Combining and minimizing assets

If your web page includes many CSS and/or JavaScript files, the page will open very slowly because the  
browser sends a large number of HTTP requests to download each file in separated threads. To reduce  
the number of requests and connections, we can combine and compress multiple CSS/JavaScript files into  
one or very few files in production mode, and then include these compressed files on the page instead of  
the original ones.

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>
* Download the compiler. j ar file from <https://developers.google.com/closure/compiler/>
* Download the yuicompressor.jar file from [https://githiih.rom/yni/ynirompressor/releases](https://github.com/yui/yuicompressor/releases)
* Download and install the Java Runtime Environment (JRE) from <http://www.java.com>

How to do it...

Follow these steps to combine and minimize assets:

1. Open the source HTML code of the index page of your application. Check whether it is similar to  
   the following structure:

< ! DOCTYPE html>

<html lang="en-US">

<head>

<title>My Yii Application</title>

<link href="/assets/9b3b2888/css/bootstrap.css" rel="stylesheet">

<link href="/css/site.css" rel="stylesheet">

</head>

<body>

<script src="/assets/25f82b8a/jquery.js"></script>

<script src="/assets/f4307424/yii.js"></script>

<script src="/assets/9b3b2888/js/bootstrap.js"></script>

</body>

</html>

The page includes three JavaScript files.

1. Open the config/console. php file and add the @webroot and @web alias definitions:

<?php

Yii::setAlias('@webroot', DIR . '/../web');

Yii::setAlias('@web', '/');

1. Open a console and run the following command:  
   yii asset/template assets.php
2. Open the generated assets. php file and configure it as follows:

<?php  
return [

'jsCompressor' => 'java -jar compiler.jar --js {from} --js\_output\_file {to}',  
'cssCompressor' => 'java -jar yuicompressor.jar --type css {from} -o {to}',  
'bundles' => [

'app\assets\AppAsset',

'yii\bootstrap\BootstrapPluginAsset',

],

'targets' => [

'all' => [

'class' => 'yii\web\AssetBundle',

'basePath' => '@webroot/assets',

'baseUrl' => '@web/assets',

'js' => 'all-{hash}.js',

'css' => 'all-{hash}.css',

],

],

'assetManager' => [

'basePath' => '@webroot/assets',

'baseUrl' => '@web/assets',

],

];

5. Run the combining command yii asset assets.php config/assets-prod.php. If this is  
successful you must get the config/assets-prod. php file with the following configuration:

<?php  
return [

'all' => [

'class' => 'yii\\web\\AssetBundle',

'basePath' => '@webroot/assets',

'baseUrl' => '@web/assets',

'js' => [

' all-fe792d4766bead53e7a9d851adfc6ec2.js',

],

'css' => [

'all-37cfb42649f74eb0a4bfe0d0e715c420.css',

],

],

'yii\\web\\JqueryAsset' => [

'sourcePath' => null,

'js' => [],

' css ' => [],

'depends' => [

'all',

],

],

'yii\\web\\YiiAsset' => [

'sourcePath' => null,

'js' => [],

'css' => [],

'depends' => [

'yii\\web\\JqueryAsset',

'all',

],

],

'yii\\bootstrap\\BootstrapAsset' => [

'sourcePath' => null,

'js' => [],

' css ' => [],

'depends' => [

'all',

],

],

'app\\assets\\AppAsset' => [

'sourcePath' => null,

'js' => [],

'css' => [],

'depends' => [

'yii\\web\\YiiAsset',

'yii\\bootstrap\\BootstrapAsset',

'all',

],

],

'yii\\bootstrap\\BootstrapPluginAsset' => [

'sourcePath' => null,

'js' => [],

' css ' => [],

'depends' => [

'yii\\web\\JqueryAsset',

'yii\\bootstrap\\BootstrapAsset',

'all',

],

],

];

1. Add the configuration for the assetManager component into the config/web. php file:

'components' => [

// ...

'assetManager' => [

'bundles' => YII\_ENV\_PROD ? require( DIR . '/assets-prod.php') : [],

],

],

1. Turn on production mode in web/index .php:  
   defined('YII\_ENV') or define('YII\_ENV', 'prod');
2. Reload the page in your browser and see the HTML code again. Now it must contain single lines to  
   include our compressed files:

< ! DOCTYPE html>

<html lang="en-US">

<head>

<title>My Yii Application</title>

<link href="/assets/all-37cfb42649f74eb0a4bfe0d0e715c420.css"  
rel="stylesheet">

</head>

<body>

<script src="/assets/all-fe792d4766bead53e7a9d851adfc6ec2.js"></script>  
</body>

</html>

How it works...

First of all, our page had a set of included files:

<link href="/assets/9b3b2888/css/bootstrap.css" rel="stylesheet">

<link href="/css/site.css" rel="stylesheet">

<script src="/assets/25f82b8a/jquery.js"></script>

<script src="/assets/f4307424/yii.js"></script>

<script src="/assets/9b3b2888/js/bootstrap.js"></script>

Next, we generated the assets. php configuration file and specified bundles for compressing:

'bundles' => [

'app\assets\AppAsset',

'yii\bootstrap\BootstrapPluginAsset',

],

Note

Note: We could specify all intermediate asset bundles such as yii\web\jqueryAsset and  
yii\web\YiiAsset, but these assets are already specified as dependencies of AppAsset and  
BootstrapPluginAsset, and the compressing command automatically resolves all these dependencies.

The AssetManager publishes all assets into the classic subdirectories in web/assets and after publishing it  
runs compressors to combine all CSS and JS files into all-{hash}.js and all-{hash}.css.

Check whether the CSS file includes other resources by relative paths such as the bootstrap.css file:  
@font-face {

font-family: 'Glyphicons Halflings';

src: url('../fonts/glyphicons-halflings-regular.eot');

}

If it is so, then in the combined file, our compressor changes all relative paths for storing all relationships as  
follows:

@font-face{

font-family: 'Glyphicons Halflings';

src: url('9b3b2888/fonts/glyphicons-halflings-regular.eot');

}

After processing, we get the assets-prod. php file with the bundles configuration of the assetManager  
component. It defines the new virtual asset as a dependency of clean copies of the original bundles:

return [

'all' => [

'class' => 'yii\\web\\AssetBundle',

'basePath' => '@webroot/assets',

'baseUrl' => '@web/assets',

'js' => [

'all-fe792d4766bead53e7a9d851adfc6ec2 .js',

],

'css' => [

' all-37cfb42649f74eb0a4bfe0d0e715c420.css',

],

],

'yii\\web\\JqueryAsset' => [

'sourcePath' => null,

'js' => [],

'css' => [],

'depends' => [

'all',

],

],

// ...

]

Now we can require this configuration into the config/web. php file:

'components' => [

// ...

'assetManager' => [

'bundles' => require( DIR . '/assets-prod.php'),

],

],

Alternatively, we can require the file for the production environment only:

'components' => [

// ...

'assetManager' => [

'bundles' => YII\_ENV\_PROD ? require( DIR . '/assets-prod.php') : [],

],

],

Note

Note: Do not forget to regenerate all compressed and combining files after any updates of the original  
resources.

See also

* [For more information about assets refer to the following URL: http://www.yiiframework.com/doc-](http://www.yiiframework.com/doc-2.0/guide-structure-assets.html)  
  2.0/guide-structure-assets.html
* For Closure Compiler refer to the following URL: <https://developers.google.com/closure/compiler/>
* For YUI Compressor refer to the following URL: [https://github.com/vmyvnirompressor/](https://github.com/yui/yuicompressor/)