

Zend_View Notice Suppression

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_View::supressNotices() Component Proposal

Proposed Component Name	Zend_View::supressNotices()
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_View::supressNotices()
Proposers	Ralph Schindler
Revision	1.1 - 1 August 2006: Updated from community comments. (wiki revision: 3)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

The general idea is to allow developers to code an application with `error_reporting(E_ALL | E_STRICT)`; while allowing Zend_View scripts to execute with the same error reporting MINUS `E_NOTICE`. The idea here is that during script execution, there is generally a lower level of strictness when it comes to using an object that may not have a pre-set property or using an array without a preset key.

2. References

- [Similar Trick in PHP Manual](#)

3. Component Requirements, Constraints, and Acceptance Criteria

This api addition would require a public method called `supressNotices($bool)` that accepts a bool on whether to include that functionality at render time.

4. Dependencies on Other Framework Components

- Zend_View

5. Theory of Operation

The component is instantiated with a mind-link that ...

6. Milestones / Tasks

Describe some intermediate state of this component in terms of design notes, additional material added to this page, and / code. Note any significant dependencies here, such as, "Milestone #3 can not be completed until feature Foo has been added to ZF component XYZ." Milestones will be required for acceptance of future proposals. They are not hard, and many times you will only need to think of the first three below.

- Milestone 1: [design notes will be published here](#)
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

If a milestone is already done, begin the description with "[DONE]", like this:

- Milestone #: [DONE] Unit tests ...

7. Class Index

no classes

8. Use Cases

Typically, a single form might be used for both the C and the U in the CRUD mantra. Imagine using Zend_Db_Table for this CRUD application, and that for updating you have passed the row object to the form.phtml script. The current only way to suppress notices is by doing the following.

```

<form action="/form-admin/save/" method="POST">

    Date <br />
    <?= $this->formText('date', @$this->rowthing->date); ?>
    <br /><br />

    Topic<br />
    <?= $this->formText('topic', @$this->rowthing->topic); ?>
    <br /><br />

    Abstract<br />
    <?= $this->formTextarea('abstract', @$this->rowthing->abstract); ?>
    <br /><br />

    <input type="submit" value="Save" />

</form>

```

The problem with the above is that if Zend_Db_Table_Row decides to throw an error, that will get suppressed as well as the @ does not discriminate.

The following (similar) code would be needed at render time

```

$_viewPreviousErrorLevel=error_reporting();
error_reporting($_viewPreviousErrorLevel& ~E_NOTICE);
// script execution here
error_reporting($_viewPreviousErrorLevel);

```

9. Class Skeletons

none

```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>

```