Using multiple configurations to simplify the  
deployment

The Advanced application template uses different configuration files for each of its applications:

common

config

main . php  
main-local.php  
params.php  
params-local.php

console

config

main . php  
main-local.php  
params.php  
params-local.php

backend

config

main . php  
main-local.php  
params.php  
params-local.php

frontend

config

main . php  
main-local.php  
params.php  
params-local.php

Each entry web/index. php script merges own set of configuration files:

$config = yii\helpers\ArrayHelper::merge(

require( DIR . '/../../common/config/main.php'),

require( DIR . '/../../common/config/main-local.php'),

require( DIR . '/../config/main.php'),

require( DIR . '/../config/main-local.php')

);

$application = new yii\web\Application($config);

$application->run();

Each config/main. php file merges parameters:

<?php

$params = array\_merge(

require( DIR . '/../../common/config/params.php'),

require( DIR . '/../../common/config/params-local.php'),

require( DIR . '/params.php'),

require( DIR . '/params-local.php')

);

return [

// ...

'params' => $params,

];

This system allows you to configure both common and specific application properties and components of  
our applications. And we can store default configuration files on the version control system and ignore all  
the \* -local. php files.

All local files templates are prepared in the environments directory. When you run php init in your  
console and choose a needle environment, this initialization script makes copies of the corresponded files  
and places them into target folders.

But the Basic application template does not contain an agile configuration system and provides only the  
following files:

config

console.php  
web.php  
db.php  
params.php

Let’s try to add an advanced configuration system to the yii2 -app - basic application template.

Getting ready

Create a new yii2 -app-basic application by using the Composer package manager, as described in the  
official guide at <http://www.yiiframework.com/doc-2.0/guidestart-installation.html>.

How to do it...

1. Create the config/common.php file:

<?php

$params = array\_merge(

require( DIR . '/params.php'),

require( DIR . '/params-local.php')

);

return [

'basePath' => dirname( DIR ),

'components' => [

'cache' => [

'class' => 'yii\caching\FileCache',

],

'mailer' => [

'class' => 'yii\swiftmailer\Mailer',

],

'db' => [],

],

'params' => $params,

];

1. Create the config/common-local file:

<?php  
return [

'components' => [

'db' => [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=yii2basic',

'username' => 'root',

'password' => '',

' charset' => ' utf8',

],

'mailer' => [

'useFileTransport' => true,

],

],

];

1. Remove the config/db. php file.
2. Remove repetitive code from config/console. php:

<?php

Yii::setAlias('@tests', dirname( DIR ) . '/tests');

return [

'id' => 'basic-console',

'bootstrap' => ['log', 'gii'],

'controllerNamespace' => 'app\commands',

'modules' => [

'gii' => 'yii\gii\Module',

],

'components' => [

'log' => [

'targets' => [

[

'class' => 'yii\log\FileTarget',  
'levels' => ['error', 'warning'],

],

],

],

],

];

1. Create the config/console-local.php file with an empty array:

<?php  
return [

];

1. Change the config/web. php file:

$config = [

'id' => 'basic',

'bootstrap' => ['log'],

'components' => [

'user' => [

'identityClass' => 'app\models\User',  
'enableAutoLogin' => true,

],

'errorHandler' => [

'errorAction' => 'site/error',

],

'log' => [

'traceLevel' => YII\_DEBUG ? 3 : 0,

'targets' => [

[

'class' => 'yii\log\FileTarget',  
'levels' => ['error', 'warning'],

],

],

],

],

];

if (YII\_ENV\_DEV) {

// configuration adjustments for 'dev' environment  
$config['bootstrap'][] = 'debug';  
$config['modules']['debug'] = 'yii\debug\Module';

$config['bootstrap'][] = 'gii';  
$config['modules']['gii'] = 'yii\gii\Module';

}

return $config;

1. Move the request configuration into config/web-local.php:

<?php  
return [

'components' => [

'request' => [

'cookieValidationKey' => 'TRk9G1La5kvLFwqMEQTp6PmC1NHdjtkq',

],

],

];

1. Remove the e-mail ID from config/params .php:

<?php  
return [

'adminEmail' => '',

];

1. Paste the ID into config/params-local. php:

<?php  
return [

'adminEmail' => 'admin@example.com',

];

1. Remove the dsn string from tests/codeception/config/config .php:

<?php

/\*\*

\* Application configuration shared by all test types  
\*/

return [

'controllerMap' => [

// ...

],

'components' => [

'db' => [

'dsn' => '',

],

'mailer' => [

'useFileTransport' => true,

],

'urlManager' => [

'showScriptName' => true,

],

],

];

1. Put the string into a new tests/codeception/config/config -local. php file:

<?php  
return [

'components' => [

'db' => [

'dsn' => 'mysql:host=localhost;dbname=yii2\_basic\_tests',

],

],

];

1. Add configuration merging to the web/index. php file:

$config = yii\helpers\ArrayHelper::merge(

require( DIR . '/../config/common.php'),

require( DIR . '/../config/common-local.php'),

require( DIR . '/../config/web.php'),

require( DIR . '/../config/web-local.php')

);

1. Add configuration merging to the console entry script, yii:

$config = yii\helpers\ArrayHelper::merge(

require( DIR . '/config/common.php'),

require( DIR . '/config/common-local.php'),

require( DIR . '/config/console.php'),

require( DIR . '/config/console-local.php')

);

1. Add configuration merging to the testing configurations of the unit, functional, and acceptance tests  
   from tests/codeception/config:

return yii\helpers\ArrayHelper::merge(

require( DIR . '/../../../config/common.php'),

require( DIR . '/../../../config/common-local.php'),

require( DIR . '/../../../config/web.php'),

require( DIR . '/../../../config/web-local.php'),

require( DIR . '/config.php'),

require( DIR . '/config-local.php'),

[

// ...

]

);

1. Add configuration merging to the testing environment console’s entry script,  
   tests/codeception/bin/yii:

$config = yii\helpers\ArrayHelper::merge(

require(YII\_APP\_BASE\_PATH . '/config/common.php'),  
require(YII\_APP\_BASE\_PATH . '/config/common-local.php'),  
require(YII\_APP\_BASE\_PATH . '/config/console.php'),  
require(YII\_APP\_BASE\_PATH . '/config/console-local.php'),

require( DIR . '/../config/config.php'),

require( DIR . '/../config/config-local.php')

);

1. As a result, you must get the following content in your configuration directory:

config

common.php  
common-local.php  
console.php  
console-local.php  
web.php  
web-local.php  
params.php  
params-local.php

1. After all, you can add a new . gitignore file with this content into your config and  
   tests/codeception/config directories so you can ignore local configuration files by the Git  
   version control system:

/\*-local.php

How it works...

You can store common application components configuration in the config/common. php file and also set  
specific configurations for web and console applications. You can put your temporary and secure  
configuration data into the \*-iocal. php files.

Also, you can copy the initialization shell script from yii2-app-advanced.

1. Create a new environments directory and copy your templates into it:

environments

dev

config

common-local.php  
console-local.php  
web-local.php  
params-local.php

web

index.php  
index - test.php  
tests

codeception

config

config.php  
config-local.php

yii

prod

config

common-local.php  
console-local.php  
web-local.php  
params-local.php

web

index.php

yii

1. Create the environments/index.php file with this code:

<?php  
return [

'Development' => [

'path' => ' dev',

'setWritable' => [

'runtime',

'web/assets',

],

'setExecutable' => [

'yii',

'tests/codeception/bin/yii',

],

'setCookieValidationKey' => [

'config/web-local.php',

],

],

'Production' => [

'path' => ' prod ',

'setWritable' => [

'runtime',

'web/assets',

],

'setExecutable' => [

'yii',

],

'setCookieValidationKey' => [

'config/web-local.php',

],

],

];

1. Remove the default Installer : : postCreateProject configuration from your composer . json:  
   "extra": {

"asset-installer-paths": {

"npm-asset-library": "vendor/npm",

"bower-asset-library": "vendor/bower"

}

}

1. [Copy the init and init. bat scripts from the Advanced template, https ://github. c om/yiisoft/yii2-](https://github.com/yiisoft/yii2-app-advanced)  
   app-advanced and you can run the initialization process using the command php init after the  
   cloning of the project from the repository.

See also

[For more information about application configurations refer to http://www.yiiframework.com/doc-](http://www.yiiframework.com/doc-2.0/guide-concept-configurations.html)  
2.0/guide-concept-confjgnrations.html.