'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' => [