Dave Hollingworth: Internationalization (i18n) for PHP Developers

Character Sets

ISO/IEC 8859-1

Unicode UTF-8

Country Code +
Region Code = Locale

ISO 639-1:2002 ISO 3166

URL rewriting

Apache Mod Rewrite

.htaccess files

IntlDateFormatter

number formatter class

php gettext extension

MoTranslator

POEDIT

For .po & .mo files

`Locale::canonicalize`

HTTP_Accept_Language_Header

ipinfo.io Gelocalisation

Long Strings

Markdown

erusev/parsedown

$_SERVER

php setcookie

$_COOKIE

1. Intro to Translating Content using PHP:

Simple Example of Dynamic page content in different languages | Character sets and encoding | Language codes International Standard – ISO 639-1 | Language Code from URL segment | Language code from top-level Domain

2. Locale Identifiers & Validating the Language from the URL:

Add a i18n Class with a list of supported Languages | Combine language Codes & Region Codes to Create Locale Identifiers | Canonicalise the Locale code from the URL | Get the Best Match from the List of Locales | Redirect to the default locale if the Value in URL is invalid | Declare the language on the HTML document

3. Detecting the Visitors Preferred Language:

Get User's Preferred Language from Browser | Parse the Accept Language header to get Locale List | Compare Supported Languages to Browser preferences | Get Best Match to Browser Preferences using Just Language Code | Refactor the Code | Geolocation: matching Users IP address to location | Use Geolocation API to get Country Code from IP Address

4. Using gettext with PHP:

Options for Storing Translations in Separate Files | The PHP gettext Extension & Function for Marking Translatable strings | Create Folders to store gettext Translation Files | Install poeditor and Create a .po Translation File | Configure gettext to use the .mo Translation file

5. Using MpTranslator as a gettext compatible alternative:

MoTranslator as .mo file alternative to gettext | Configure PoEdit to extract MoTranslator translations | Use the Simpler MoTranslator object API | Disadvantages & Advantages of using Real or keyword messages

6. Translating Dynamic content, variable substitution, plurals, decimals & Dates:

Including Variables in Translated Strings: use Sprintf with gettext | Display Plural of Messages using ngettext | Decimal Separators: Format Decimal Numbers Based on Locale | Translate Day & Month names in Dates based on Locale

7. Translating Content Unsuited to gettext:long text, databases and images:

Handle Long Strings of Text in Different Files | Display Message if File Containing Translation is Unavailable | Use Plain Text Formatting Language to Help Translators | Use a Markdown Parser to Convert to HTML | Translate Multiple columns from a Database Table | Display Localised version Images That Contain Text

8. Selecting & Remembering Language:

Add Navigation Links for Switching Between Languages | Language Switching Links: What They Should Say & Where To Put Them | Remove Code Duplication: Extract Common i18n Code Out To Separate Files | Add a Second Page and include Common i18n Code | Generate the Data for Language navigation Links from $_SERVER data | Add Language navigation Links to HTML | Remember Selected Locale in a Cookie | Redirect to the Locale Stored in the Cookie

1.1 Simple Example of Dynamic page content in different languages

```php
<?php $lang = 'es’; ?>
<!DOCTYPE html>
<html>

<head>
  <title>
    <?php
    if ($lang == 'en') {
      echo 'Example';
    } elseif ($lang == 'es') {
      echo 'Ejemplo';
    }
    ?>
  </title>
</head>
<body>
  <h1>
    <?php
    if ($lang == 'en') {
      echo 'Home';
    } elseif ($lang == 'es') {
      echo 'Inicio';
    }
    ?>
  </h1>
  <p>
    <?php
    if ($lang == 'en') {
      echo 'Hello World';
    } elseif ($lang == 'es') {
      echo 'Hola y Bienvenido';
    }
    ?></p>
</body>
</html>
```

```php
<?php

$lang = 'es';

if ($lang == 'en') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>

<head>
  <meta charset=‘UTF-8>
  <title><?= $trans['title'] ?></title>
</head>

<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>

</body>

</html>
```

1.2 Character sets & Character encoding

There are multiple character sets that can be used to encode text.

ISO/IEC 8859-1 8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1, is part of the ISO/IEC 8859 series of ASCII-based standard character encodings

Unicode Unicode (also known as *The Unicode Standard* and TUS) is a character encoding standard maintained by the Unicode Consortium designed to support the use of text in all of the world's writing systems that can be digitized.

UTF-8 **UTF-8** is a character encoding standard used for electronic communication. Defined by the Unicode Standard, the name is derived from Unicode Transformation Format – 8-bit. As of July 2025, almost every webpage is transmitted as UTF-8

When, Encoding characters UTF 8 should be used and declared as the first html head tag (`<meta charset="UTF-8">`) because it contains pretty much every character we will ever come across. If it is not in UTF-8, the browser may not be able to interpret some characters and they display as a question mark inside a black diamond.

Inicio ISO/IEC 8859-1

Inicio UTF-8

¡Hola y bienvenido!

1.3 Language codes International Standard – ISO 639-1

ISO 639-1:2002, Codes for the representation of names of languages—Part 1: Alpha-2 code, is the first part of the ISO 639 series of international standards for language codes. Part 1 covers the registration of "set 1" two-letter codes. There are 183 two-letter codes registered as of June 2021. The registered codes cover the world's major languages.

The language code can be obtained from the query string of the URL

http://localhost/hollingworthPHPi18n/?lang=es

```php
<?php

$lang = $_GET['lang'];

if ($lang == 'en') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>

<body>

  <h1><?= $trans['header'] ?></h1>

  <p><?= $trans['welcome'] ?></p>

</body>
</html>
```

1.4 URL rewriting – Language Code from URL segment

URL in browser	URL on server
example.com/en	example.com/?lang=en
example.com/es/about.php	example.com/about.php?lang=es

URL rewriting can be done with **Apache Mod Rewrite** or **.htaccess files**

root/.htaccess

```
RewriteEngine on
RewriteRule ^([a-z]{2})/(.*)$ /$2?lang=$1 [L,NC,QSA]
RewriteRule ^([a-z]{2})$ index.php?lang=$1 [L,NC,QSA]
```

A .htaccess file is added to the root folder of the project that rewrites the ?lang=$1 part of the url to just $1/. The second line of the .htaccess file take any url that is two letters long and rewrites to index.php?lang=$1 where $1 is the two-letter language code.

http://localhost/hollingworthPHPi18n/en

Home

Hello and welcome!

http://localhost/hollingworthPHPi18n/es

Inicio

¡Hola y bienvenido!

1.5 Language Code from Sub Domain

The .htaccess file is removed.

Now the host or domain name is made up of two parts: a 2-letter language code followed by a dot followed by the host or server name.

http://en.localhost/hollingworthPHPi18n/

Home

Hello and welcome!

http://es.localhost/hollingworthPHPi18n/

Inicio

¡Hola y bienvenido!

```php
<?php

list($lang, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

if ($lang == 'en') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>
<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>
</html>
```

1.6 Language Code from Top Level Domain

Another way to get the language code would be to register a different **top-level domain** for each language. For example, the canon website uses canon.co.uk for the UK and canon.es for Spain. Note that not all languages have a top-level domain.

For this to work I have to configure the hosts in the **C:\Windows\System32\drivers\etc\hosts** and define the virtual host and the two aliases in **C:\xampp\apache\conf\extra\httpd-vhosts.conf**

http://hollingworthPHPi18n.co.uk **Home**

Hello and welcome!

http://hollingworthPHPi18n.es **Inicio**

¡Hola y bienvenido!

```php
<?php

if (substr($_SERVER['HTTP_HOST'], -6) == '.co.uk') {
  $lang = 'en';
} elseif (substr($_SERVER['HTTP_HOST'], -3) == '.es') {
  $lang = 'es';
}

if ($lang == 'en') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>
<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>
</html>
```

```php
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en', 'es']);

list($lang, $domain) = explode('.', $_SERVER['HTTP_HOST'],
2);

if ($lang == 'en') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>
<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>
</html>
```

2.1 Add a i18n Class with a list of supported Languages

This now gets the language code from the sub-domain.

root/src/App/I18n.php

```php
<?php

namespace App;

class I18n
{
  private $supported_locales;

  public function __construct(array $supported_locales)
  {
    $this->supported_locales = $supported_locales;
  }
}
```

2.2 Combine language Codes & Region Codes to Create Locale Identifiers

Some languages differ by region and country i.e. UK-English vs US-English. The regions are identified by an additional two letters i.e. GB or US where the country codes are defined in **ISO 3166**

Locale	Language (Region)
en_GB	English (Great Britain)
en_US	English (United States)
es_ES	Spanish (Spain)
es_MX	Spanish (Mexico)

The language (ISO 639-1) can be combined with the country (ISO 3166) to form a locale. Note the format where the language is in lowercase followed by an underscore then the country in uppercase. Note that different operating systems and browsers prefer an underscore or a hyphen separating the language and country, but fortunately PHP provides inbuilt functions to handle this.

2.3 Canonicalise the Locale code from the URL

For example, if I change the code to make English GB the accepted language, when I use this as a subdomain the browser is converting GB to lowercase so it fails.
http://en_gb.localhost/hollingworthPHPi18n/

Warning: Undefined variable $trans in C:\xampp\htdocs\hollingworthPHPi18n\index.php on line 40

Warning: Trying to access array offset on value of type null in C:\xampp\htdocs\hollingworthPHPi18n\index.php on line 40

I solve this by using the php function canonicalize which also works with a hyphen.

http://en-gb.localhost/hollingworthPHPi18n/

```php
<?php
require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.',
$_SERVER['HTTP_HOST'], 2);

$lang = Locale::canonicalize($subdomain);

if ($lang == 'en_GB') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}
?>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>
<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>
</html>
```

2.4 Get the Best Match from the List of Locales

It is highly unusual that the language_locale would be used in the URL, Normally it is just the language code so I need to match a language code to a supported locale.

Code from the URL	Supported Locale
en	en_GB
es	es_MX

In the I18n class, I add a function to get the best matched locale to the language code.

```
...
  public function getBestMatch(string $lang)
  {
    $lang = \Locale::canonicalize($lang);

    if (in_array($lang, $this->supported_locales)) {

      return $lang;
    } else {

      foreach ($this->supported_locales as $supported_locale) {

        if (substr($supported_locale, 0, 2) == $lang) {

          return $supported_locale;
        }
      }
    }

    return null;
  }
}
```

root/src/App/I18n.php

In the index.php file, I call the best match function to get a language_LOCALE return from the two-letter language in the URL.

```php
<?php
require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$lang = $i18n->getBestMatch($subdomain);

if ($lang == 'en_GB') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($lang == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}

?>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>
<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>
</html>
```

2.5 Redirect to the default locale if the Value in URL is invalid

In the I18n class, the getDefault method returns the first language in the array of supported locales.

root/src/App/I18n.php

```
public function getDefault()
{
  return $this->supported_locales[0];
}
```

If the getBestmatch returns NULL, then the default is returned and the page is redirected to this URL

http://en_us.localhost/hollingworthPHPi18n/ redirects to http://en.localhost/hollingworthPHPi18n/

http://es_mx.localhost/hollingworthPHPi18n/ redirects to http://en.localhost/hollingworthPHPi18n/ but just using es in the URL stays on the Spanish language.

root/index.php

```
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$lang = $i18n->getBestMatch($subdomain);

if ($lang === null) {

  $default = substr($i18n->getDefault(), 0, 2);

  header("Location: http://" . $default .
".localhost/hollingworthPHPi18n/");
  exit;
}
```

2.6 Declare the language on the HTML document

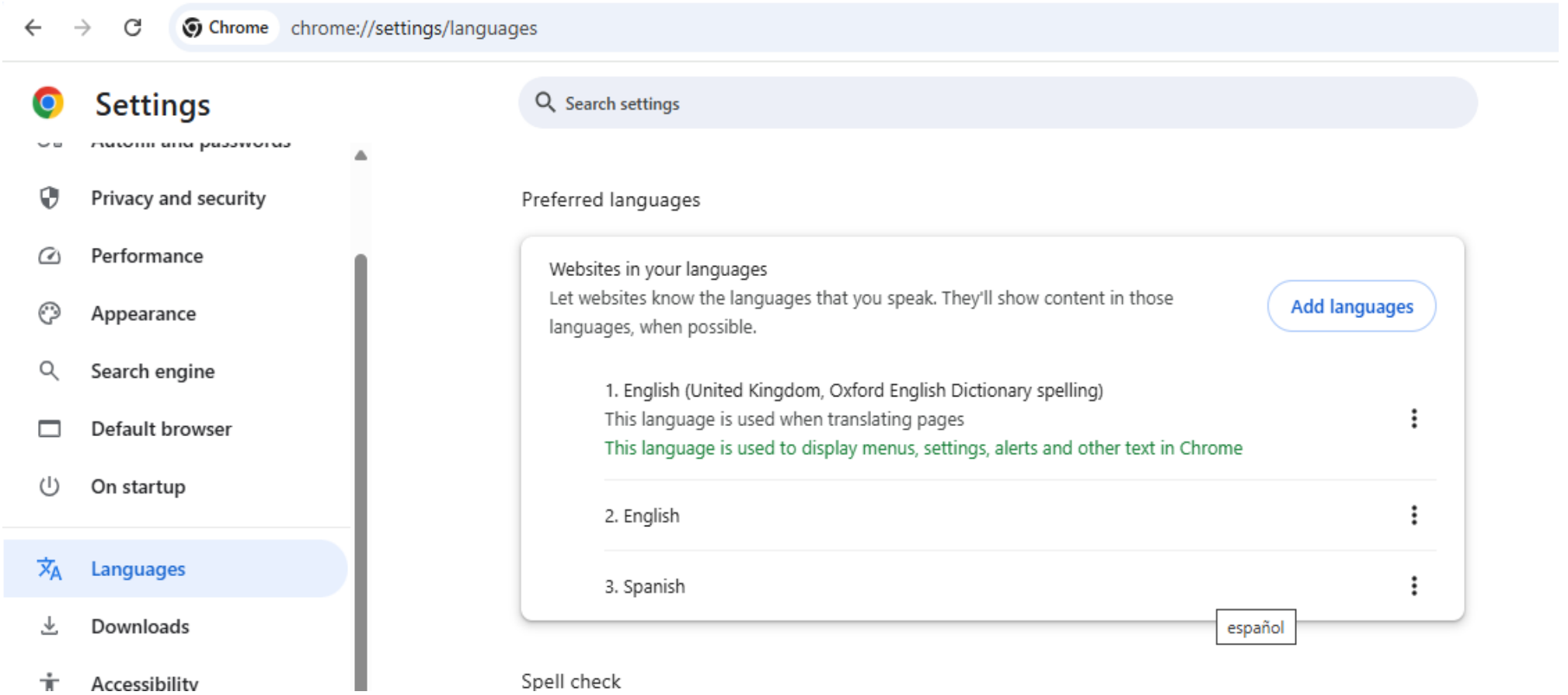
```
...
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $lang) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $trans['title'] ?></title>
</head>

<body>
  <h1><?= $trans['header'] ?></h1>
  <p><?= $trans['welcome'] ?></p>
</body>

</html>
```

root/index.php

The **HTML tag takes a language value** which can be language or language-locale where locale can be in upper or lowercase. Note that PHP uses an underscore to separate language and locale where as the HTML uses a hyphen so I am using string replace to swap them out.

3.1 Get User's Preferred Language from Browser

Browsers have settings for the user's preferred language where an ordered list of first choice language, second choice language etc can be configured. This Locale information is passed via the **HTTP request accept language header** to the website.



The accept language HTTP header can be accessed in the php $_SERVER super global.

root/index.php

```php
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.',
$_SERVER['HTTP_HOST'], 2);

$lang = $i18n->getBestMatch($subdomain);

var_dump($_SERVER['HTTP_ACCEPT_LANGUAGE']);
```

q is a weighing factor with 1 being the highest value. The higher the value the more preferred the language is.

string(32) "en-GB-oxendict,en;q=0.9,es;q=0.8"

3.2 Parse the Accept Language header to get Locale List

If the HTTP_ACCEPT_LANGUAGE header is present, then it splits each language locale into a part on the coma and loops over it in a foreach loop.

This function then splits in the ;q= part and extracts the locale sorting the array by highest q value into a list of preferred locales.

root/src/App/I18n.php

```php
public function getAcceptedLocales()
  {
    if ($_SERVER['HTTP_ACCEPT_LANGUAGE'] == '') {
      return [];
    }

    $accepted_locales = [];

    $parts = explode(',', $_SERVER['HTTP_ACCEPT_LANGUAGE']);

    foreach ($parts as $part) {

      $locale_and_pref = explode(';q=', $part);

      $locale = trim($locale_and_pref[0]);
      $pref = $locale_and_pref[1] ?? 1.0;

      $accepted_locales[$locale] = $pref;
    }

    arsort($accepted_locales);

    return array_keys($accepted_locales);
  }
```

root/index.php

```php
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.',
$_SERVER['HTTP_HOST'], 2);

$lang = $i18n->getBestMatch($subdomain);

var_dump($_SERVER['HTTP_ACCEPT_LANGUAGE']);

print_r($i18n->getAcceptedLocales());
```

```
string(32) "en-GB-oxendict,en;q=0.9,es;q=0.8“
Array (
  [0] => en-GB-oxendict
  [1] => en
  [2] => es
)
```

3.3 Compare Supported Languages to Browser preferences

root/src/App/I18n.php

```php
public function getBestMatchFromHeader()
  {
    $accepted_locales = $this->getAcceptedLocales();

    foreach ($accepted_locales as $locale) {

      $locale = \Locale::canonicalize($locale);

      if (in_array($locale, $this->supported_locales)) {

        return $locale;
      }
    }
```

root/index.php

```php
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = array_pad(explode('.', $_SERVER['HTTP_HOST'], 2),
2, null);

$lang = $i18n->getBestMatch($subdomain);

if ($lang === null) {

  $lang = $i18n->getBestMatchFromHeader();
  var_dump($lang);
  exit;

  $default = substr($i18n->getDefault(), 0, 2);

  header("Location: http://" . $default . ".phpi18n.localhost/");
  exit;
}
```

string(5) "en_GB"

3.4 Get Best Match to Browser Preferences using Just Language Code

If the HTTP_ACCEPT_LANGUAGE header is passing a language and location i.e. es-ES then it would not match on the accepted language of es.

So, I have modified the gtBestmatchFromHeader function to walk over the language and region so that if it does not match language & region then it will match on only the language.

root/src/App/I18n.php

```php
public function getBestMatchFromHeader()
  {
    $accepted_locales = $this->getAcceptedLocales();

    array_walk($accepted_locales, function (&$locale) {
      $locale = \Locale::canonicalize($locale);
    });

    foreach ($accepted_locales as $locale) {
      if (in_array($locale, $this->supported_locales)) {
        return $locale;
      }
    }

    foreach ($accepted_locales as $locale) {

      $lang = substr($locale, 0, 2);

      foreach ($this->supported_locales as $supported_locale) {
        if (substr($supported_locale, 0, 2) == $lang) {
          return $supported_locale;
        }
      }
    }

    return null;
  }
```

3.5 Refactor the Code

root/src/App/I18n.php

```
...
public function getLocaleForRedirect() {

  $locale = $this->getBestMatchFromHeader();

  if ($locale !== null) {
    return $locale;
  }

  return $this->getDefault();
}
```

Now it uses the HTTP Accept Language header to determine the best match language then set the sub domain accordingly or to a default of en.

root/index.php

```
<?php

require 'src/App/I18n.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain .
".localhost/hollingworthPHPi18n");
  exit;
}

if ($locale == 'en_GB') {

  $trans = [
    'title' => 'Example',
    'header' => 'Home',
    'welcome' => 'Hello and welcome!'
  ];
} elseif ($locale == 'es') {

  $trans = [
    'title' => 'Ejemplo',
    'header' => 'Inicio',
    'welcome' => '¡Hola y bienvenido!'
  ];
}
...
```

3.6 Geolocation: matching Users IP address to location

One way to do this is to download and maintain fresh a database of IP addresses and locations from a provider such as **MaxMind** (fees apply) then look up the users ip address locally on my server.

Another method would be to use an API call to a service such as **ipinfo.io** which requires **free signup** to obtain an access token. They have an **official php library** to access the service which can be downloaded from the command line using **composer require ipinfo/ipinfo**

```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.',
$_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain .
".localhost/hollingworthPHPi18n");
  exit;
}
...
```

I can include the ipinfo components using the composer autoloader.

3.7 Use Geolocation API to get Country Code from IP Address

```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

// list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);
list($subdomain, $domain) = array_pad(explode('.', $_SERVER['HTTP_HOST'], 2), 2, null);

// $locale = $i18n->getBestMatch($subdomain);

$locale = $i18n->getBestMatchFromIPAddress();
```

root/index.php

```php
  public function getBestMatchFromIPAddress()
  {
    try {
      $client = new \ipinfo\ipinfo\IPinfo('48c7a8a61d902f');
      $details = $client->getDetails('80.24.0.0');
    } catch (\ipinfo\ipinfo\IPinfoException $e) {
      echo $e->getMessage();
    }
    var_dump($details);
  }
}
```

root/src/App/I18n.php

object(ipinfo\ipinfo\Details)#34 (26) { **["country"]=> string(2) "ES"** ["country_name"]=> string(5) "Spain" ["country_flag"]=> array(2) { ["emoji"]=> string(8) "ES" ["unicode"]=> string(15) "U+1F1EA U+1F1F8" } ["country_code"]=> NULL ["country_flag_url"]=> string(58) "https://cdn.ipinfo.io/static/images/countries-flags/ES.svg" ["country_currency"]=> array(2) { ["code"]=> string(3) "EUR" ["symbol"]=> string(3) "€" } ["continent"]=> array(2) { ["code"]=> string(2) "EU" ["name"]=> string(6) "Europe" } ["latitude"]=> string(7) "37.3828" ["longitude"]=> string(7) "-5.9732" ["loc"]=> string(15) "37.3828,-5.9732" ["is_eu"]=> bool(true) ["ip"]=> string(9) "80.24.0.0" ["hostname"]=> string(35) "0.red-80-24-0.staticip.rima-tde.net" ["anycast"]=> NULL ["city"]=> string(7) "Sevilla" ["org"]=> string(34) "AS3352 TELEFONICA DE ESPANA S.A.U." ["postal"]=> string(5) "41005" ["region"]=> string(9) "Andalusia" ["timezone"]=> string(13) "Europe/Madrid" ["asn"]=> NULL ["company"]=> NULL ["privacy"]=> NULL ["abuse"]=> NULL ["domains"]=> NULL ["bogon"]=> NULL ["all"]=> array(17) { ["ip"]=> string(9) "80.24.0.0" ["hostname"]=> string(35) "0.red-80-24-0.staticip.rima-tde.net" ["city"]=> string(7) "Sevilla" ["region"]=> string(9) "Andalusia" ["country"]=> string(2) "ES" ["loc"]=> string(15) "37.3828,-5.9732" ["org"]=> string(34) "AS3352 TELEFONICA DE ESPANA S.A.U." ["postal"]=> string(5) "41005" ["timezone"]=> string(13) "Europe/Madrid" ["country_name"]=> string(5) "Spain" ["is_eu"]=> bool(true) ["country_flag"]=> array(2) { ["emoji"]=> string(8) "ES" ["unicode"]=> string(15) "U+1F1EA U+1F1F8" } ["country_flag_url"]=> string(58) "https://cdn.ipinfo.io/static/images/countries-flags/ES.svg" ["country_currency"]=> array(2) { ["code"]=> string(3) "EUR" ["symbol"]=> string(3) "€" } ["continent"]=> array(2) { ["code"]=> string(2) "EU" ["name"]=> string(6) "Europe" } ["latitude"]=> string(7) "37.3828" ["longitude"]=> string(7) "-5.9732" } }

If no HTTP language header comes then it can use the IP geo localisation as a fallback to detect the language

```
...
private function getBestMatchFromIPAddress()
  {
    try {
      $client = new \ipinfo\ipinfo\IPinfo('48c7a8a61d902f');
      $details = $client->getDetails($_SERVER['REMOTE_ADDR']);

      var_dump($details);

      if (isset($details->country)) {
        return $this->getBestMatch($details->country);
      }
    } catch (\ipinfo\ipinfo\IPinfoException $e) {
      echo $e->getMessage();
    }
    var_dump($details);
  }

  public function getLocaleForRedirect()
  {
    // 1. Detect language from HTTP header
    $locale = $this->getBestMatchFromHeader();
    if ($locale !== null) {
      return $locale;
    }

    // 2. detect language from Geolocation IP address
    $locale = $this->getBestMatchFromIPAddress();
    if ($locale !== null) {
      return $locale;
    }

    // 3. if unable to determine languege use default
    return $this->getDefault();
  }
}
```

root/src/App/I18n.php

root/index.php

```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}
```

4.1 Options for Storing Translations in Separate Files

At the moment I am storing the translations in an associative array. But there are othr options to store them as separate files. From the **symphony translation documentation**, the translations could be stored as:

1. XML file using a format called **XLIFF** which is quite verbose.

1. YAML files are more readable.

1. **PO Files**

PO files are what php gettext uses to perform translations and although it has been around for years, is still the industry standard and is what many php applications such as WordPress use today.

4.2 The PHP gettext Extension & Function for Marking Translatable strings

To use PHP gettext, the **php gettext extension** has to be installed and enabled.

I mark all the strings That I want to translate by wrapping the raw text in the gettext function.

I can remove the if block for language locales.

Note that if at this point there is an undefined function error, then it means that the gettest extension is not installed.

root/index.php

```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain .
".localhost/hollingworthPHPi18n");
  exit;
}

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $lang) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= gettext('Example') ?></title>
</head>

<body>
  <h1><?= gettext('Home') ?></h1>
  <p><?= gettext('Hello and welcome!') ?></p>
</body>

</html>
```

The extension can be found in the php.ini file and should not be commented out (extension=gettext)

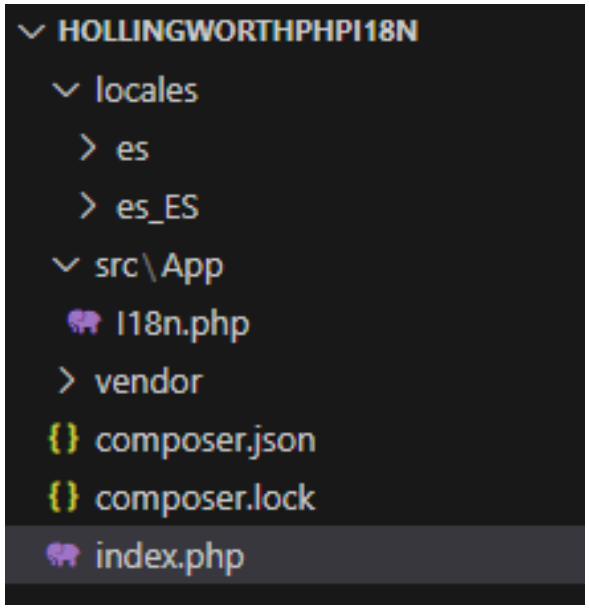
The gettext function also has an alias of an underscore so that when writing code, it become less intrusive.

Now that I am telling gettext to translate the text, I need a folder and file system to store the translations.

```php
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $lang) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= _('Example') ?></title>
</head>

<body>
  <h1><?= _('Home') ?></h1>
  <p><?= _('Hello and welcome!') ?></p>
</body>

</html>
```

root/index.php

4.3 Create Folders to store gettext Translation Files

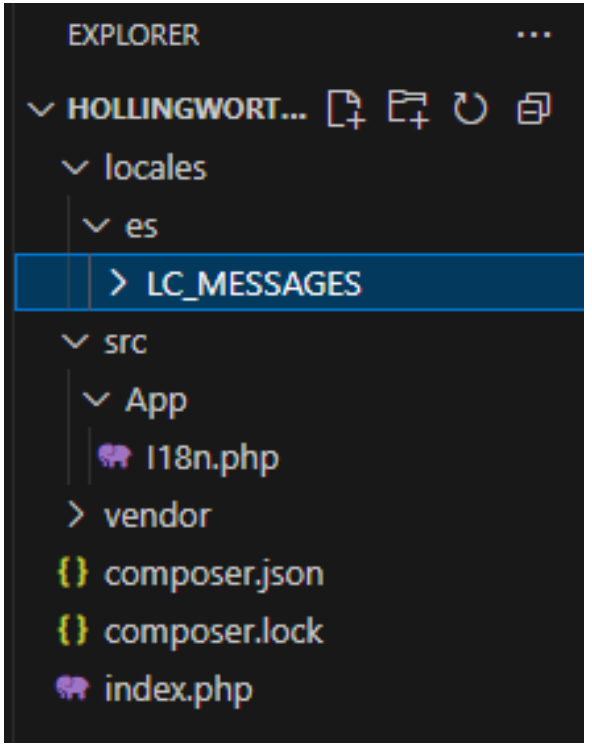


The gettext translation folder should be stored in the root of the project. It can be named anything so I have called mine, locales.

Within this folder sits a folder for each language. The naming convention is specific and important. If I want a folder for a specific language but with no region then it is named in lower case as the 2-letter language code, i.e. Spanish is es.

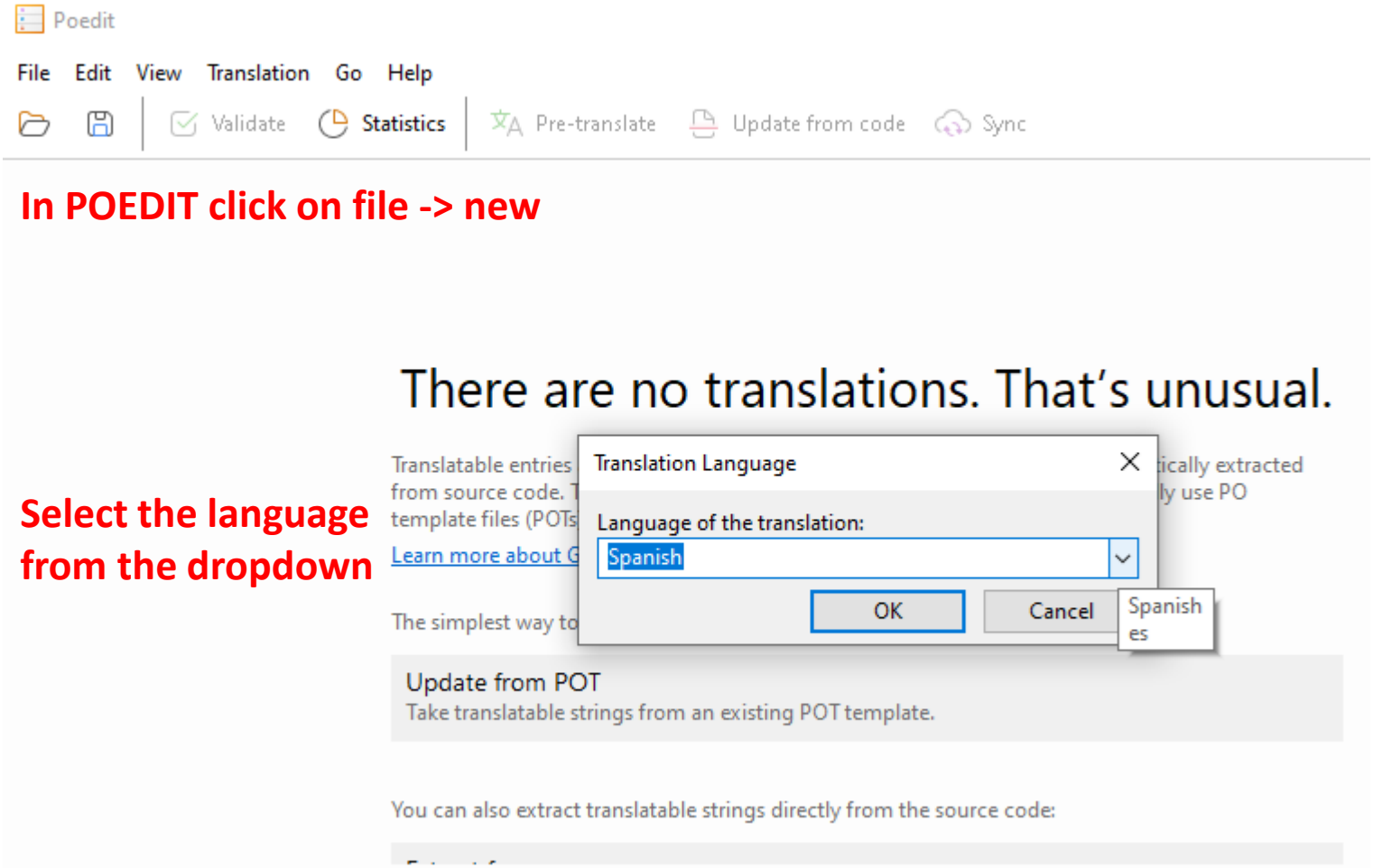
If I want a folder for a specific language and region, then I name it country code (2 lowercase letters) underscore region code in uppercase. i.e. es_ES would contain translation files specific for Spanish for Spain.

Within, the locale folder, a sub-folder called LC_LANGUAGES (uppercase) is needed.



4.4 Install poeditor and Create a .po Translation File

POEDIT: Powerful and intuitive translation editor for PO, XLIFF, JSON or Flutter formats

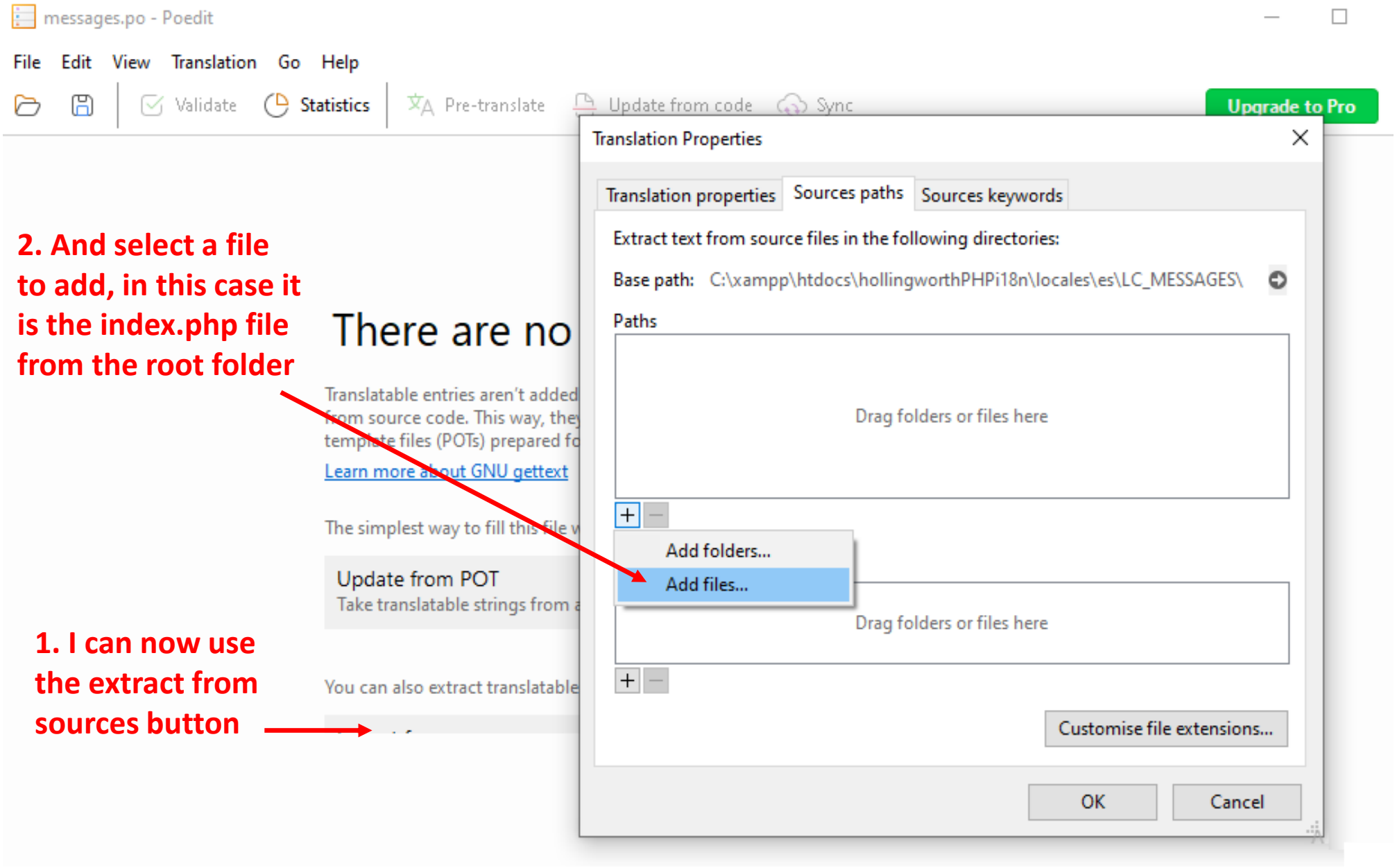


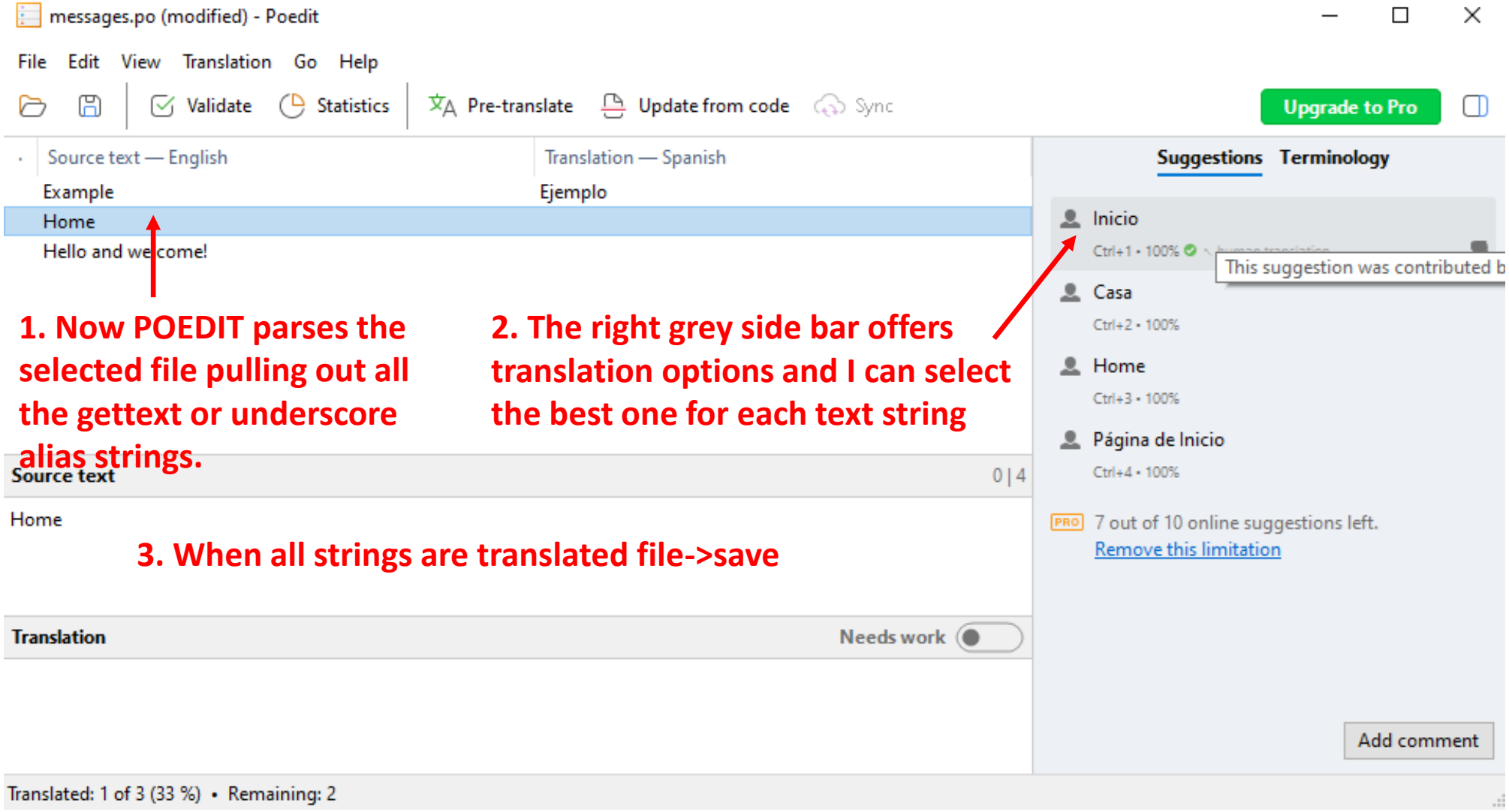
In POEDIT click on file -> new

Select the language from the dropdown

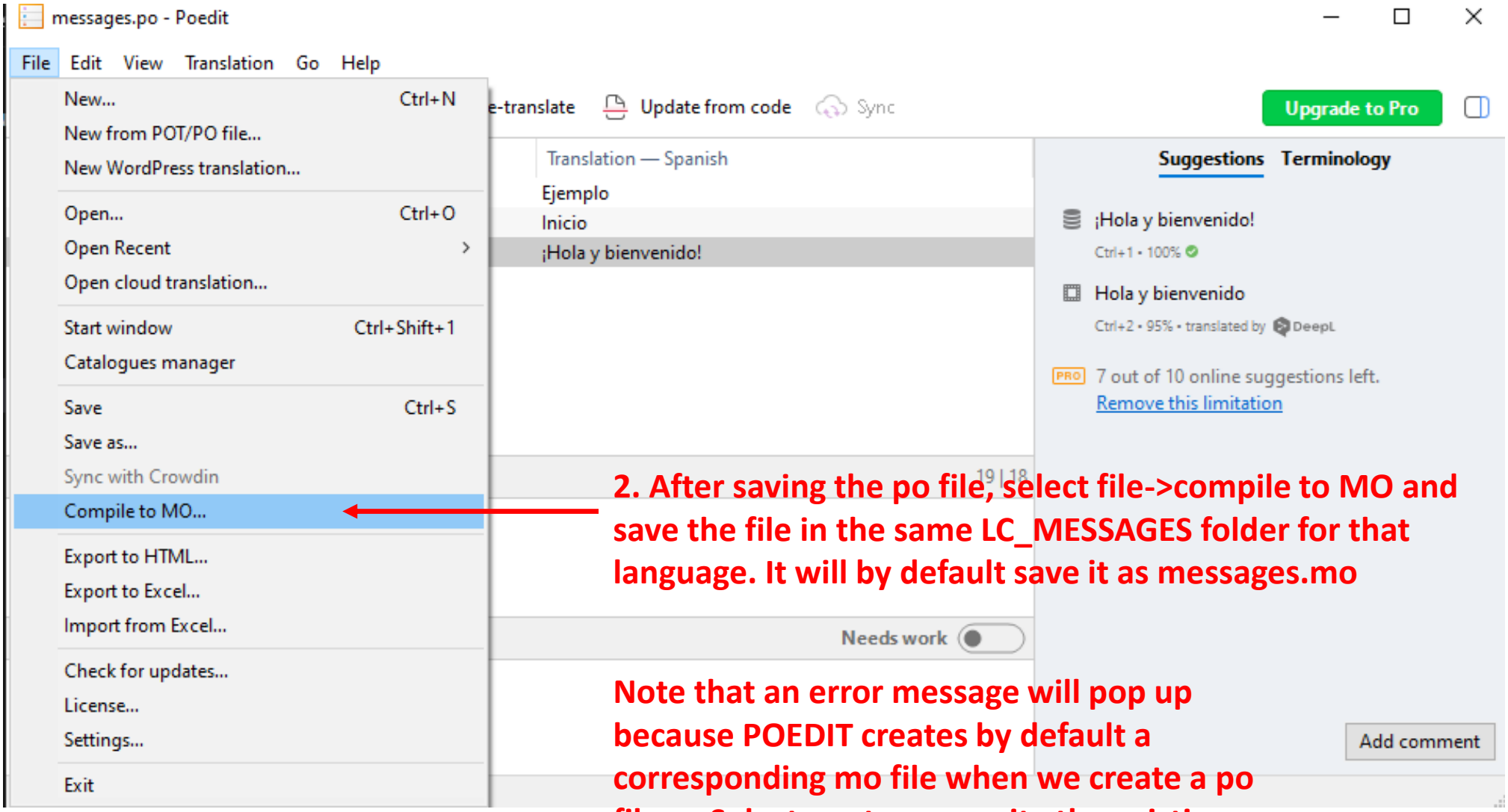
The language can be language only (es) or language plus region (es_ES)

Now save the file in the LC_MESSAGES folder for that locale/language. The file can be named anything but the standard is to call it messages.po in lowercase.


messages.po - Poedit
File Edit View Translation Go Help
Validate
Statistics
Pre-translate
Update from code
Sync
Upgrade to Pro
2. And select a file to add, in this case it is the index.php file from the root folder
There are no
Translatable entries aren't added
from source code. This way, they
template files (POTs) prepared fo
Learn more about GNU gettext
The simplest way to fill this file w
Update from POT
Take translatable strings from a
You can also extract translatable
1. I can now use the extract from sources button
Translation Properties
Translation properties
Sources paths
Sources keywords
Extract text from source files in the following directories:
Base path: C:\xampp\htdocs\hollingworthPHPi18n\locales\es\LC_MESSAGES\
Paths
Drag folders or files here
Add folders...
Add files...
Drag folders or files here
Customise file extensions...
OK
Cancel


messages.po (modified) - Poedit
File Edit View Translation Go Help
Validate
Statistics
Pre-translate
Update from code
Sync
Upgrade to Pro
Source text — English
Translation — Spanish
Example
Ejemplo
Home
Hello and welcome!
1. Now POEDIT parses the selected file pulling out all the gettext or underscore alias strings.
2. The right grey side bar offers translation options and I can select the best one for each text string
Source text
0 | 4
Home
3. When all strings are translated file->save
Translation
Needs work
Suggestions
Terminology
Inicio
Ctrl+1 • 100%
This suggestion was contributed b
Casa
Ctrl+2 • 100%
Home
Ctrl+3 • 100%
Página de Inicio
Ctrl+4 • 100%
PRO
7 out of 10 online suggestions left.
Remove this limitation
Add comment
Translated: 1 of 3 (33 %) • Remaining: 2

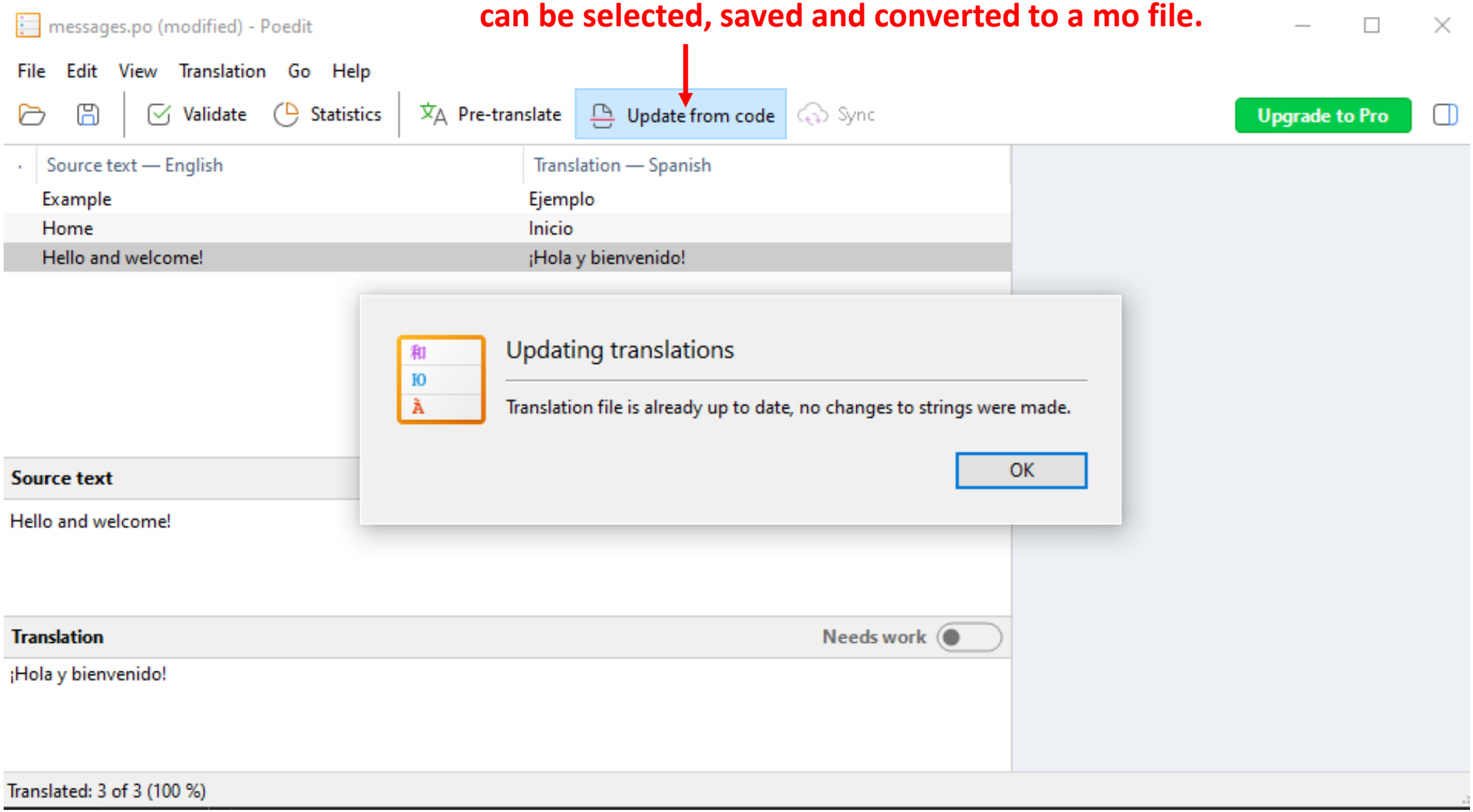
1. Gettext does not read mo files directly but uses machine object or .mo files which are a type of binary file.



2. After saving the po file, select file->compile to MO and save the file in the same LC_MESSAGES folder for that language. It will by default save it as messages.mo

Note that an error message will pop up because POEDIT creates by default a corresponding mo file when we create a po file so Select yes to overwrite the existing mo file.

1. If the source file is changed, the update from code button can be used and it will rescan the source file o that new translations can be selected, saved and converted to a mo file.



```
msgid ""
msgstr ""
"Project-Id-Version: \n"
"POT-Creation-Date: 2025-09-02 18:16+0100\n"
"PO-Revision-Date: 2025-09-02 18:20+0100\n"
"Last-Translator: \n"
"Language-Team: \n"
"Language: es\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"X-Generator: Poedit 3.7\n"
"X-Poedit-Basepath: ../../..\n"
"X-Poedit-SearchPath-0: index.php\n"

#: index.php:29
msgid "Example"
msgstr "Ejemplo"

#: index.php:33
msgid "Home"
msgstr "Inicio"

#: index.php:34
msgid "Hello and welcome!"
msgstr "¡Hola y bienvenido!"
```

A po file contains a header and below the original text with what line of code they came from and below each original text is the translation.

Using POEDIT to generate po files requires some understanding of the language being translated to which is why it is used mostly by professional translators.

4.5 Configure gettext to use the .mo Translation file

setLocale sets the locale that php will use.

The file of the translation.mo file is refered to the domain, i.e. messages so this can be set to a variable.

textDomain sets the text somain value to the variable domain.

BindTextDomain tells php where the mo files are. I specify locales which is why the subfolders need to adhere to the specific structure.

Bind_textdomain_codeset specifiies what character encoding is used.

putenv is to force the environment variables to the language to get gettext working.

It still does not work because gettext is notoriously tricky to get working in php

root/index.php

```php
<?php
...

putenv("LANG=$locale");
putenv("LANGUAGE=$locale");

setlocale(LC_ALL, $locale);

$domain = 'messages';
textdomain($domain);
bindtextdomain($domain, 'locales');
bind_textdomain_codeset($domain, 'UTF-8');

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= _('Example') ?></title>
</head>

<body>
  <h1><?= _('Home') ?></h1>
  <p><?= _('Hello and welcome!') ?></p>
</body>

</html>
```

https://stackoverflow.com/questions/62353692/gettext-on-windows-10-xampp-doesnt-translate-my-text

5.1 MoTranslator as .mo file alternative to gettext

MoTranslator is a package created by the same team as phpMyAdmin works with .mo files so is easy to use and can be installed from the command line with **composer require phpmyadmin/motranslator**

I need to use the autoloader to bring in MoTranslator functions. I then call loadFunctions() on MoTranslator.

All the other functions emulate gettext and are anmed with an underscore infront.

In the HTML where I was using an underscore to call gettext I now use a double underscore the call MoTranslator's gettext emulator function.

root/index.php

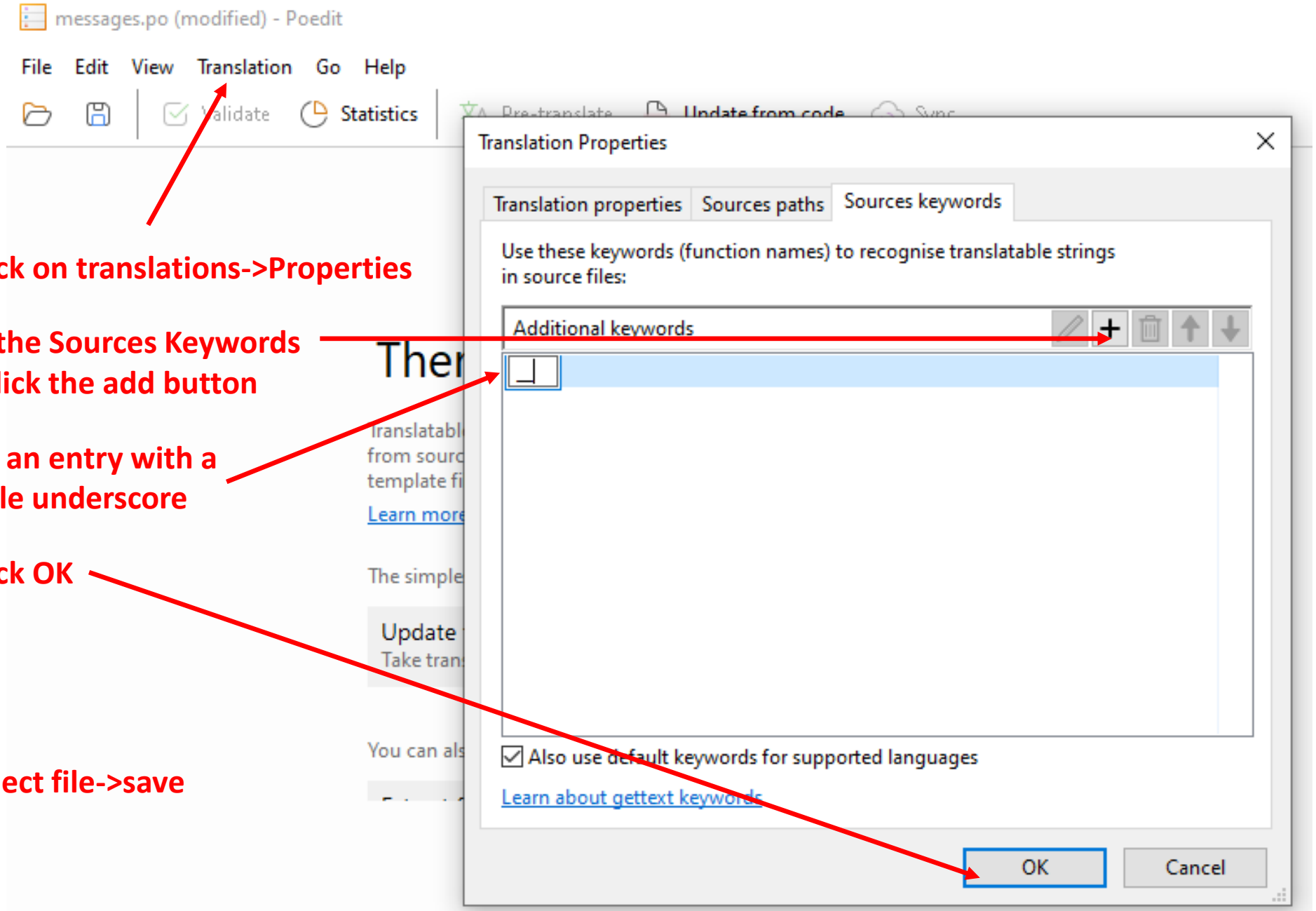
```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

...

PhpMyAdmin\MoTranslator\Loader::loadFunctions();

_setlocale(LC_ALL, $locale);
$domain = 'messages';
_textdomain($domain);
_bindtextdomain($domain, 'locales');
_bind_textdomain_codeset($domain, 'UTF-8');

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= __('Example') ?></title>
</head>

<body>
  <h1><?= __('Home') ?></h1>
  <p><?= __('Hello and welcome!') ?></p>
</body>

</html>
```

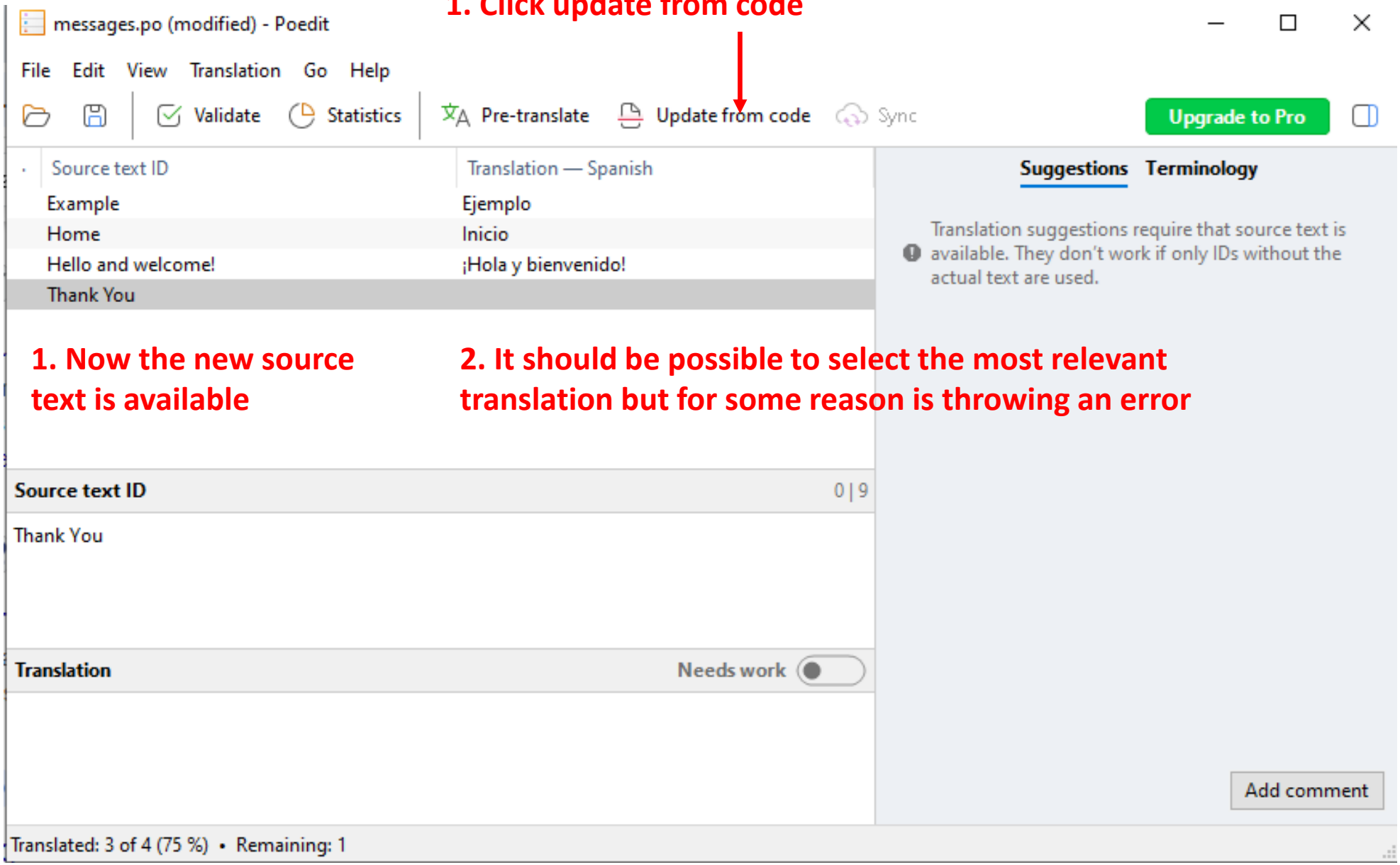
5.2 Configure PoEdit to extract MoTranslator translations

I have added another line of hml to the php.index File but POEDIT will not find them because they now have two underscores and not one.

root/index.php

```
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= __('Example') ?></title>
</head>

<body>
  <h1><?= __('Home') ?></h1>
  <p><?= __('Hello and welcome!') ?></p>
  <p><?= __(‘Thank You') ?></p>
</body>

</html>
```


messages.po (modified) - Poedit
File Edit View Translation Go Help
Validate
Statistics
Translation Properties
Translation properties
Sources paths
Sources keywords
Use these keywords (function names) to recognise translatable strings in source files:
Additional keywords
Also use default keywords for supported languages
Learn about gettext keywords
OK
Cancel
1. Click on translations->Properties
2. In the Sources Keywords tab click the add button
3. Ad an entry with a double underscore
4. Click OK
5. Select file->save


1. Click update from code
messages.po (modified) - Poedit
File Edit View Translation Go Help
Validate
Statistics
Pre-translate
Update from code
Sync
Upgrade to Pro
Source text ID
Translation — Spanish
Example
Ejemplo
Home
Inicio
Hello and welcome!
¡Hola y bienvenido!
Thank You
Suggestions
Terminology
Translation suggestions require that source text is available. They don't work if only IDs without the actual text are used.
1. Now the new source text is available
2. It should be possible to select the most relevant translation but for some reason is throwing an error
Source text ID
0 | 9
Thank You
Translation
Needs work
Add comment
Translated: 3 of 4 (75 %) • Remaining: 1

5.3 Use the Simpler MoTranslator object API

```php
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= $translator->gettext('Hello and welcome!') ?></p>
  <p><?= $translator->gettext('Thank You') ?></p>
</body>

</html>
```

I create a new MoTranslator object passing in the path to the .mo file

I nolonger have access to the two-underscore method so I have to call the translator->gettext method instead.

The advantage to this method is that I can remove all the gettext configuration calls. The disadvantage is that I loose the double underscore function which adds more code to the HTML.

I can also now put the .mo files anywhere I want. I could just name the po and mo files as the language i.e. es.po and es.po. Then change the path tp these files to be locales/$locale.mo

5.4 Disadvantages & Advantages of using Real or keyword messages

I can replace the actual text in the html with a unique keyword or key phrase for each string. Then I create a .mo file for all languages including the original en-GB language.

The advantage of this is that it can make the html very short and readable, but it does mean an additional mo file for the original language. And if I change A keyword then I need to update all the mo files with the new translations.

6.1 Including Variables in Translated Strings: use Sprintf with gettext

Sprintf uses specifiers. For example %s would be a string.

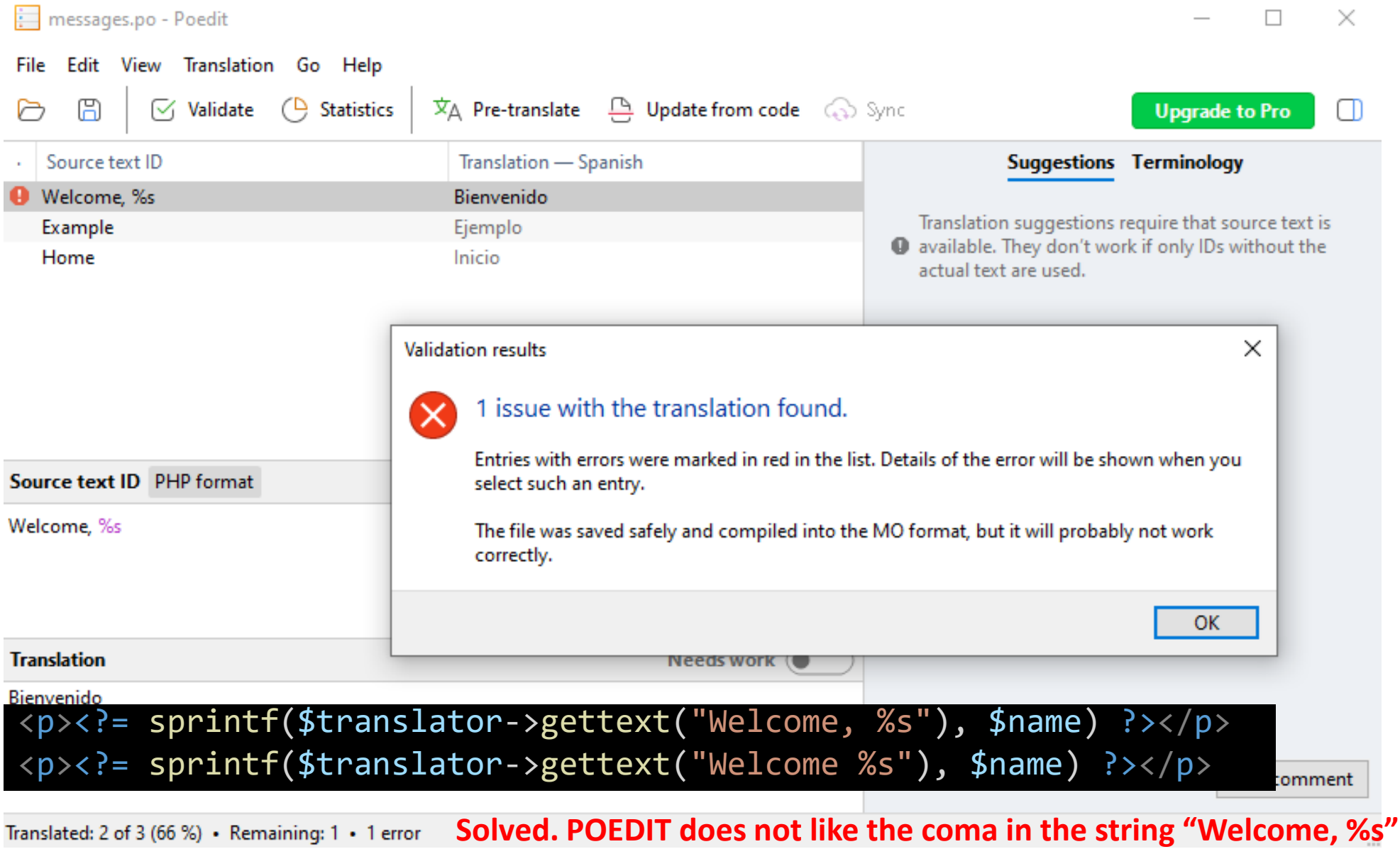
Specifiers

Specifier	Description
%	A literal percent character. No argument is required.
b	The argument is treated as an integer and presented as a binary number.
c	The argument is treated as an integer and presented as the character with that ASCII.
d	The argument is treated as an integer and presented as a (signed) decimal number.
e	The argument is treated as scientific notation (e.g. 1.2e+2).
E	Like the e specifier but uses uppercase letter (e.g. 1.2E+2).
f	The argument is treated as a float and presented as a floating-point number (locale aware).
F	The argument is treated as a float and presented as a floating-point number (non-locale aware).
g	General format.

```php
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= sprintf($translator->gettext("Welcome, %s"), $name) ?></p>
</body>

</html>
```

The part of the text including the specifier is wrapped in the translator->gettext function and the outer part including the variable is wraped in the sprint.

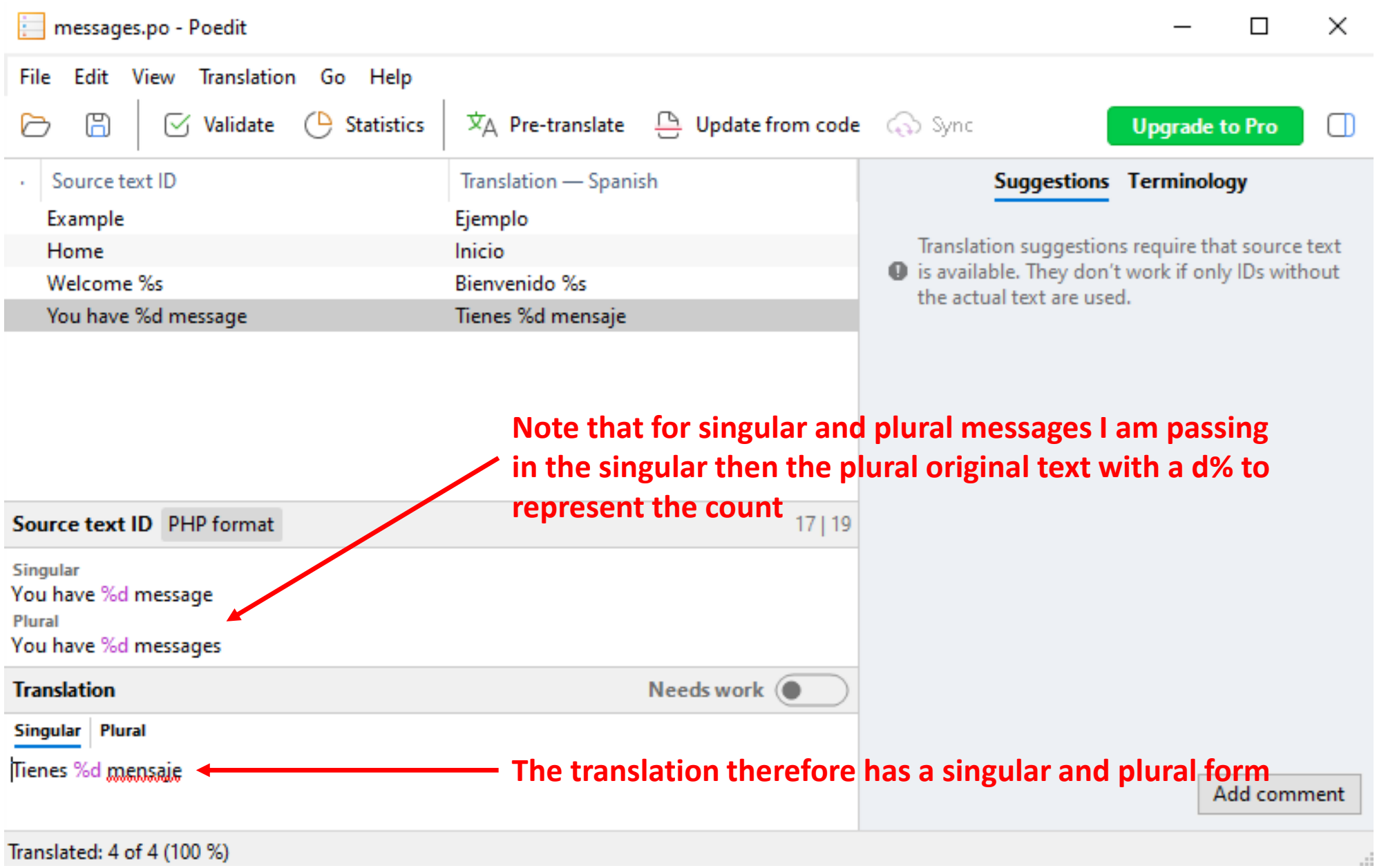
POEDIT should be able to understand this but for some reason is throwing an error.



Solved. POEDIT does not like the coma in the string "Welcome, %s"

6.2 Display Plural of Messages using ngettext

```php
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= sprintf($translator->gettext("Welcome %s"), $name) ?></p>
  <p><?= sprintf($translator->ngettext("You have %d message", "You have %d messages", $count), $count) ?></p>
</body>

</html>
```


messages.po - Poedit
File Edit View Translation Go Help
Validate
Statistics
Pre-translate
Update from code
Sync
Upgrade to Pro
Source text ID
Translation — Spanish
Example
Ejemplo
Home
Inicio
Welcome %s
Bienvenido %s
You have %d message
Tienes %d mensaje
Suggestions
Terminology
Translation suggestions require that source text is available. They don't work if only IDs without the actual text are used.
Note that for singular and plural messages I am passing in the singular then the plural original text with a d% to represent the count
Source text ID
PHP format
17 | 19
Singular
You have %d message
Plural
You have %d messages
Translation
Needs work
Singular
Plural
Tienes %d mensaje
The translation therefore has a singular and plural form
Add comment
Translated: 4 of 4 (100 %)

```
msgid ""
msgstr ""
"Project-Id-Version: \n"
"POT-Creation-Date: 2025-09-02 21:48+0100\n"
"PO-Revision-Date: 2025-09-02 21:52+0100\n"
"Last-Translator: \n"
"Language-Team: \n"
"Language: es\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"
"X-Generator: Poedit 3.7\n"
"X-Poedit-Basepath: ../../..\n"
"X-Poedit-KeywordsList: __\n"
"X-Poedit-SearchPath-0: index.php\n"

#: index.php:34
msgid "Example"
msgstr "Ejemplo"

#: index.php:38
msgid "Home"
msgstr "Inicio"

#: index.php:39
#, php-format
msgid "Welcome %s"
msgstr "Bienvenido %s"
```

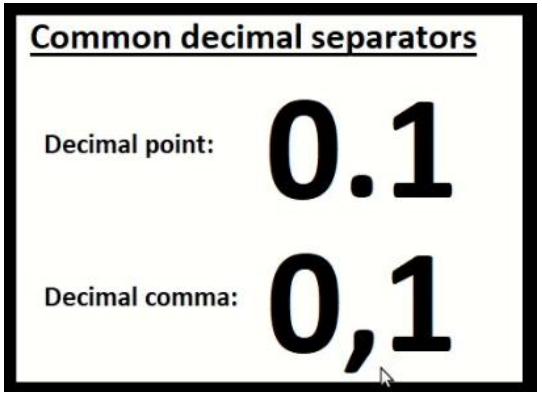
```
#: index.php:40
#, php-format
msgid "You have %d message"
msgid_plural "You have %d messages"
msgstr[0] "Tienes %d mensaje"
msgstr[1] "Tienes %d mensajes"

#~ msgid "Hello and welcome!"
#~ msgstr "¡Hola y bienvenido!"

#~ msgid "Thank You"
#~ msgstr "Gracias"
```

2. For plurals the original message and translation has the singular and plural with a %d denoting the integer value.

1. Note how string variables are shown in the .mo file. It uses the %s noation.

6.3 Decimal Seperators: Format Decimal Numbers Based on Locale

In English speaking countries and much of the world a dot is used to represent a decimal separator but in latin based languages such as Spanish, a coma is used to represent a decimal separator.




```php
$translator = new
PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

$name = "Dave";
$count = 1;
$pi = 3.14259;

setlocale(LC_ALL, 'es_ES.UTF-8');

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= sprintf($translator->gettext("Welcome %s"), $name) ?></p>
  <p><?= sprintf($translator->ngettext("You have %d message", "You have %d
messages", $count), $count) ?></p>
  <p><?= $pi ?></p>
</body>
</html>
```

Depending on how the server is setup this will probably not work, even though the locale is manually set to Spanish, the $pi is still displaying with a dot decimal separator and not a coma.

```php
$formatter = new NumberFormatter($locale, NumberFormatter::DECIMAL);

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= sprintf($translator->gettext("Welcome %s"), $name) ?></p>
  <p><?= sprintf($translator->ngettext("You have %d message", "You have %d messages", $count), $count) ?></p>
  <p><?= $formatter->format($pi) ?></p>
</body>

</html>
```

The PHP built in **number formatter class** can be used to format numbers according to locale where I invoke the class then use the format method later

Inicio

Bienvenido Dave

Tienes 1 mensaje

3,143

However, it rounds the number to three decimal places which is the default. I can manually specify this attribute.

```php
$formatter = new NumberFormatter($locale, NumberFormatter::DECIMAL);
$formatter->setAttribute(NumberFormatter::FRACTION_DIGITS, 5);
```

6.4 Translate Day & Month names in Dates based on Locale

```
<p><?= strftime("%A, %d %B %Y", $timestamp) ?></p>
```

```
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

$name = "Dave";
$count = 1;
$pi = 3.14259;
$timestamp = strtotime('20 July 1969');

$formatter = new NumberFormatter($locale, NumberFormatter::DECIMAL);
$formatter->setAttribute(NumberFormatter::FRACTION_DIGITS, 5);

$date_formatter = new IntlDateFormatter($locale, null, null);
$date_formatter->setPattern('EEEE, d MMMM Y');

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= sprintf($translator->gettext("Welcome %s"), $name) ?></p>
  <p><?= sprintf($translator->ngettext("You have %d message", "You have %d messages", $count), $count) ?></p>
  <p><?= $formatter->format($pi) ?></p>
  <p><?= $date_formatter->format($timestamp) ?>
</body>

</html>
```

The **php strftime function** will only work if the server locale is set so itf it is a server for which you do not have control such as shared hosting then this will not work. Additionally, this function has been deprecated as pof PHP 8.1.0. it is much better to use the built in PHP **IntlDateFormatter** class.

7.1 Handle Long Strings of Text in Different Files

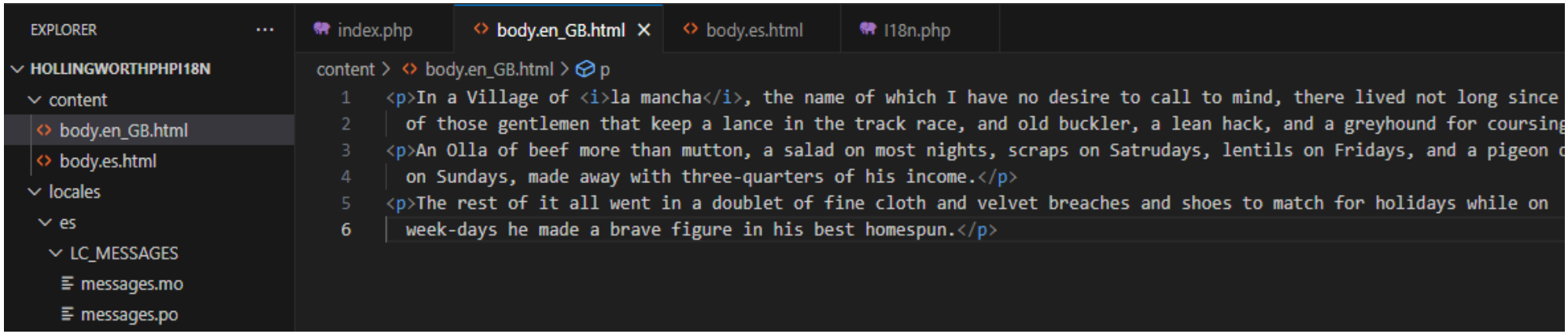
The **GNU documentation, section 4.3.3** recommends that messages should not be more than one paragraph long.

“4.3.3 Split at paragraphs
Translatable strings should be limited to one paragraph; don’t let a single message be longer than ten lines. The reason is that when the translatable string changes, the translator is faced with the task of updating the entire translated string. Maybe only a single word will have changed in the English string, but the translator doesn’t see that (with the current translation tools), therefore she has to proofread the entire message.

Many GNU programs have a ‘--help’ output that extends over several screen pages. It is a courtesy towards the translators to split such a message into several ones of five to ten lines each. While doing that, you can also attempt to split the documented options into groups, such as the input options, the output options, and the informative output options. This will help every user to find the option he is looking for.”

```php
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <h1><?= $translator->gettext('Home') ?></h1>

  <p>In a Village of <i>la mancha</i>, the name of which I have no desire to call to mind, there lived not long since one of those gentlemen that keep a lance in the track race, and old buckler, a lean hack, and a greyhound for coursing</p>
  <p>An Olla of beef more than mutton, a salad on most nights, scraps on Satrudays, lentils on Fridays, and a pigeon or so on Sundays, made away with three-quarters of his income.</p>
  <p>The rest of it all went in a doublet of fine cloth and velvet breaches and shoes to match for holidays while on week-days he made a brave figure in his best homespun.</p>

</body>
</html>
```

Instead of having a multi paragraph block of HTML text in the document, a separate folder called content is created and inside are files named by the body dot country code underscore regional code dot html.These files will contain the text blocks.



```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);
// list($subdomain, $domain) = array_pad(explode('.', $_SERVER['HTTP_HOST'], 2), 2, null);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {
  $locale = $i18n->getLocaleForRedirect();
  $subdomain = substr($locale, 0, 2);
  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}

$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <?php require "content/body.$locale.html" ?>
</body>

</html>
```

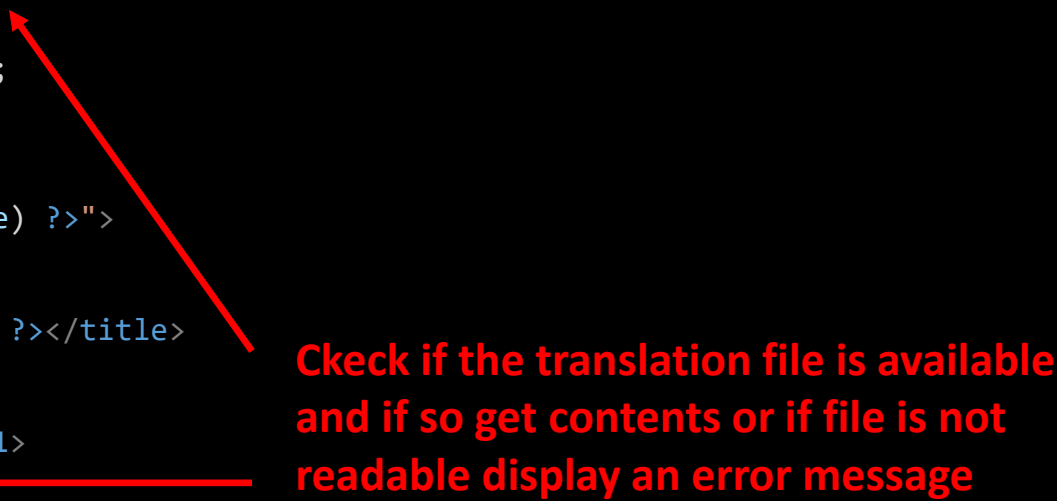
The text block from the specific language file can be included in the HTML

7.2 Display Message if File Containing Translation is Unavailable

If the translation file is unavailable, then PHP will throw an error.

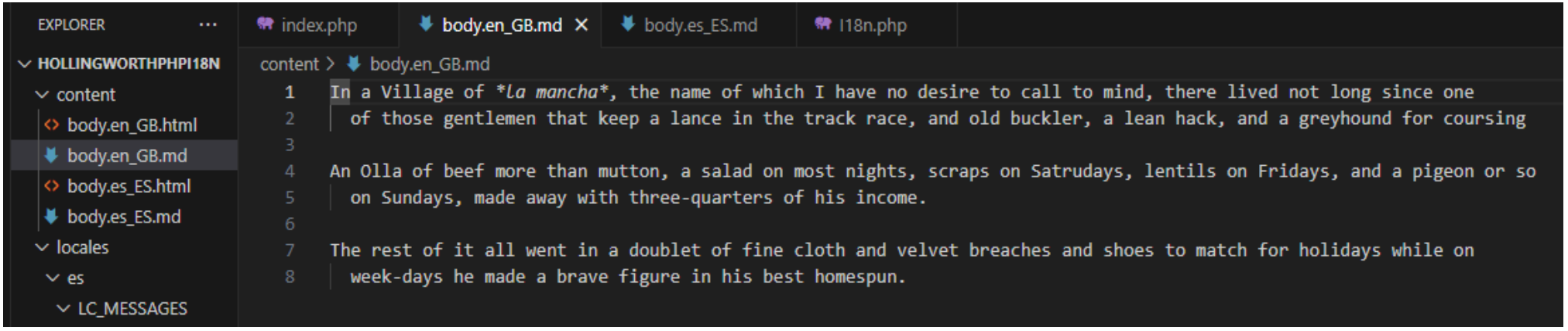
```
Inicio

Warning: require(content/body.es.html): Failed to open stream: No such file or
directory in C:\xampp\htdocs\hollingworthPHPi18n\index.php on line 36

Fatal error: Uncaught Error: Failed opening required 'content/body.es.html'
(include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\hollingworthPHPi18n\index.php:36
Stack trace: #0 {main} thrown in C:\xampp\htdocs\hollingworthPHPi18n\index.php on line
36
```

```php
<?php
require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {
  $locale = $i18n->getLocaleForRedirect();
  $subdomain = substr($locale, 0, 2);
  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");
$filename = "content/body.$locale.html";
if (is_readable($filename)) {
  $content = file_get_contents($filename);
} else {
  $content = "Content for $locale not found";
}
?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <?= $content ?>
</body>
</html>
```

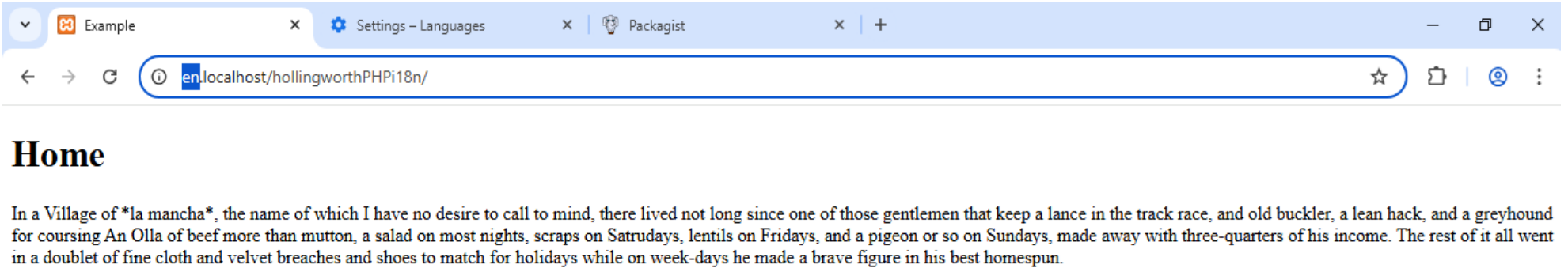


7.3 Use Plain Text Formatting Language to Help Translators

If passing a large text file containing HTML tags to a human translator, it is possible that they would not understand complex HTML markup so using simple **markdown** uses single characters to represent tags then a parser is used to convert it into HTML

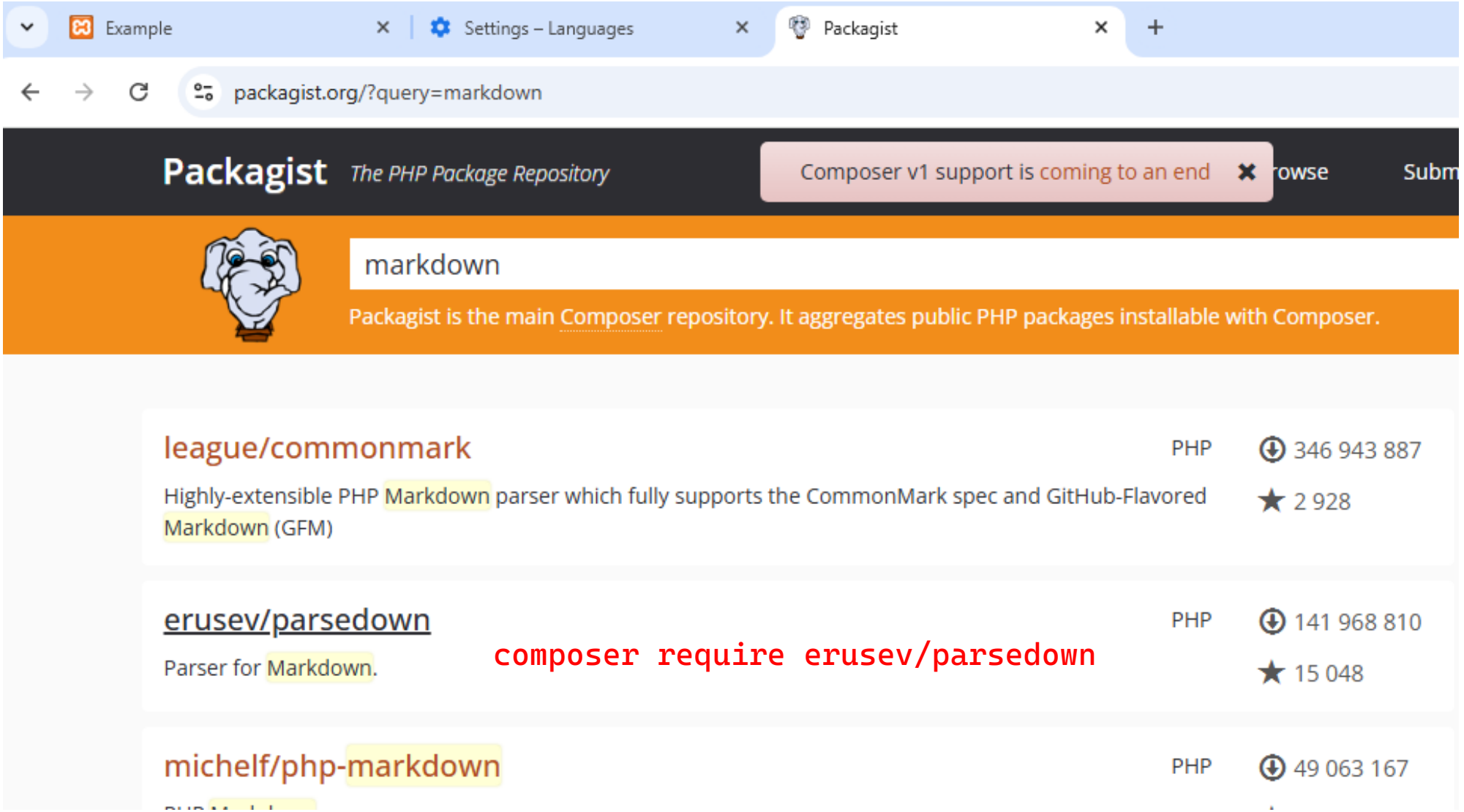


In Markdown, a line breaks denotes paragraphs and asterix is used to denote emphasis.



7.3 Use a Markdown Parser to Convert to HTML

In [Packagist](), I am going to use the erusev/parsedown parser because it is one of the most popular.



```php
<?php
require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);
list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {
  $locale = $i18n->getLocaleForRedirect();
  $subdomain = substr($locale, 0, 2);
  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");
$filename = "content/body.$locale.md";
if (is_readable($filename)) {
  $parser = new Parsedown;
  $content = file_get_contents($filename);
  $content = $parser->text($content);
} else {
  $content = "Content for $locale not found";
}
?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <?= $content ?>
</body>
</html>
```

I can instantiate the parser class and pass the contets of the text file to the parser. Now it will render the text with HTML tags.

7.4 Translate Multiple columns from a Database Table

	id	title_en_GB	title_es	description_en_GB	description_es	size
Edit Copy Delete	1	Thing	Cosa	This is a thing	Esto es una cosa	20
Edit Copy Delete	2	Wotsit	Chisme	A high quality wotsit	Un chisme de alta calidad	30

```php
<?php

require 'src/App/I18n.php';
require __DIR__  . '/vendor/autoload.php';

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}
```

```php
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

$conn = new PDO('mysql:host=localhost;dbname=hollingworthphpi18n_2;charset=utf8', 'root', '');
$sql = "SELECT title_$locale AS title, description_$locale AS description, size
        FROM product";
$stmt = $conn->query($sql);
$products = $stmt->fetchAll();

?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <table border="1">
    <thead>
      <th><?= $translator->gettext('Title') ?></th>
      <th><?= $translator->gettext('Description') ?></th>
      <th><?= $translator->gettext('Size') ?></th>
    </thead>
    <tbody>
      <?php foreach ($products as $product): ?>
        <tr>
          <td><?= $product['title'] ?></td>
          <td><?= $product['description'] ?></td>
          <td><?= $product['size'] ?></td>
        </tr>
      <?php endforeach; ?>
    </tbody>
  </table>
</body>
</html>
```

Using PDO object to connect to the database where there is a separate column for each desired language

These translations come from the .mo file

The column data is using the aliased column name.

es.localhost/hollingworthPHPi18n/

Inicio

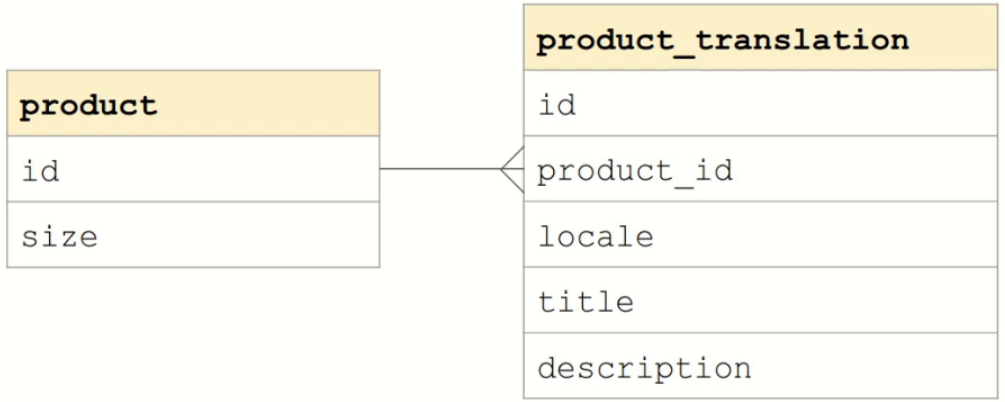
Título	Descripción	Tamaño
Cosa	Esto es una cosa	20
Chisme	Un chisme de alta calidad	30

en.localhost/hollingworthPHPi18n/

Home

Title	Description	Size
Thing	This is a thing	20
Wotsit	A high quality wotsit	30

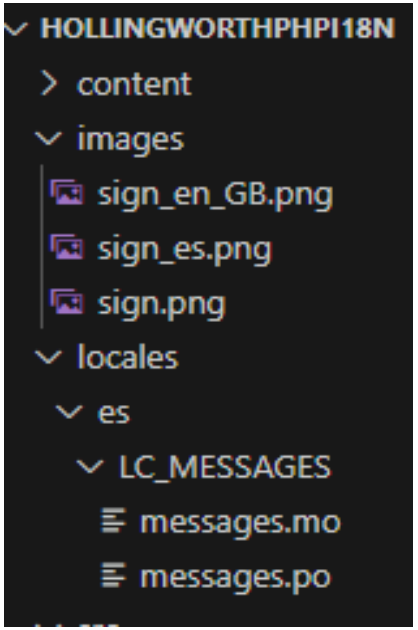
Using a separate column in he table for each language is not very scalable if many languages are included so another method might be to have a separate table for each language linked the master table via a foreign key.



7.5 Display Localised version Images That Contain Text

Here is an image that contains English text, if we wanted to display this in another language then I would need another image with the text in that language.

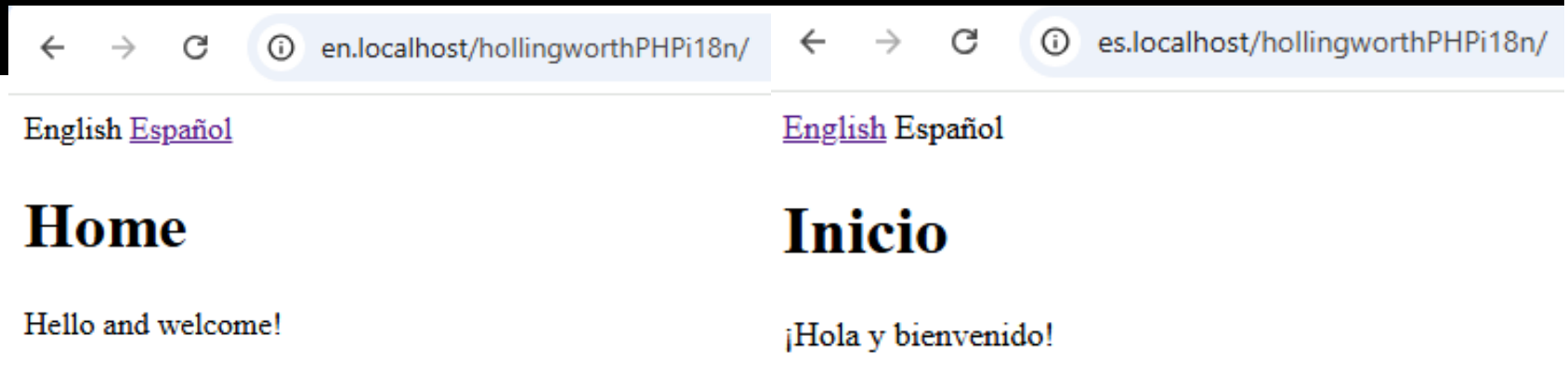

```php
$translator = new
PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");
?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <h1><?= $translator->gettext('Home') ?></h1>
  <img src="./images/sign_<?= $locale ?>.png">

</body>

</html>
```



8.1 Add Navigation Links to Switching Between Languages

```php
$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");
?>
<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">
<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>
<body>
  <nav>
    <?php if ($locale == 'en_GB'): ?>
      English
    <?php else: ?>
      <a href="http://en.localhost/hollingworthPHPi18n/">English</a>
    <?php endif; ?>
    <?php if ($locale == 'es'): ?>
      Español
    <?php else: ?>
      <a href="http://es.localhost/hollingworthPHPi18n/">Español</a>
    <?php endif; ?>
  </nav>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= $translator->gettext('Hello and welcome!') ?></p>
</body>
</html>
```

If the locale is en_GB then a link to the Spanish page is added. If the language is Spanish then a link to the en_GB page is added.



8.2 Language Switching Links: What They Should Say & Where To Put Them

Why flags do not represent languages

https://www.flagsarenotlanguages.com/blog/why-flags-do-not-represent-language/

Flags are symbols that represent countries or nations.

Languages represent a shared method of communication between people.

Flags are unique to a country or nation: but languages are often spoken across national borders. By using a flag for a language, you may **confuse** or even **offend** users.

8.3 Remove Code Duplication: Extract Common i18n Code Out To Separate Files

From the index.php file, I have removed the part where I am explicitly requiring the i18n class. (`require 'src/App/I18n.php';`) because I am using composers autoloader so I can use that instead.
I need to modify the composer.json file to add a section to include the autoloader. Then from the command line re-run the composer with a `composer dump-autoload` command.

```
{
    "require": {
        "ipinfo/ipinfo": "^3.2",
        "phpmyadmin/motranslator": "^5.4",
        "erusev/parsedown": "^1.7"
    },
    "autoload": {
        "psr-4": {
            "": "src"
        }
    }
}
```

Root/includes/i18n_init.php

```
<?php

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);
$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {
  $locale = $i18n->getLocaleForRedirect();
  $subdomain = substr($locale, 0, 2);
  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}

$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");
```

I cut out the top php block of locale determining code from the index.php file and put it in a new includes file.

Root/includes/lang_nav.php

```php
<nav>
  <?php if ($locale == 'en_GB'): ?>
    English
  <?php else: ?>
    <a href="http://en.localhost/hollingworthPHPi18n/">English</a>
  <?php endif; ?>
  <?php if ($locale == 'es'): ?>
    Español
  <?php else: ?>
    <a href="http://es.localhost/hollingworthPHPi18n/">Español</a>
  <?php endif; ?>
</nav>
```

I cut out the top html code for the language navigation and put it into a new includes file

Root/index.php

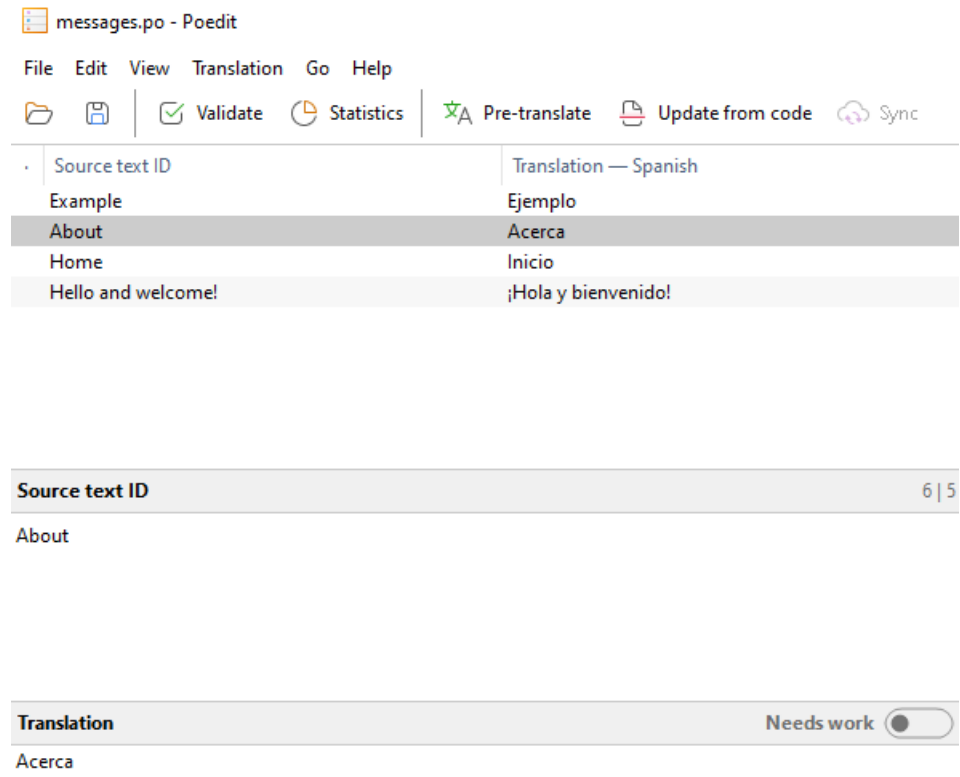
```php
<?php
require __DIR__  . '/vendor/autoload.php';
require './includes/i18n_init.php';
?>

<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <?php require './includes/lang_nav.php' ?>
  <h1><?= $translator->gettext('Home') ?></h1>
  <p><?= $translator->gettext('Hello and welcome!') ?></p>
</body>

</html>
```

Now the index.php file is a lot cleaner and I just require the i18n_init.php at the top and require the lang_nav.php at the top of the body of the HTML.

8.4 Add a Second Page and include Common i18n Code

```php
<?php
require __DIR__  . '/vendor/autoload.php';
require './includes/i18n_init.php';
?>

<!DOCTYPE html>
<html lang="<?= str_replace('_', '-', $locale) ?>">

<head>
  <meta charset="UTF-8">
  <title><?= $translator->gettext('Example') ?></title>
</head>

<body>
  <?php require './includes/lang_nav.php' ?>
  <h1><?= $translator->gettext('About') ?></h1>
</body>

</html>
```

Root/about.php



I added the about.php page to the source paths in POEDIT then updaded the translations for the about page saving them to the .PO file then converting to .MO file.

8.6 Generate the Data for Language navigation Links from $_SERVER data

```php
<?php

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {
  $locale = $i18n->getLocaleForRedirect();
  $subdomain = substr($locale, 0, 2);
  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}

$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

$link_data = $i18n->getLinkData(['en' => 'English', 'es' => 'Español']);
```

I call a method in the i18n class to getLinkData and I pass in a MD array of locales and languages

```php
public function getLinkData(array $languages)
{
  $protocol = isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? 'https' : 'http';
  $port = $_SERVER["SERVER_PORT"] == "80" ? '' : ":{$_SERVER['SERVER_PORT']}";
  $hostname_parts = explode('.', $_SERVER['HTTP_HOST'], 2);
  $url = $protocol . '://' . '%s.' . $hostname_parts[1] . $port . $_SERVER['REQUEST_URI'];
  $data = [];

  foreach ($languages as $code => $label) {

    $data[] = [
      'url' => sprintf($url, $code),
      'label' => $label,
      'is_current' => $code == $hostname_parts[0]
    ];
  }

  return $data;
}
```

In the i18n class, I use the $_SERVER superglobal to get build the URL based on the language of the subdomain, protocol, port and URI

Next, I build an MD array containing URL, Label and is current (bool)

8.7 Add Language Navigation Links to HTML

```php
<nav>
  <?php foreach ($link_data as $link): ?>

    <?php if ($link['is_current']): ?>

      <?= $link['label'] ?>

    <?php else: ?>

      <a href="<?= $link['url'] ?>"><?= $link['label'] ?></a>

    <?php endif; ?>

  <?php endforeach; ?>
</nav>
```

Now I can use the link data MD array to programmatically build the language links

8.8 Remember Selected Locale in a Cookie

```php
<?php

$i18n = new App\I18n(['en_GB', 'es']);

list($subdomain, $domain) = explode('.', $_SERVER['HTTP_HOST'], 2);

$locale = $i18n->getBestMatch($subdomain);

if ($locale === null) {

  $locale = $i18n->getLocaleForRedirect();

  $subdomain = substr($locale, 0, 2);

  header("Location: http://" . $subdomain . ".localhost/hollingworthPHPi18n");
  exit;
}

$translator = new PhpMyAdmin\MoTranslator\Translator("locales/$locale/LC_MESSAGES/messages.mo");

$link_data = $i18n->getLinkData(['en' => 'English', 'es' => 'Español']);

setcookie('locale', $locale, time() + 60 * 60 * 24 * 15, "/", "", "", TRUE);
```

To remember the language preference of the user, I am setting a cookie in the browser that has a life of 15 days.

8.9 Redirect to the Locale Stored in the Cookie

```php
private function getBestMatchFromCookie()
  {
    if (isset($_COOKIE['locale'])) {
      return $this->getBestMatch($_COOKIE['locale']);
    }
    return null;
  }

  public function getLocaleForRedirect()
  {
    // 1. Detect Language from Cookie
    $locale = $this->getBestMatchFromCookie();
    if ($locale !== null) {
      return $locale;
    }

    // 2. Detect language from HTTP header
    $locale = $this->getBestMatchFromHeader();
    if ($locale !== null) {
      return $locale;
    }

    // 3. detect language from Geolocation IP address
    $locale = $this->getBestMatchFromIPAddress();
    if ($locale !== null) {
      return $locale;
    }

    // 4. if unable to determine languege use default
    return $this->getDefault();
  }
```

In the i18n class, I add a private function to retrieve and return the cookie locale value if the cookie is set.

Now in the getLocaleForRedirect function, I detect the language from the cookie as a first preference so that if someone has visited the sight before then the language they were last browsing in is the language that the site will automatically load in.

The End