

Website documentation

iTEC 2015, Timișoara

Version 1.0

Last change: 03.04.2015

Created by Mihai-Alin Eparu

1. Concept base

The entire website was designed with a single scope: to represent your

company in the best way possible in the online media. This website fits both your needs and those of your future visitors, and that’s what I think is the most important in any product.

In the simplest words, this website is fully customizable and fully responsive for all of the devices which use late technology (at least HTML5 and CSS3-compatible browsers). Although it has some browser requirements, I think that those requirements are satisfied by all the future visitors. In the [requirements](#Requirements) section you can see all of this website requirements.

Technically speaking, all of your content is dynamically shown to the user, so when the server receives a request from the visitor, it checks into the database to see if it can find that route, and if that route is found, it shows to the user what he wants. And that’s all! Also, the user will not even know what’s happening behind the scenes because this website uses AJAX to get it’s content, so all of the users’ requests will be treated asynchronously. Even more, because you’re part of an international group, you can deliver your content in multiple languages using just simple keys which you define in your administration panel, and the user can change instantly[[1]](#footnote-1) the language in they will see the pages.

1. First look over the website and the administration panel.

As I’ve said in the concept base, instead of individual, hard-coded

physical pages you have routes. The routing process takes place at two places: at the server-side and at the client-side. First, using Apache’s mod\_rewrite function, every request of type /*%word1%* or /*%word1%*/*%word2%* (in which *%word1%* represents the route and *%word2%* represents the attributes) will be redirected to the main page (index.php) and the GET parameters *route* and *attributes* will be set according to the request. Then, the main page checks those parameters and sets the Javascript variables *route* and *attributes*, which will finally be transferred to an AJAX request to another web page that will search in the database for the route and then return the content which the user requested.

Also, there are several specially-created plugins, such as Translr, Scrollspy and the statistics plugin which makes your life easier. All of those plugins are described below in this documentation.

Now, on the administration panel you can manage all of your content. And that’s the most amazing thing of this website! You now have a fully customizable, good looking website, which also supports internationalization. Based on your requests, trying to fit all of your content needs, I created 2 ready-to-go widgets (jobs and timeline) which you can update right from your administration panel! Every field on which you see the blue button with a white globe[[2]](#footnote-2) supports Translr keys, so you can select an existing key or add a new key right from the current adding/editing form, without having to refresh the page.

Next, the menu is certainly not a problem because you now can add, edit and reorder menu elements from your administration panel.

Also, you can grant administrating roles to other colleagues of yours, in case that you want multiple administrators on your website.

In the last section, but not the least, you have the statistics. You can see the basic statistics (those which are referring only to visitors, sessions and visits), but also the detailed statistics, which are referring to the users’ devices.

You can see below more information about each of the concepts presented above.

1. Understanding content

The main categories of content are: jobs, timeline elements and pages.

As I’ve said above, the available jobs are shown in a special widget that

you can append to a page (which will have the ID *jobs*), but they also have a dedicated page in which the user can see all the job details and even send an application. The fields available for a job are:

* *Title* (T) (R) : The name of the available position
* *Expire date* (R) : The date after the job will not be shown in the website. Although, the jobs are still kept in the database after this date.
* *Ref. #* (O) : The reference ID (only for your information)
* *City* (O): The city in which the job will take place
* *Applications email* (O) : The e-mail to which the applications should be sent when an user wants to apply for a job. If you don’t allow online applications, please uncheck the checkbox next to the field.
* *Content* (CT) (R) (HE) : The content which will be shown to the user.

As your request, I’ve implemented a timeline (which will have the *timeline* ID) in which you can add multiple timeline elements (events), which have the following fields:

* *Date* (R) : The date of the event (on the website only the month and year will be shown)
* *Title* (T) (R) : The name of the event.
* *Content* (T) (R) (HT) : The description of the event.

Pages are simply pages! All you must know about them is that they have

dynamically generated content and that’s why you should take care of a few things like encoding or using quotation marks. The fields which you should complete in order to create a page are:

* *Title* (T) (R) : The name of the page (it will also represent the document title at rendering)
* *Route* (R) : A string which represents the *unique* identifier of this page and which is used by the page renderer to get to the page.
* *Content* (T) (CT) (R) (HT) : The content of the page *in JSON format*. Basically, this represents the elements of the page in an array. You will find more details about this in [Understanding pages](#Understanding_pages).

1. Understanding routes

Routes are basically paths that the page renderer uses to display the

content which the user requested. Usually, a route is of type *%route\_name%*/*%route\_attributes*%, but the *%route\_attributes%* field is optional.

Basically, as I’ve explained above, the route is the path that helps the page renderer get to the content and display it to the user. The route name should be *unique*, so that the content is displayed correctly.

The fields that routes use are the following:

* *Route* (R) : An unique string which can contain letters (no casing), numbers or `\_` (underscore) which represents the route name. *No other characters are allowed!*
* *Attributes* (O) : The variables which the page renderer should receive from the user. Please put them in order of the requirement. They will be filled in the order you specify. For example, if you want to receive the parameters $type and $id for a custom PHP page you have created, you should put the string $type/$id (variables identifiers separated by slash). You can have as many attributes as you want.
* *Route to* (R) : The name of the page or the file you want to get to. As you can see, a route can map also to a file (e.g. PHP custom file). So you can go ahead and create new content which uses PHP! If you map to a PHP file, that file will receive in a GET parameter called *attr* the JSON encoded attributes array which you can later decode using json\_decode function.
* *Attributes sent* (O, mandatory if you fill in the *Attributes* field) : This is the set of parameters of type key=value, in which the value will be the attribute that you received from the user and the key will represent the attribute name in the *attr* array. For the example above, you could use the string type=$type/id=$id (also separated by slash). The parameters key names could be different from their identifier (attribute name) (e.g. t=$type/id=$id). Also, if you want to go to a specific element from that page, you can set this field to the ID of that element from the page (e.g. #timeline)

1. Understanding pages

As I’ve said before, a page is just a page! It has a title and some

elements. Although, those elements are dynamically created using a JSON encoded array. And this could be a little tricky.

First, these are the principal elements included in the *Content* array (you also have a template as a predefined value for this field) :

* *jumbotron* (R) : This represents the jumbotron element and can have multiple characteristics:
  + *type* (R) (image|video)
  + *size* (O) („small”|”default”)
  + *src* (R) (*URL*)
  + *h1* (T) (O) (*string*)
  + *divider* (O) (true|false)
  + *h2* (T) (O) (*string*)
  + *buttons* (O) (*array*)
    - *id* (R) (*string*)
      * *type* (R) („firststop”|”primary”)
      * *href* (O) (*URL*)
      * *title* (O) (*string*)
* *scrollspy* (O) („true”|”default”)
* *content* (R) (*array*)
  + *type* (R) („div”|”jobs”|”timeline”|”contact”|”newsletter”)
  + *id* (O) (*string*)
  + *classes* (O) *(string*)
  + *custom\_css* (O) (*string*)
  + *title* (T) (O) (*string*)
  + *content* (O) (*BASE64 encoded HTML string*) – you can use <http://www.base64encode.org/> for the encoding
  + *custom\_scripts* (*string*)

Please check your JSON array encoding! This can cause many unwanted

errors.

1. Understanding Translr

Basically, Translr is a script which downloads all the keys for the current

language and tries to parse in every field that is marked apropiately the value of the key. So you have 6 *data-* tag attributes which you can use in all of your HTML code (including individual websites – only if you include the translr.js script file and call the updateTranslr() function at the page load)

Those tags are:

* *data-translr-value* (fills the HTML tag with the key value by *replacing the content*)
* *data-translr-title* (sets the HTML tag *title* attribute with the key value)
* *data-translr-placeholder* (sets the HTML tag *placeholder* attribute with the key value – for inputs)
* *data-translr-img* (sets the HTML tag *src* attribute with the key value – for images)
* *data-translr-check=”true”* (checks for *@translr-%key%* keys in the content and replaces them with a new span element which has the *data-translr-value* attribute)
* *data-translr-ptitle* (sets the page title with the key value – only for title tag)

1. Newsletter API

I thought that if you want to keep your visitors in touch with your news,

you can send newsletters. So I implemented a new form that inserts the e-mails in the database and after that you can get those addresses in multiple formats (JSON, XML, CSV, Plain text or email list) and use it to complete your mission.

1. Website requirements

First of all, this website requires an internet connection. ☺

After that, your visitors have to use a HTML5 and CSS3 compatible

browser (Google Chrome/webkit-based browser is recommended).

Secondly, your web server must use PHP and support MySQL Databases.

Recommended browsers versions:

* Google Chrome : almost any
* Firefox : >21
* Internet Explorer : >10 (11-12 recommended)
* Opera : almost any of the latest versions (which uses webkit)

This website is responsive, so it doesn’t matter if the user enters the

website from mobile or desktop.

1. **Used technologies summary**

* HTML5
* CSS3
* PHP
* MySQL
* AJAX
* jQuery
* jQuery UI
* Bootstrap
* Fontawesome
* Other custom scripts (written by me)

Legend

(T) – Supports Translr keys (@%key%)

(R) – Required field

(O) – Optional field

(CT) – Supports Translr content keys (@translr-%key%)

(HE) – HTML Editor field (supports HTML editing)

(HT) – Supports HTML tags (HTML content)

1. Using Translr, a specially-created plugin which also uses AJAX and Cookies. [↑](#footnote-ref-1)
2.  [↑](#footnote-ref-2)