# Exercise: Web Development Basics

This document defines the in class assignments from the ["Web Development Basics" Course @ Software University](https://softuni.bg/courses/web-development-basics/).

## Localization system

Translating with providing static content replicated in folders for each language was once the best approach to localize a website. Luckily it was twenty years ago and the times have slightly changed.

Your task this time is to provide localization system using cookies and an array of translated strings. Consider the user is clicking on a link with text “BG” or “EN”, your backend should parse this input, test whether it’s a valid language and then each provided tag is translated to the respective language.

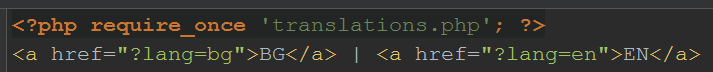
Some of the modern systems use tags to translate strings. For example if you have a big string for greeting the user and you need to translate it to any language, you provide to your translating system the tag ‘greeting\_user’ and it returns the translated text for the respective language.

### Task 1. Create links for changing language via GET

* Create a php file, let’s say, index.php, that will be your presentation to the end user
* Create two anchors that are sending to the backend via GET a language parameter for BG and EN

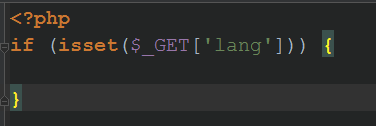


Clicking on each link does nothing except reloading the page. You need to provide a backend code that accepts the request. Let’s create a new file in the same folder called “translations.php” and include it in your presentation file



### Task 2. Parsing the request

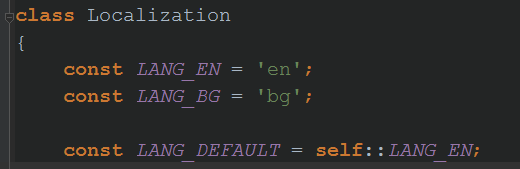
Now our translations.php will be executed procedurally before the presentation. Let’s take a look in it whether the ‘lang’ parameter is set.



If it is set we can trigger our translation logic. But for the sake of the good practices and avoiding code repetition we may do some things before that

### Task 3. Creating a class that holds the tags, translations and available langs

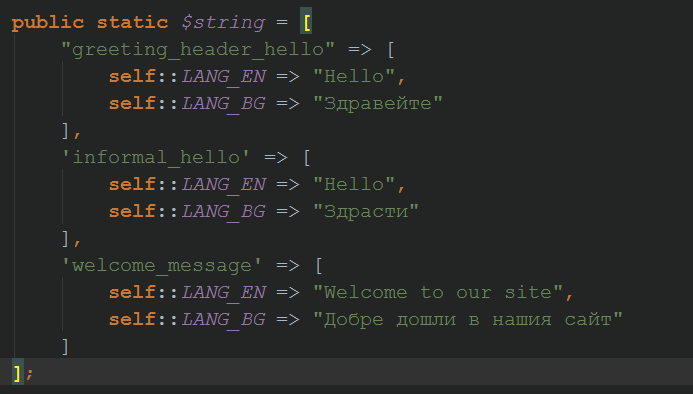
* Define constants for the available languages and a constant for the default language. Assign the last one to the English language



* Now we need a field that can be accessed from anywhere which holds the following structure

“tag\_name” => [ “english” => “tag\_translation\_in\_english”, “bulgarian” => “tag\_translation\_in\_bulgarian”]

* Let’s define some tags and their translations in this way, using the constants we have already defined

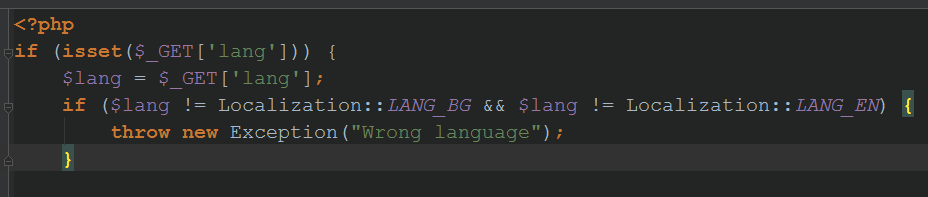


* Cool, eh? But still our application does nothing.

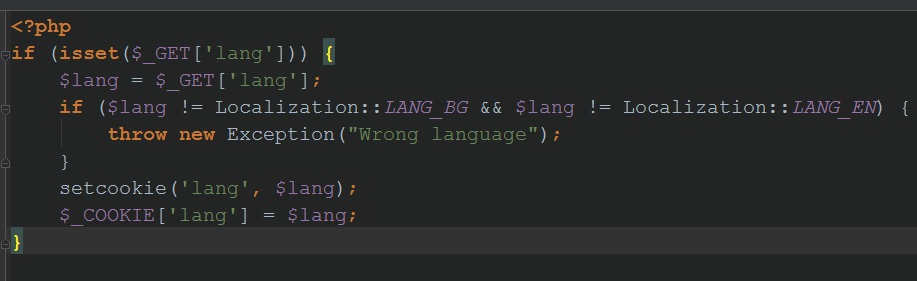
### Task 4. Use the newly created class to assign cookies

Now we have some tags, available languages and a default language.

* When the request with language is sent, we can use it and check whether it’s a valid language by comparing it with our constants in the Localization class

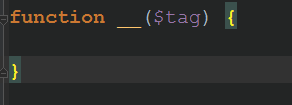


* If the language is valid after our check, we can assign it to a cookie using the setcookie() function and the $\_COOKIE superglobal

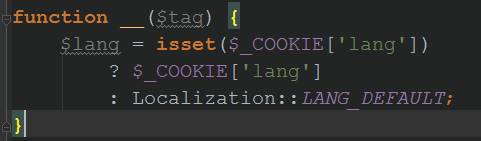


### Task 5. Creating a function that translates a given tag

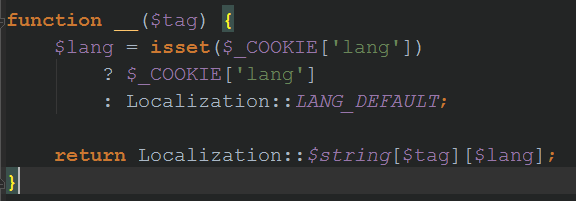
We have prepared everything in order to translate a given tag. Let’s create a function that will be used in the presentation file to translate tags. Because we don’t want to pollute the presentation with big function names or something we will declare our function name with two underscores \_\_(). In fact php uses one underscore \_() for native translation via gettext, but it’s out of scope now



Now we can use the value in the cookie in order to take something from the static strings array. Except that when the site is accessed for first the, no cookie is assigned and no language we have. We need to use the default language if there is no language in the cookies superglobal



Let’s now return the translated string by tag, by lang (e.g. strings[“informal\_hello”][“bg”])



We are now ready to try it in our presentation. Let’s translate some strings there

### Task 6. Translate some strings in the presentation

Just use the \_\_() function with provided tag in order to translate it.

