Creating a multilanguage application

Every day, we meet more and more international companies, software products, and information resources  
that publish content on multiple languages. Yii2 provides built-in i18n support for making multilanguage  
applications.

In this recipe, we are translating the application interface to different languages.

Подготовка

Создайте новое yii2-app-basic приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
<http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation>

How to do it...

1. Change the main menu labels in the views/layouts/main. php file to use the Yii: : t(' app/nav',  
   '...') method:

echo Nav::widget([

'options' => ['class' => 'navbar-nav navbar-right'],

'items' => [

['label' => Yii::t('app/nav', 'Home'), 'url' => ['/site/index']],

['label' => Yii::t('app/nav', 'About'), 'url' => ['/site/about']],

['label' => Yii::t('app/nav', 'Contact'), 'url' => ['/site/contact']],

], " '

]);

1. Change all your titles and breadcrumbs to use the common Yii ::t(' app, '...') method:

$this->title = Yii::t('app', 'Contact');

$this->params['breadcrumbs'][] = $this->title;

1. Also, change all the labels of your buttons:

<div class="form-group">

<?= Html::submitButton(Yii::t('app', 'Submit'), ['class' => 'btn btn-  
primary'']) ?>

</div>

Change other hard-coded messages as well:

<p>

<?= Yii::t('app', 'The above error occurred while the Web server was processing  
your request.') ?>

</p>

1. Change the attribute labels of your ContactForm model:

class LoginForm extends Model  
{

public function attributeLabels()

{

return [

'username' => Yii::t('app/user', 'Username'),

'password' => Yii::t('app/user', 'Password'),

'rememberMe' => Yii::t('app/user', 'Remember Me'),

];

}

}

Also, change the attribute labels of the LoginForm model:

class ContactForm extends Model  
{

public function attributeLabels()

{

return [

'name' => Yii::t('app/contact', 'Name'),

'email' => Yii::t('app/contact', 'Email'),

'subject' => Yii::t('app/contact', 'Subject'),

'body' => Yii::t('app/contact', 'Body'),

'verifyCode' => Yii::t('app', 'Verification Code'),

];

}

}

It will output translated labels for the current language instead of originals.

1. To prepare translations, create the messages directory. Right now, we can create translation files  
   for all needed languages. We can do it manually, but there is a helpful crawler that can scan all  
   project files and extract all messages from Yii ::t() constructions. Let’s use it.
2. Generate the configuration file for the message scanner:

./yii message/config-template config/messages.php

1. Open the configuration file and set the following values:

<?php  
return [

'sourcePath' => '@app',

'languages' => ['de', 'fr'],

'translator' => 'Yii::t',

'sort' => false,

'removeUnused' => false,

'markUnused' => true,

'only' => ['\*.php'],

'except' => [

'.svn',

'■git',

'■gitignore',

'.gitkeep',

'■hgignore',

' . hgkeep',

'/messages',

'/vendor',

],

'format' => 'php',

'messagePath' => '@app/messages',

'overwrite' => true,

'ignoreCategories' => [

'yii',

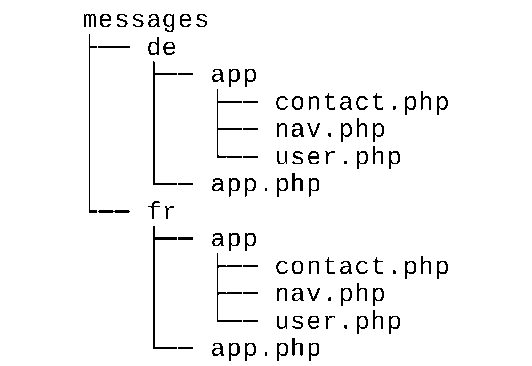
],

];

1. Run crawler while passing this configuration file to it:

./yii message config/messages.php

1. After the process, we must get the following directory structure:



1. For example, the messages/de/app/contact file contains the following content:

<?php...  
return [

'Body' => ' ',

' Email' => '',

' Name' => ' ',

'Subject' => '',

];

1. It is a plain PHP array with original sentences in keys and translated messages in values.
2. Just put in the values needed to translate messages from Deutsch:

<?php.  
return [

'Password' => 'Passwort',

'Remember Me' => 'Erinnere dich an mich',

'Username' => 'Benutzername',

];

1. Attach these translations to the ii8n component of application in the config/web. php file:

$config = [

'id' => 'basic',

'basePath' => dirname( DIR ),

'bootstrap' => ['log'],

'components' => [

'i18n' => [

'translations' => [

'app\*' => [

'class' => 'yii\i18n\PhpMessageSource',

'sourceLanguage' => 'en-US',

],

],

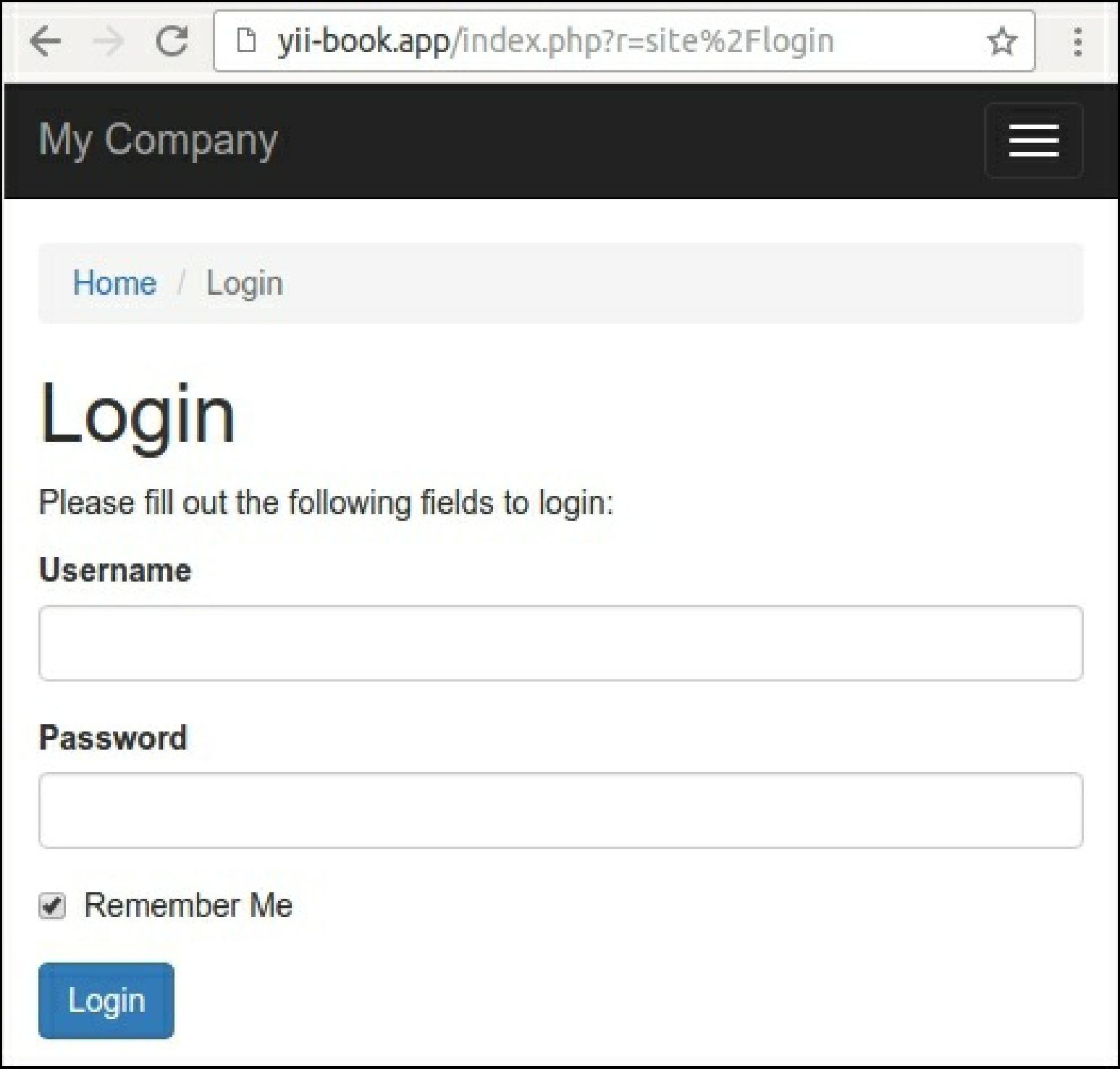
],

'db' => require( DIR . '/db.php'),

],

'params' => $params,

];



14. Open the login page with the default language:

15. Switch the application language to de:

$config = [

'id' => 'basic',

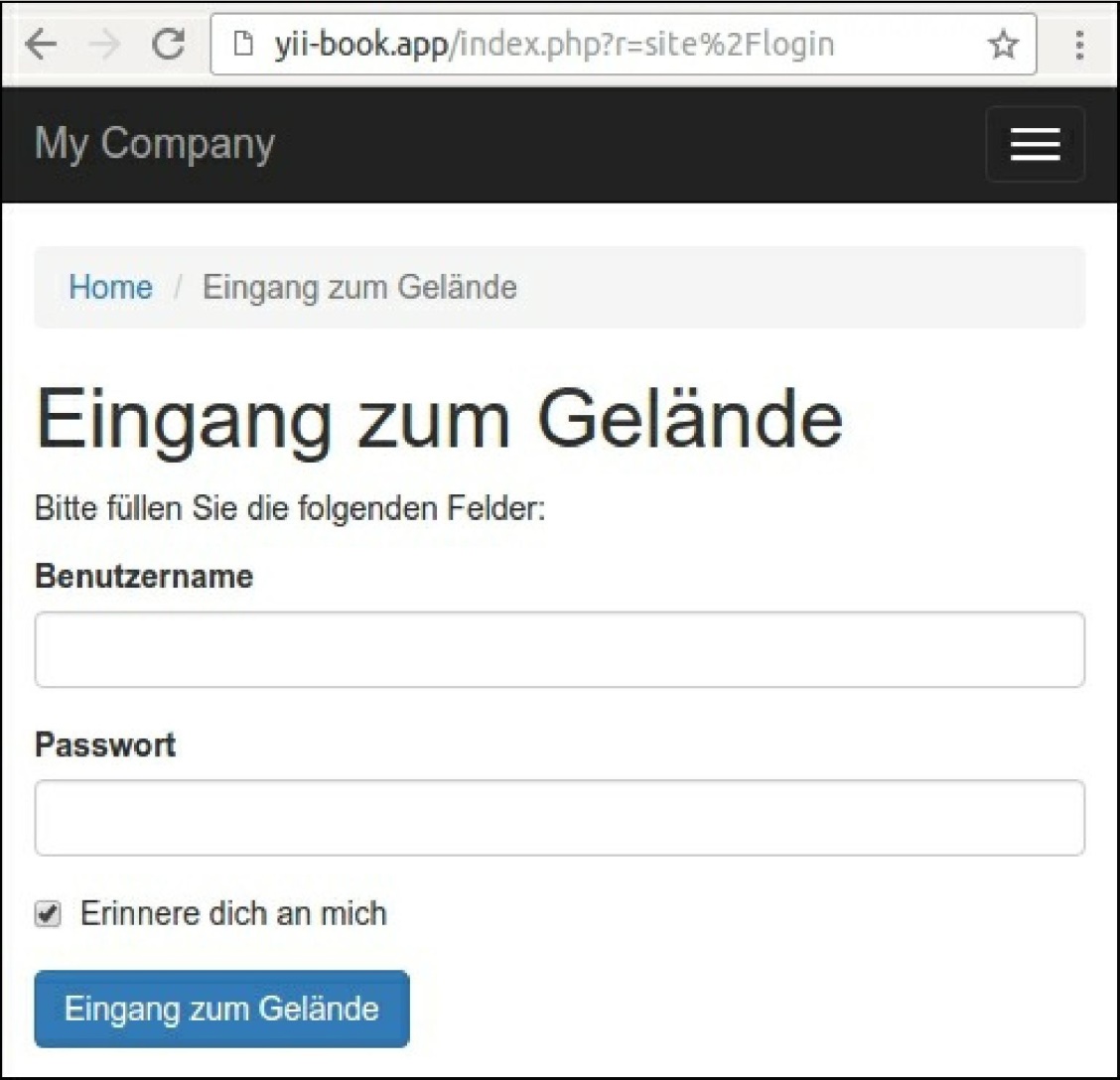
'language' => 'de',

'basePath' => dirname( DIR ),

'bootstrap' => ['log'],

];

Then refresh the login page:



16. The built-in frameworks messages and default validation errors will be translated automatically.

How it works...

Yii2 provides the Yii ::t() method for translating interface messages via the ii8n component, which  
supports different types of sources. In this recipe, we use yii\ii8n\hpMessageSource, which stores  
translated messages in plain PHP files.

The framework does not have artificial intelligence and does not translate messages by itself. You must  
put prepared translations in files or in the database and framework to get the needed message from this  
message source.

You can set the current language manually:

$config = [

'id' => ' basic',

' language' => 'de',

];

Instead of setting the language in the configuration file, you can switch the application language in runtir

Yii::$app->language = 'fr';

For example, if you store the user language in the lang field of the User model, you can create the  
language loader:

<?php

namespace app\bootstrap;

use yii\base\BootstrapInterface;

class LanguageBootstrap implements BootstrapInterface  
{

public function bootstrap($app)

{

if (!$app->user->isGuest) {

$app->language = $app->user->identity->lang;

}

}

}

Register this class in the bootstrapping list:

$config = [

'id' => ' basic',

'basePath' => dirname( DIR ),

'bootstrap' => ['log', 'app'bootstrap\LanguageBoostrap'],

];

Now, every authenticated user will see the interface in their own language.

Also, you can override the yii\web\UrlManager class for passing the current language as a GET  
parameter or as a prefix of a URL. Also, as an alternative you can store selected languages in browser  
cookies.

When you generate models and another code with Gii, you can check the following option:



All labels in the generated code will be embraced into the Yii: :t() calls.

Note

We did not cover the translating of model content in this recipe. However, for example, you can store

translated texts in separate tables (such as the post\_lang table for post model table) in a database and use  
the value of the Yii: :$app->language property to get the current language and extract needed content  
for your models by the value.

See also

[For more information about internationalization in Yii2, refer to http://www.yiiframework.com/doc-](http://www.yiiframework.com/doc-2.0/guide-tutorial-i18n.html)  
2.0/guide-tutorial-i18n.html.