Episode 1 - Installing and setting up your blog

Hello,

In this tutorial we will create our own SEO friendly blog based on Yii v2 framework. For more information please check Episode 0 - What you will create.

Prereq

This tutorial series gives you the step by step instructions on how to create awesome Yii applications. You will learn how to work with Yii 2 framework by examples. It is really simple, but I assume that you have at least some experience with OOP PHP, MySQL, HTML and JavaScript already. If you will have any questions or if you will need deeper explanation, please leave your questions in the comments section below this article.

Installation

So let's start, first we need to install and setup our Yii application project. For this project we will use the advanced application template. If you need more information what is Yii Advanced application template and how to install Composer, please visit our another episode: [Yii2 Essentials - Yii2 Installation](#).

If you have already installed Composer and you are familiar with Yii application templates, please continue reading here. Now you need to open console and **navigate to the root directory of your web server**, then run these commands in console:

```
composer global require "fxp/composer-asset-plugin:~1.1.1"
```

and

```
composer create-project yiisoft/yii2-app-advanced blog 2.0.8
```

We must initialize our application after installation. Advanced application template have two default environments: `dev` and `prod`. First is for development. It has all the developer tools and debug turned on. Second is for production server deployments. It has debug and developer tools turned off.

Type and run these commands:

```
cd blog
```

and

```
php init
```

To choose `dev` environment type "0" and press RETURN. After this you must confirm your selection. To confirm type "yes" and than press RETURN again.

You should see something like this:

```
Yii Application Initialization Tool v1.0

Which environment do you want the application to be initialized in?

   [0] Development
   [1] Production

   Your choice [0-1, or "q" to quit] 0

   Initialize the application under 'Development' environment? [yes|no] yes

   Start initialization ...

    generate backend/config/main-local.php
    generate backend/config/params-local.php
    generate backend/web/index-test.php
    generate backend/web/index.php
    generate common/config/main-local.php
    generate common/config/params-local.php
    generate console/config/main-local.php
    generate console/config/params-local.php
    generate frontend/config/main-local.php
    generate frontend/config/params-local.php
    generate frontend/web/index-test.php
    generate frontend/web/index.php
    generate yii
    generate cookie validation key in backend/config/main-local.php
    generate cookie validation key in frontend/config/main-local.php
       chmod 0777 backend/runtime
       chmod 0777 backend/web/assets
       chmod 0777 frontend/runtime
       chmod 0777 frontend/web/assets
       chmod 0755 yii
       chmod 0755 tests/codeception/bin/yii

   ... initialization completed.
```

Now you should be able to go to

```
http://localhost/blog/frontend/web
```

And you should see this:

Setting basic things

.htaccess

We do not want to point to `/frontend/web` folder manually every time. To do this automatically we must create 3 new .htaccess files. First we will create in our application root directory (`/.htaccess`) with this content:

```apache
<IfModule mod_autoindex.c>

    Options -Indexes

</IfModule>



<IfModule mod_rewrite.c>

    Options +FollowSymlinks

    RewriteEngine On


    RewriteCond %{REQUEST_URI} ^/backend

    RewriteRule ^backend/(.*)$ backend/web/$1 [L]



    RewriteCond %{REQUEST_URI} !^public

    RewriteRule ^(.*)$ frontend/web/$1 [L]



    RewriteCond %{HTTP_USER_AGENT} libwww-perl.*

    RewriteRule .* - [F,L]

</IfModule>



<Files ~ "(.json|.lock|.git)">

    Order allow,deny

    Deny from all

</Files>
```

```
RewriteRule (^\.|/\.) - [F]
```

Second file we will create in the `/frontend/web` directory (`/frontend/web/.htaccess`) and third in the `/backend/web` directory (`/backend/web/.htaccess`) with this (the same) content:

```
Options +FollowSymLinks
IndexIgnore */*


RewriteEngine on



RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d



RewriteRule . index.php
```

Now we should be able to go to this URL:

```
http:
```

and see Yii welcome page.

Pretty URLs

Front end

Below is sample of the current application URL address pointing to `actionIndex()` method of the `SiteController` class:

```
http://localhost/blog/frontend/web/index.php?r=site%2Findex
```

We do not want to have so long and ugly URLs. Also for the good position in search engines like Google (SEO) are pretty URLs necessary. To setup pretty URLs in advanced application template we need to edit `frontend` or `backend` config files. Now we only need to setup frontend routing. So, lets do it. Open your `/frontend/config/main.php` configuration file and add this code to components array:

```
    'components' => [



    'urlManager' => [

        'enablePrettyUrl' => true,

        'showScriptName' => false,

        'rules' => [

        ],

    ],



],
```

After saving this configuration file and refreshing our blog, we will see pretty URLs. The URL address bellow points to same controller and action as URL before, but is much more better readable.

```
http://localhost/blog/frontend/web/site/index
```

Back end

We do not need to set up pretty URLs for the `backend` application now. The reason is that only `frontend` will be accessible for standard users and for search engines. But if do you want to set up pretty URLs also for `backend`, just edit the `/backend/config/main.php` configuration file the same way as the `frontend` configuration file.

Application name

The name property specifies the application name that may be displayed to end users. We can set up this property in our `common/config/main.php` configuration file. Add this line of code after the `vendorPath` property:

```
'name' => 'My-Yii Blog',
```

Now we can get the application name with this code:

```
Yii::$app->name
```

Let's use it in our front end main layout file (`frontend/views/layouts/main.php`). Set the `brandLabel` property in `NavBar` widget like this:

```
'brandLabel' => Yii::$app->name,
```

and in our footer container:

```
<p class="pull-left">&copy; <?= Yii::$app->name ?> <?= date('Y') ?></p>
```

It is not necessary to configure this property if none of our code is using it, but I decided to configure it to help you understand Yii config files and theirs properties.

Database connection

Before setting up the database connection we have to create new database. So, create new MySQL database with name `blog`. Leave this database empty for now. We will create tables later.

Before proceeding, make sure you have installed both the PDO PHP extension and the PDO driver for the database you are using (e.g. pdo_mysql for MySQL). This is a basic requirement if your application uses a relational database. [1]

Updated 25.5.2016:

Instead of editing `common/config/main-local.php` file we will edit `environments/dev/common/config/main-local.php` configuration file [2]. It is better to do it this way because we will use these credentials only in local development. For production you will need to use another credentials.

Now we need to connect our blog to the MySQL database. To do this, we will edit the `environments/dev/common/config/main-local.php` file. Change the parameters to be correct for your database and MySQL server:

```
'db' => [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=blog',
    'username' => 'root',
    'password' => 'root',
    'charset' => 'utf8',
],
```
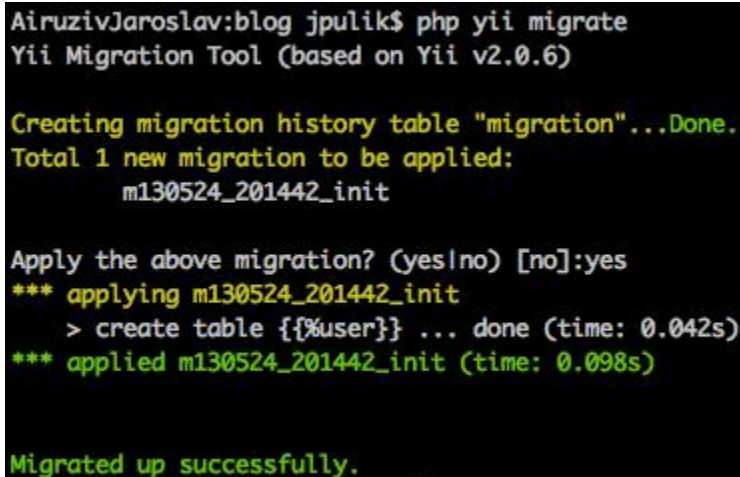
After this, run

```
php init
```

command again, choose 0, type *yes* and *All* to overwrite old config files and then press RETURN to confirm.

Testing connection

We will test our database connection by running Yii migration command trough console:

```
php yii migrate
```

Yii Migration Tool will ask you: Apply the above migration? Type "yes" and press RETURN. You should see:

```
AiruzivJaroslav:blog jpulik$ php yii migrate
Yii Migration Tool (based on Yii v2.0.6)

Creating migration history table "migration"...Done.
Total 1 new migration to be applied:
        m130524_201442_init

Apply the above migration? (yes|no) [no]:yes
*** applying m130524_201442_init
    > create table {{%user}} ... done (time: 0.042s)
*** applied m130524_201442_init (time: 0.098s)


Migrated up successfully.
```

If your migration was done successfully, you have just created User table in your `blog` database.

Congratulations! Your blog is now ready to Sign up and Log in users.

We will continue building our blog in next episode. If do you have any questions regarding to this episode, please write them below to the comments section.

Download files from this episode: [episode_01.zip](#).