Leveraging HTTP caching

Instead of only server-side caching implementation you can use client-side caching via specific HTTP- headers.

In this recipe, we will cover full-page caching on the basis of the Last-Modified and ETag headers.

Getting ready

Create a new yii2-app-basic application using the Composer package manager, as described in the official guide at [http://www.yijframework.com/doc-2.0/gurde-start-insta]lation.html](http://www.yiiframework.com/doc-2.0/guide-start-installation.html).

1. Create and run migration as follows:

<?php

use yii\db\Migration;

class m160308\_093233\_create\_example\_tables extends Migration {

public function up()

{

$this->createTable('{{%article}}', [

'id' => $this->primaryKey(),

'created\_at' => $this->integer()->unsigned()- >notNull(),

'updated\_at' => $this->integer()->unsigned()->notNull(),

'title' => $this->string()->notNull(),

'text' => $this->text()->notNull(),

]);

}

public function down()

{

$this->dropTable('{{%article}}');

}

}

2. Create an Article model as follows:

<?php

namespace app\models; use Yii;

use yii\behaviors\TimestampBehavior; use yii\db\ActiveRecord;

class Article extends ActiveRecord {

public static function tableName()

{

return '{{%article}}';

}

public function behaviors()

{

return [

TimestampBehavior::className(),

];

}