ionize defined view for the article :

```
<ion:articles [ filter="title:='" from="page_name" from_categories="category1,category2"
from_categories_condition="and/or" num="3" order_by="date DESC" pagination="true"
paragraph="2" scope="global" view="path_to_view" ] >
  <ion:article />
</ion:articles>
```

Directly in the page view :

```
<ion:articles [ filter="title:='" from="page_name" from_categories="category1,category2"
from_categories_condition="and/or" num="3" order_by="date DESC" pagination="true"
paragraph="2" scope="global" view="path_to_view"] >
    <h2><ion:title /></h2>
    <ion:content />
</ion:articles>
```

articles > article

Container for one article tags.

This tag is commonly used to use a specific view as partial to display the articles content.

If you only wish to display a list of articles within your page view, it is not mandatory to use this tag.

Attributes

type	Name of wished type Returns only articles with this type.
view	Force the view to the this attribute view. The path is relative to the themes/your_theme/views/ folder
paragraph	Limit the content to the defined number of paragraph

Usage

See the tag `<ion:articles />` above.

article > title

This tag as enclosure possibility.

Can be used nested to the `<ion:articles >` tag or the `<ion:article >` tag.

Returns

The article name

Attributes

tag	The HTML tag to use to enclose the title.
-----	---

Usage

```
<!-- Not enclosed : Return <h2></h2> if the article title is empty -->
<ion:title/>

<!-- Return the article title, with enclosing tags or nothing if the title is empty. -->
<ion:title tag="h2"/>
```

article > subtitle

This tag can be as much values as the number of defined languages.

Can be used nested to the `<ion:articles >` tag or the `<ion:article >` tag.

Return

The current article subtitle, in the current language.

Attributes

tag The HTML tag to use to enclose the subtitle.

Usage

```
<!-- Not enclosed : Will return <h2></h2> if the tag value is empty -->
<h2><ion:subtitle /></h2>

<!-- Conditional enclosing : Will return nothing if the tag value is empty -->
<ion:subtitle tag="h2" />
```

article > id_article

Return

The current article ID.

Can be useful to name each image gallery of each article if multiple articles are displayed on the same page.

Usage

```
<!-- Returns the current article ID.
     Example : 3
-->
<ion:article_id/>
```

article > name

Return

The current article name.

Usage

```
<ion:name />
```

article > view

Return

The current article view path, relative to the */themes/your_theme/views* folder.

Usage

```
<!-- Example of return : default/last_article -->
<ion:view/>
```

article > author

Return

The article author name.

Attributes

tag HTML tag to use to enclose the content.

Usage

```
<!-- Example of return : martin -->
<ion:author/>
```

article > author_email

Return

The author email

Usage

```
<!-- Example of return : emile@youpi.com -->
<ion:author_email />
```

article > date

Return

Article date, based on defined Ionize date.

If no date is defined in Ionize, this tag will return the article creation date. Else, it will return the publication date.

If the format attribute is not set, returns the date with the format : **Y-m-d H:i:s** .

Example of standard return : **2010-03-29 20:14:55**

If one of the following format is used, the corresponding translation will be searched in the **/themes/your_theme/language/xx/date_lang.php file** :

D Name of the day
I
F Name of the month
M Name of the month, long

That means the day and month names will be displayed in the according language.

Attributes

format PHP output format.
 If set, returns the date using this format string.
 See PHP doc for supported date formats : <http://www.php.net/date/>

tag The HTML tag to use to enclose the date value.

Usage

```
<ion:date format="D" />
```

article > content

Return

The current article text content.

Can be enclosed in the given tag.

Attributes

paragraph Limit the content to the defined number of paragraph.

tag HTML tag to use to enclose the content.

article > url

Return

The URL to the article.

Will return the URL in the format : ***http://your_domain/page_name/article_name***

Note : If the link value of the article is set in Ionize, this tag will return the link instead of the standard URL to the article.

Important : If more than one language are defined and set to “online” in Ionize, the returned URL will automatically include the current language code.

In edition mode : If you're connected to Ionize as an editor, the URL will include the language code if more than one languages are set. The offline languages code are also included.

Attributes

lang true / false

If set to true, the returned URL will include the language code, even there is only one language defined in Ionize.

This can be useful is you plan to translate further the website and don't want the URL to be changed when you put the new language online.

article > link

Return

The current article link set through Ionize.

If the link is internal to the website, it will try to find if more than one languages are online and add the language code to the url.

Attributes

lang true / false

If set to true, the returned URL will include the language code, even there is only one language defined in Ionize.

This can be useful is you plan to translate further the website and don't want the URL to be changed when you put the new language online.

Medias tags

These media tags are available in containers :

- page
- articles
- article

medias

Container for all medias attached to the page or the article.

Attributes

type picture, music, video, file
 Mandatory.
 Indicates the media type to return.

Usage

```
<ion:medias type="picture">  
  <img title="<ion:title />" alt="<ion:alt />" src="<ion:src />" />  
</ion:medias>
```

medias > id_media

Returns

The media ID.

medias > title

Returns

The media title, using the current language.

medias > link

Returns

The media link. This is not the link to the media but the link set to the media through Ionize.

medias > alt

Returns

The media alternative text, using the current language.

medias > base_path

Returns

The path to the current media folder. This path does not include the file name !

medias > file_name

Returns

The media file name

medias > description

Returns

The media description, using the current language.

medias > copyright

Returns

The media file copyright.

medias > src

Returns

The complete path to the media, including the media file name.

If you create thumbnails with ionize, this tag can help you getting a given thumbnail complete path.

Attributes

folder thumbnail folder name.
 If set, returns the complete path to the thumbnail having this folder in its Ionize definition.
 Example : You created one thumbnail and set the folder to "medium".
 To get this picture thumbnail, set this attribute to "medium".

Usage

```
<ion:medias type="picture">  
  " />  
</ion:medias >
```

medias > size

Returns

The picture size.

Note : Can only be used with medias of type "picture". This tag will return nothing with other type of medias.

Attributes

folder	Folder name of the thumbnail wished size. If set, will try to get the according thumbnail size. If not set, will return the big picture size.
dim	height / width Mandatory attribute. Return the asked dim.

```
<ion:medias type="picture">  
  " />  
</ion:medias >
```

Categories tags

Useful to build an articles categories menu.

The page called by each item of the categories tag is the current page.

The URL will change, indicating we are in a category filtering.

If pagination is used, the pagination link will also change.

categories

Containers for categories.

Attributes

active_class	Name of the CSS class to use for the current selected category. Set to "active" if not set.
view	Path to the view to use, relative to the /themes/your_theme/views folder.
from	Page name from which get the categories. If not set, the tag will use the current page.

Usage

```
<ul>
<ion:categories active_class="active">
  <li><a class="<ion:active_class />" href="<ion:url />"><ion:title /></a></li>
</ion:categories >
</ul>
```

categories > name

Return

The category name

categories > title

Return

The category title, using the current language.

categories > url

Return

The current category URL. If more than one language are online, returns the URL with the included lang code.

If you're connected as editor, all languages will be showed when you visit the website.

Attributes

lang	true / false If set to true, force the category to use the lang code in the URL even only one language is online.
------	--

categories > active_class

Returns

The CSS class name set as attribute to the tag `<ion:categories />`.

Archives tags

Useful to build an articles archives menu.

The page called by each item of the archives tag is the current page.

The URL will change, indicating we are in an archive filtering.

If pagination is used, the pagination link will also change.

archives

Containers for archives links to a period filtered articles list.

Attributes

active_class	Name of the CSS class to use for the current selected archive. Set to "active" if not set.
view	Path to the view to use, relative to the <code>/themes/your_theme/views</code> folder.
format	Format of the period. PHP format.

Usage

```
<ul>
<ion:archives with_month="true">
  <li><a class="<ion:active_class />" href="<ion:url />"><ion:period /> - <ion:nb
/></a></li>
</ion:archives>
</ul>
```

archives > name

Return

The current archive name

archives > title

Return

The archive title, using the current language.

archives > url

Return

The current archive URL. If more than one language are online, returns the URL with the included lang code.

If you're connected as editor, all languages will be showed when you visit the website.

Attributes

lang true / false
If set to true, force the archive link to use the lang code in the URL even only one language is online.

archives > active_class

Return

The CSS class name set as attribute to the tag <ion: archives />.

archives > nb

Return

Number of articles in this period of time.

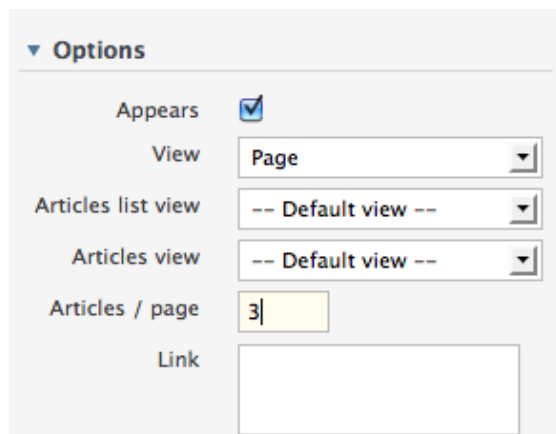
Pagination tags

The pagination tag gives the ability to filter the number of displayed articles and to display a pagination menu with the pages number.

So hundred of articles can be displayed through more than ten pages of five articles for example.

Pagination is set through the Ionize's page attribute called : **"Articles / page"**. This attribute is set in the page options panel in Ionize.

Set to 0, it means no pagination.



The screenshot shows a web interface for editing page options. It features a section titled "Options" with a dropdown arrow. Below this, there are several settings:

- "Appears" with a checked checkbox.
- "View" with a dropdown menu set to "Page".
- "Articles list view" with a dropdown menu set to "-- Default view --".
- "Articles view" with a dropdown menu set to "-- Default view --".
- "Articles / page" with a text input field containing the number "3".
- "Link" with an empty text input field.

The options panel as displayed when editing a page.

Important :

This setting goes with the tag **"pagination"** in your views, which display the menu to go from one page to another.

You need to implement this tag in your views in order to make pagination work properly.

pagination

Returns

the pagination numbered menu.

The visualization options are the same as the pagination lib of CodeIgniter, except that the open and closing tags of each element are not set : only the tag name is set.

Notice : The pagination menu elements default enclosing tags definition take place in the config file : ***/themes/your_theme/config/pagination.php***

This config file is optional.

The text of the menu items which need translation (for example “next page”) are located in the language folder of your theme :

/themes/your_theme/language/xx/pagination_lang.php

Attributes

per_page	Number of article displayed per page Overwrites the Ionize defined number of articles per page.
filter	Article's filter (see article tag). If a filter is set for articles, this attribute must have the same value.
order_by	Article's order_by attribute If this attribute is set for the articles tag, it must have the same value.
class	Name of the CSS class to use for the main enclosing tag.
full_tag	Main enclosing tag
first_tag	Enclosing tag for the “first page” link.
last_tag	Enclosing tag for the “last page” link.
cur_tag	Enclosing tag for the “current” active page number.
next_tag	Enclosing tag for the “next page” link
prev_tag	Enclosing tag for the “prev page” link
num_tag	Enclosing tag for the page number link

Usage

```
<!-- Pagination links -->
<ion:pagination filter="title:!=" full_tag="div" class="pagination" />

<ion:articles filter="title:!=" />
```

In this example, we see that if we use the article filter attribute, we need to have the same filter for the pagination tag.

Forms tags

These tags let you write forms easily.

form_error

Return

A field validation error message, enclosed by the defined tag
Equivalent to CodeIgniter's : Form Validation->form_error('field_name')

Attributes

input	The field linked to this error message
tag	Message enclosing tag
class	Message enclosing tag CSS class
id	Message enclosing tag CSS ID

Usage

```
<ion:form_error input="username" tag="p" class="errorMessage" />
```

form_error_class

Return

An individual field error class in case of error.
Useful to add a class to a label in case of error during validation, for example.

Attributes

input	The field linked to this error class
class	Message enclosing tag CSS class

Usage

```
<label for="username" class="<ion:form_error_class input="username" class="error" />">
```

In this example, the class "error" will be added to the label only if an error occurs.

validation_errors

Return

The whole errors list, surrounded by the defined tag
Equivalent of the CodeIgniter : **Form_validation->error_string()**.

Attributes

tag	Message enclosing tag
class	Message enclosing tag CSS class
id	Message enclosing tag CSS ID

Usage

```
<ion:validation_errors tag="div" class="errorMessage" />
```

validation_errors_message

Return

A user defined error message if `form_validation->error_string()` is not empty.

Important : Does not return the `error_string()` content

Equivalent to CodeIgniter : **`Form_validation->validation_errors()`**.

Attributes

term	The translation term to use
tag	Message enclosing tag
class	Message enclosing tag CSS class
id	Message enclosing tag CSS ID

Usage

```
<ion:validation_errors_message tag="div" class="error" term="form_contact_error" />
```

In this case, the HTML DIV will contains the translated term value "form_contact_error" and will be displayed only if some errors occur.

set_value

Return

The repopulated form input field value

Equivalent to CodeIgniter : `Form_Validation->set_value('field name')`

Attributes

input	The input field name
-------	----------------------

Usage

```
<input type="text" name="email" value="<ion:set_value input="email" />" />
```

set_select

Return

The repopulated form select value

Equivalent to CodeIgniter : `Form_Validation->set_select('field name')`

Attributes

select	The select field name
value	The value of the select option

Usage

```
<select name="country">
  <option value="France" <ion:set_select select="country" value="France" />
  >France</option>
  <option value="UK" <ion:set_select select="country" value="UK" /> >UK</option>
</select>
```

set_checkbox

Return

The repopulated form checkbox

Equivalent to CodeIgniter : Form_Validation->set_checkbox('field name')

Attributes

checkbox	The checkbox field name
value	The value of the checkbox option

Usage

```
<input type="checkbox" name="mag[]" value="Smash" <ion:set_checkbox checkbox="mag"
value="Smash" /> />
<input type="checkbox" name="mag[]" value="AList" <ion:set_checkbox checkbox="mag"
value="AList" /> />
```

set_radio

Return

The repopulated form radio

Equivalent to CodeIgniter : Form_Validation->set_radio('field name')

Attributes

checkbox	The radio field name
value	The value of the radio option

Usage

```
<input type="radio" name="mag" value="Smash" <ion:set_radio radio="mag" value="Smash"
/> />
<input type="checkbox" name="mag" value="AList" <ion:set_radio radio="mag"
value="AList" /> />
```