How it works...

When yii\_debug is set to false, Yii turns OFF all the trace level logging and uses less error handling code. Also, when you set yii\_env to prod your application does not load Yii and Debug panel modules.

Setting schemaCachingDuration to a number of seconds allows caching the database schema used by Yii’s Active Record. This is highly recommended for production servers and it significantly improves the Active Record performance. In order for it to work, you need to properly configure the cache component as follows:

'cache' => [

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

],

Enabling the cache also has a positive effect on other Yii components. For example, Yii router or urlManager starts to cache routes.

Of course, you can get into a situation where the preceding settings will not help to achieve a sufficient performance level. In most cases, it means that either the application itself is a bottleneck or you need more hardware.

• Server-side performance is just a part of the big picture: Server-side performance is only one of the things that affect the overall performance. By optimizing the client side such as serving CSS, images, and JavaScript files, proper caching and minimizing the amount of HTTP-requests can give a good visual performance gain even without optimizing the PHP code.

• Things to be done without using Yii: Some things are best done without Yii. For example, image resizing on-the-fly is better in a separate PHP script in order to avoid the extra overhead.

• Active Record versus Query Builder and SQL: Use Query Builder or SQL in performance- critical application parts. Generally, AR is most useful when adding and editing records, as it adds a convenient validation layer, and is less useful when selecting records.

• Always check for slow queries first: Database can become a bottleneck in a second if a developer accidentally forgets to add an index to a table that is being read often or vice versa, or adds too many indexes to a table we are writing to very often. The same goes for selecting unnecessary data and unneeded JOINs.

• Cache or save results of heavy processes: If you can avoid running a heavy process in every page load, it is better to do so. For example, it is a good practice to save or cache results of parsing the markdown text, purify it (this is a very resource-intensive process) once, and then to use the ready-to-display HTML.

• Handling too much processing: Sometimes there is too much processing to be handled immediately. It can be building complex reports or simply sending e-mails (if your project is heavily loaded). In this case, it is better to put it into a queue and process it later using cron or other specialized tools.

See also

For more information about performance tuning and caching refer to the following URLs: