How it works...

Internally, during form rendering, we have code like this:

if ($request->enableCsrfValidation && !strcasecmp($method, 'post')) {

$hiddenInputs[] = static::hiddenInput($request->csrfParam, $request->getCsrfToken());

}

if (!empty($hiddenInputs)) {

$form .= "\n" . implode("\n", $hiddenInputs);

}

In the previous code, getcsrfToken () generates a unique token value and writes it to a cookie. Then, on subsequent requests, both the cookie and post values are compared. If they don’t match, an error message is shown instead of usual data processing.

If you need to perform a post request but don’t want to build a form using CHtml, then you can pass a parameter with a name from Yii: :app()->request->csrfParam and a value from Yii: :$app->request- >getCsrfToken().

There’s more.

Lets have a look at some more features.

Disabling CSRF-tokens for all actions

1. If you have a problem with enableCsrfvalidation you can switch it off.
2. To disable CSRF, add this code to your controller:

public function beforeAction($action) { $this->enableCsrfValidation = false; return parent::beforeAction($action);

}

Disabling CSRF-tokens for a specific action

public function beforeAction($action) {

$this->enableCsrfValidation = ($action->id !== "actionId"); return parent::beforeAction($action);

}

CSRF validation for Ajax-calls

When the enableCsrfValidation option is enabled in the main layout, add csrfMetaTags: <head>

<?= Html::csrfMetaTags() ?>

</head>

Now you will be able to simply add it to ajax-call

var csrfToken = $('meta[name="csrf-token"]').attr("content");

$.ajax({

url: 'request'