

Primitus Proposal - John Coggeshall

<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: Primitus Component Proposal

| | |
|--------------------------------|---|
| Proposed Component Name | Primitus |
| Developer Notes | http://framework.zend.com/wiki/display/ZFDEV/Primitus |
| Proposers | john at zend dot com |
| Revision | 1.1 - 1 August 2006: Updated from community comments. (wiki revision: 13) |

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

Primitus is an application framework built on top of Zend Framework to ease the rapid development of new applications built on top of ZF. It extends the foundation provided by ZF's MVC model to implement a number of ease-of-use features. As a framework designed specifically for new development it's goal is to provide structure to using the Framework in an intelligent fashion, and makes efforts to ensure applications developed with the Framework are done in the most intelligent way possible. It borrows some stylistic and feature abilities from other application frameworks without the overbearing natures many impose. Primitus is in fact an application itself, and new applications are created using Primitus by effectively "Copying" Primitus into a new application directory and running a configuration script to set application-specific variables.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

- Zend Framework .15
- Smarty 2.6.14
- PHP 5.1

4. Dependencies on Other Framework Components

- Zend
- Zend_Controller_Action
- Zend_Controller_Dispatcher
- Zend_Controller_Dispatcher_Token
- Zend_Filter_Input
- Zend_Controller_Front

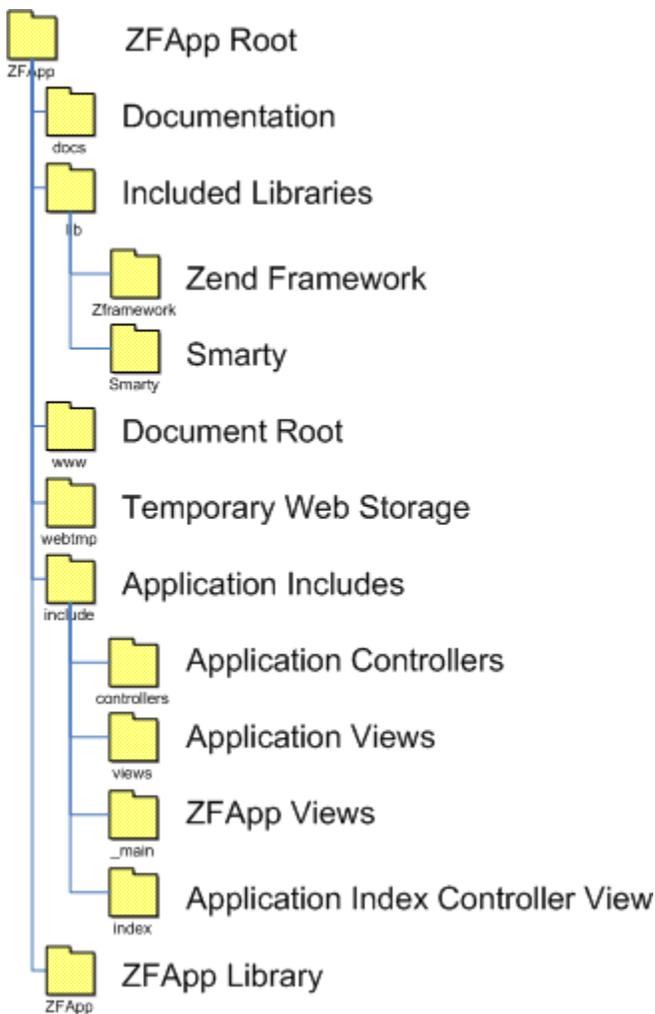
5. Theory of Operation

The Primitus Framework functions on two key principals:

- Completely self-contained (bundles necessary components)
- Standardized filesystem structure

These principals, combined with the power of Zend Framework, allow developers to quickly build powerful applications without spending significant time on the repetitious tasks involved in any application. Even many architectural requirements can be avoided, since Primitus is designed to lead the developer using sound architectural principals.

The Primitus directory structure is as follows:



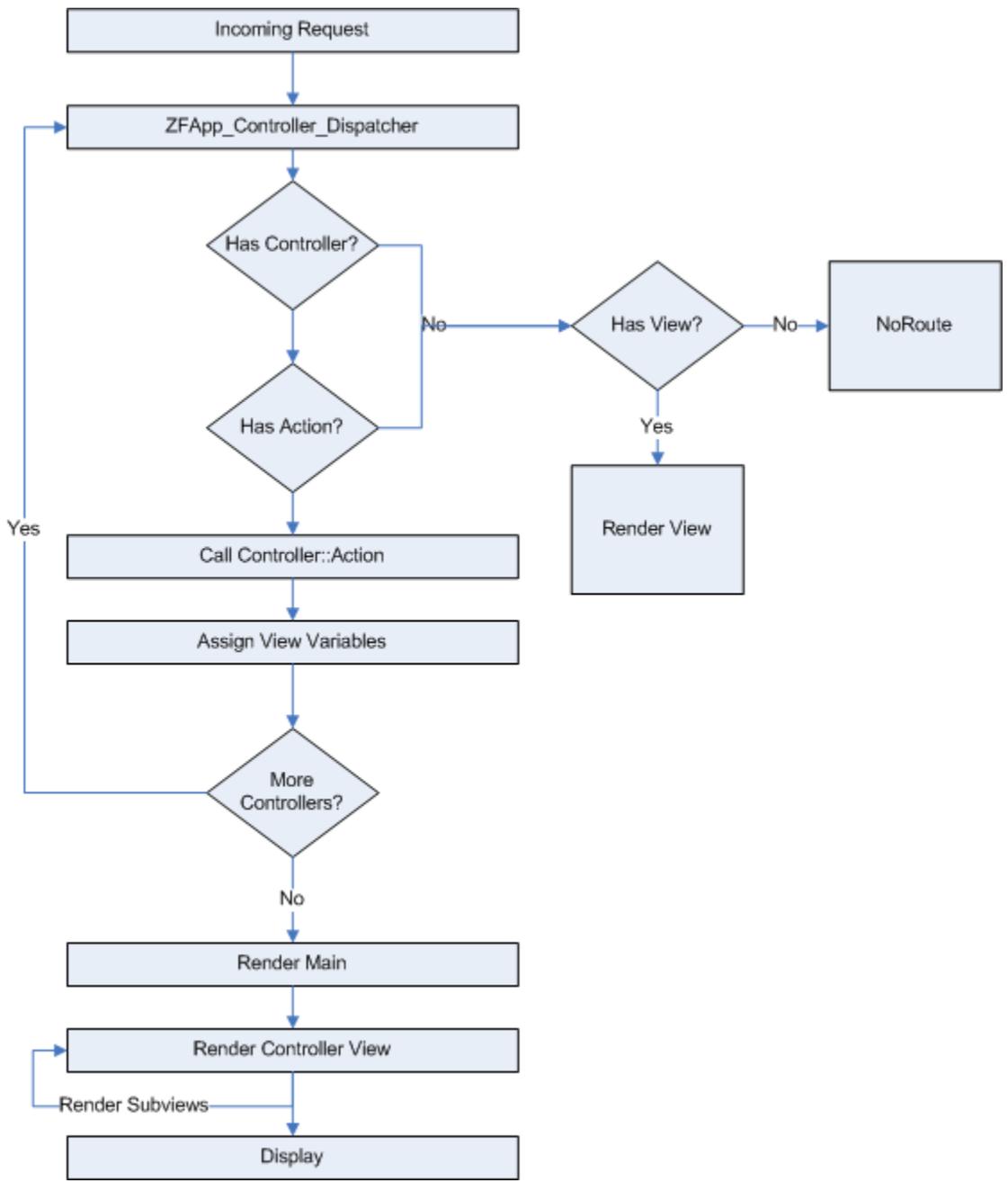
As it makes sense, all code which a normal developer should never have to touch has been placed into the Primitus/ library directory. This has both the benefit of a clear separation between user code and library code, but also makes it possible for applications build on top of Primitus to be

upgraded as new versions of Primitus are developed. Only a few special directories are required in the application-space:

- include/controllers/ApplicationController.php: A global controller, all public controllers should extend from this controller
- include/views/_main: Views in this directory are referenced directly from within the Primitus library and serve a special purpose such as error message handling, no-route conditions, and the main template
- include/views/index: Not really required by Primitus, it is only included so a new application has a nice "face" when a new application is generated.

Upon downloading Primitus, you will find that it is as much an application template as it is a framework. In fact, the only difference between Primitus and an application generated by Primitus (at least before development begins) is that constants relied on by the framework have been correctly set. These constants are determined at application generation by the appgen.php script included with the framework. These constants may eventually be removed all-together in lieu of a relative-referenced path system.

For reference, the dispatch flow which produces controller-less views is as follows:



Upon creating a new Primitus Application, you will notice a number of features provided by the framework:

- Controller-less views: Views are organized in the include/views directory as controllername/action.tpl. If the router determines it should execute the "Foo" controller's "Bar" action and no such controller exists, Primitus will also look to simply render foo/bar.tpl prior to throwing a no-route condition
- Beautified Error handling: Primitus catches all uncaught exceptions and uses the opportunity to display the user with a clean, customizable, error page describing the error, where it occurred, and the backtrace for faster development. In a production environment, this functionality can be replaced with a more user-friendly error
- Improved Control-chain introspection: Primitus keeps track of which controllers (or controller-less views) occurred, useful for tracing complex control chains
- Complete separation of Controller from View: Views are rendered piece-wise by using the "render" view function. This function comes with a default implementation which will simply display the controllername/action.tpl template, however can be completely overridden to display other types of data or custom views as needed
- Improved Filtering: Primitus re-implements Zend Framework's Input Filter to allow better access to the data, including allowing it to filter a single variable (instead of an array).
- Scoped View Variables: Unless explicitly specified, variables assigned in a controller action are scoped specifically to that action's view template and will not clash with other templates.
- Convenience in data access: Primitus controllers all implement a robust base controller which provides access to the controller's scoped view, the database, and filtered references to all user-input points (GET, POST, COOKIE, etc)

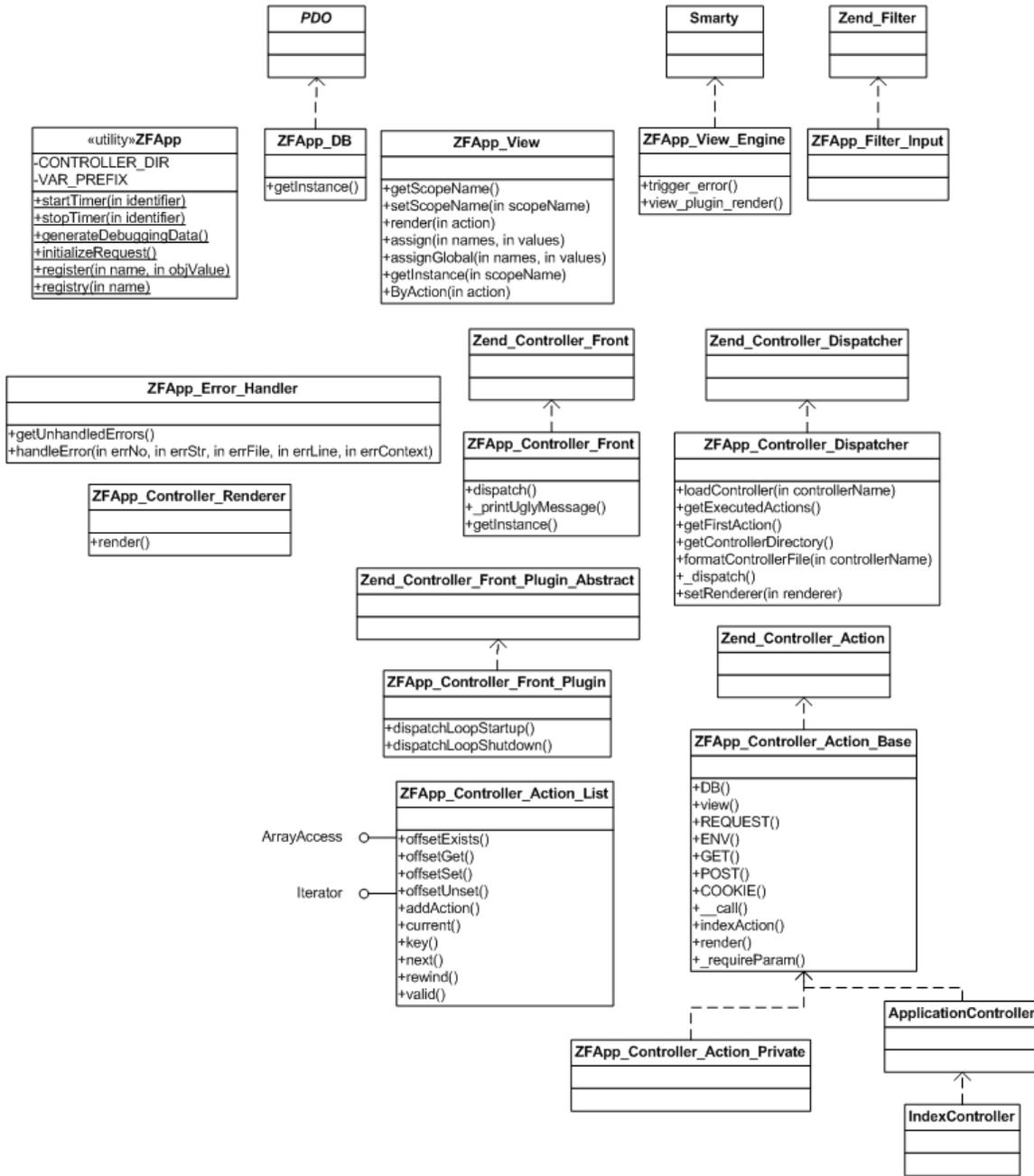
6. Milestones / Tasks

zone: Missing {zone-data:milestones}

7. Class Index

- Primitus_Controller_Front
- Primitus_Controller_Renderer
- Primitus_Controller_Dispatcher
- Primitus_Controller_Front_Plugin
- Primitus_Controller_Action_Base
- Primitus_Controller_Action_List
- Primitus_Controller_Action_Private
- Primitus
- Primitus_DB
- Primitus_View
- Primitus_View_Engine
- Primitus_View_Plugin_Render (function, not class)
- Primitus_Filter_Input
- Primitus_Error_Handler

And a relationship diagram...



8. Use Cases

9. Class Skeletons

See the Working Primitus preview release <http://private.coggeshall.org/ZFApp-preview-release-1.tar.gz>

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