Zend_Controller_Front_Scaffold

<ac:macro ac:name="info"><ac:parameter ac:name="title">New Proposal Template</ac:parameter><ac:rich-text-body>
This page has been created from a template that uses "zones." To proceed:

Edit the page
Replace sample content within each zone-data tag
Remove this notice
Save the page
When you are ready for review, remove the <ac:emoticon ac:name="warning" /> Under Construction notice

</ac:rich-text-body></ac:macro>

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is under construction and is not ready for review. </ac:rich-text-body></ac:macro>

<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_Controller_Scaffold Component Proposal

Proposed Component Name	Zend_Controller_Scaffold
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Controller_Scaffold
Proposers	Jeremy Giberson
Revision	1.0 - 21 March 2007: Created proposal (wiki revision: 6)

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

Zend_Controller_Scaffold aims to provide Scaffolding similar to ruby on rails by extending the Zend_Controller classes.

2. References

- Google
- real references to be added

3. Component Requirements, Constraints, and Acceptance Criteria

- This component provides Scaffolding through Controllers, one table per controller with List, Add, Remove and Edit actions.
- This component **will** provide the features with out the need to create any files.
- This component **will** provide fallback support--you can code parts of your controller and rely on Scaffolding for incomplete actions.
- This component **will** use Zend_View for displaying output. View files will be for form fields (input) and controller files (layout).
- This component **will** support overloading or changing the view directory to use custom views.
- This component **should** optionally rely on Zend_Filter to provide form validation for data entry and updates.

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Controller
- Zend_Filter (maybe)
- Zend_View

5. Theory of Operation

Breif:
The componenent is utilized by instanciating a Zend_Controller_Front_Scaffold object. By default, Scaffold provides the exact same behaviour as Zend_Controller_Front. With a few meaningful method calls the developer can specify a controller name for the scaffold to provide scaffold features for. If no controller file exists scaffolding will dynamically instantiate Zend_Controller_Action_Scaffold to provide actions for the controller. If however an action controller exists it will be used instead, but for any undefined action the controller will fall over to Zend_Controller_Action_Scaffold.

Detailed:
The developer will be required to instantiate a Zend_Table object for each table/data they desire Scaffolding for. Zend_Controller_Front will be replaced by Zend_Controller_Front_Scaffold why by default exhibits the same behaviour. The developer will pass a controller name and zend_table object to the controller to enable scaffolding. Optionally the developer can enable scaffolding for many zend_table objects under different controller names and relate controls by a table and field thus providing a relation between scaffolding objects.
When a URI references a Scaffold'ed controller it will check the controller directory for an existing controller, should none exist Zend_Controller_Action_Scaffold will be used in its place. If however, a controller file is in place, it will be used for all defined Actions and only undefined actions will fall back to Zend_Controller_Action_Scaffold. This will allow the developer full scaffolding ability with no pre file generation and the ability to slowly replace scaffolding with actual code.

6. Milestones / Tasks

- Milestone 1: General consensus this is a prefered implementation of Scaffolding
- Milestone 2: design notes will be published here
- Milestone 3: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 4: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 5: Unit tests exist, work, and are checked into SVN.
- Milestone 6: Initial documentation exists.

7. Class Index

- Zend_Controller_Front_Scaffold
- Zend_Controller_Action_Scaffold
- Zend_Controller_Dispatcher_Scaffold

8. Use Cases

UC-01

Scaffolding a (very) simple blog. The blog table contains a date, a title, an entry and a mood. Mood is a foreign key to a mood table populated with ascii mood descriptors [:/ ;\ ].

The bootstrap file:

```
/**
     * Load the Initial Zend Framework Class
     */
    require_once 'Zend.php';
    /**
     * AutoLoad Classes where possible!
     */
    function __autoload($className) {
        // fall back to zend framework
        Zend::loadClass($className);
    }


    /**
     * Create a Controller Front (w Scaffolding)
     */
    $controller = Zend_Controller_Front_Scaffold::getInstance();
    // set path to controllers
    $controller->setControllerDirectory('controllers');
    // set what we want scaffolding with
    $controller->scaffold('Blog', new BlogTable() );
    $controller->scaffold('Mood', new MoodTable() );
    // here we relate two scaffolds by a unique field
    // this is in the format of $parent_table, $ptable_field, $child_table, $ctable_field
    $controller->scaffoldRelate('Blog', 'mood-id', 'Mood', 'id');


    // dispatch
    $controller->dispatch();
```

This short bit of code provides two scaffolded controllers:
mywebsite.com/Blog
mywebsite.com/Mood
Each of which provides the ability to view the rowsets in the related table, add, edit and delete rows into the table.
The following is psuedo html output resulting from this code.

mywebsite.com/Blog

> *Blog*
> ----------
> *03/16/07 - Lorem ipsum et sel adul..*
> *(view | edit | delete)*
> ----------
> *:/ 03/14/07 - Tores pe et sodi carum...*
> *(view | edit | delete)*
> ----------
> *:] 03/13/07 - Etchel postori et mi..*
> *(view | edit | delete)*

mywebsite.com/Blog/new

> *Blog (New Entry)*
> *----------*
> *Date: <date select>*
> *----------*
> *Mood: <list select> <a>view (view scaffolding for MoodTable)*
> *----------*
> *Title: <text input>*
> *Entry: <textarea>*
> *----------*
> *<submit> <cancel>*

mywebsite.com/Blog/edit

> *Blog (edit)*
> *----------*
> *Date: 03/16/07*
> *----------*
> *Mood: <* *> <a>view (view scaffolding for MoodTable)*
> *----------*
> *Title: Lorem ipsum et sel adulor pak*
> *Entry: sodu pecicl nots ss..*
> *----------*
> *<submit> <cancel>*

Notice in this crude example, that the display area for Mood has a link (points to mywebsite.com/Mood) to the table it is related to. This is directly resulting from the scaffoldingRelate() method.

9. Class Skeletons

```
class Zend_Controller_Front_Scaffold extends Zend_Controller_Front {


        /**
         * Specify the zend table object that should be used to provide a scaffolding for a given
ControllerName.
         *
         *
         * @param string $controller_name
         * @param Zend_Db_Table $zdb_table
         */
        public function scaffold( $controller_name, $zdb_table ) {

        }

        /**
         * Specify foreign key relations.
         * Controller names must already exist by having called scaffold($name, $table) previously
         * Field names must be valid fields in the tables passed along with the controller name in the
previous scaffold statement
         *
         * @param String $parent_controller_name
         * @param String $pc_field
         * @param String $child_controller_name
         * @param String $cc_field
         */
        public function scaffoldRelate( $parent_controller_name, $pc_field, $child_controller_name,
$cc_field ) {

        }

        /**
         * Change the directory where scaffold looks for view files
         *
```

```
     * @param string $directory
     */
    public function setScaffoldViewDirectory($directory) {

    }
}


class Zend_Controller_Action_Scaffold extends Zend_Controller_Action {
    private $_results_per_page = 15;


    public function indexAction() {
        // select top $_rpp entries in table by primary id
        // display $_rpp entries with next/prev nav
        // link to edit/delete entry from each listing
        // provide links to new entry
    }

    public function editAction() {

    }

    public function deleteAction() {

    }

    public function insertAction() {

    }

    public function noRouteAction() {
        $req = $this->getRequest();
        $cname = $req->getControllerName();

        $this->_redirect("/$cname");
    }

    public function validateField( $field, $value ) {
        // rely on Table_Schema and Zend_Filter to
        // provide validation verses typical table field types
        // ..
        return $valid_value;
    }
}

class Zend_Controller_Dispatcher_Scaffold extends Zend_Controller_Dispatcher_Standard {

    /**
     * Check if controller physically exists in controller directory.
```

```
            * IF it does use it, otherwise just use a Zend_Controller_Action_Scaffold
            */
        }
```

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