class Post extends \yii\db\ActiveRecord {

// . .

public function behaviors()

{

return [

[

'class' => 'yii\behaviors\TimestampBehavior',

'attributes' => [

BaseActiveRecord::EVENT\_BEFORE\_INSERT => 'creation\_date', BaseActiveRecord::EVENT\_BEFORE\_UPDATE => 'modified\_date',

]

]

];

}

// ..

}

In this example, we’ve pointed to the creation\_date and modified\_date attributes before creating and updating our model accordingly by dint of using special ActiveRecord events: event\_before\_insert and

EVENT\_BEFORE\_UPDATE.

**In addition...**

You may want to save the timestamp for custom scenarios. Let’s say you want to update the last\_login field, for example, for a specific controller action. In this situation, you can trigger the timestamp update for your specific attribute using the following:

$model->touch('last\_login');

Be aware that touch() can’t be used for new models. You will get InvalidCallException in this case:

$model = new Post();

$model->touch('creation\_date ' );

The touch () method calls model saving inside itself so you don’t need to save the model after calling it.

See also

[For further information, refer to http://www.yiiframework.com/doc-2.0/gurde-concept-](http://www.yiiframework.com/doc-2.0/guide-concept-behaviors.html%23using-timestampbehavior) behaviors.html#using-timestampbehav'or.