Framework Comparison - CakePHP vs. Zend Framework (ZF) vs. CodeIgniter

Navigation

Databases and Configuration | Data Models | Contributions and Inclusions | Overall Differences / Focus | Links of Interest

> *Here is a basic research document comparing some of the basics of each framework. I am comparing:*
> *cakePHP.org | framework.zend.com | codeigniter.com*

Code and Approach Comparison:

Databases and Configuration

- **Zend Databases and Configuration**
  - http://framework.zend.com/docs/quickstart/create-a-model-and-database-table
  - as you can see its saved in an .ini file and is not all that human readable.
  - Or you can create a class that extends the Zend_Config class which is found here:
    - http://framework.zend.com/manual/en/zend.config.html
    - note that after array has been created it must be passed to the Zend_Factory (the zend core) with:
    - $config = new Zend_Config($configArray);
    - and then a static method can be called to execute it:
    - $db = Zend_Db::factory($config->database->adapter,
    - $config->database->params->toArray();
    - after data is saved and executed it can be read with:
    - $host = $config->database->get('host', 'localhost');
- **CakePHP Databases and Configuration**
  - http://book.cakephp.org/view/310/Configuration
  - Uses very easy to read array
  - Database information is automatically included into the core of the system and can be referenced
  - For creating your own configurations you should use the configuration class
    - http://book.cakephp.org/view/42/The-Configuration-Class
    - you can write to it with
    - Configure::write('Company.name','Pizza, Inc.');
    - and read from it with
    - Configure::read('Company');
- **CodeIgniter Databases and Configuration**
  - http://codeigniter.com/user_guide/database/configuration.html
  - As you can see it comes ready to handle multiple databases at the flic of a switch.
  - Database information is saved and is automatically loaded from that array into every model, component, helper, etc.etc, and can be referenced in the future with $this->db->methodName();
  - you can config load data with:
    - $this->config->load('item name');
  - You can create new configuration files by simply creating a new settingsName.php file and putting an array in there with your configuration files.
    - http://codeigniter.com/user_guide/libraries/config.html for more info.
  - New Configurations can be automatically loaded by adding its name to the "autoload" array saved in autoload.php
- **Summary of Configuration Approaches**
  - **Zend** uses a app.ini file that is parsed by the system. It doesn't use php syntax instead opts for a traditional desktop application style configuration and it doesn't show room for expansion.
  - **CakePHP** uses a array style database configuration but requires a class to be created for new configurations but loading/writing configuration is as easy as calling a single function.
  - **CodeIgniter** uses simple file based array configuration saving. Once any configuration file is loaded into the system it can be access with a simple function. All core configurations and new user configurations are treated the same and can be access the same in all contexts.

Data Models

- **Zend Data Models**
  - http://framework.zend.com/docs/quickstart/create-a-model-and-database-table
  - Key Code Snippet: "class Model_DbTable_GuestBook extends Zend_Db_Table_Abstract"
    - note that is this model needs to extend the Db_Table_Abstract_Class and is the Data Model for only the GuestBook DB.
  - Next Zend creates a Guestbook Model.
    - http://framework.zend.com/docs/quickstart/create-a-model-and-database-table
    - Key Code Snippet: "Model_GuestBook"
      - note that it extends nothing, which means a core Model will have to be written for our application that inherits some key functions that all our models will need.
    - You'll notice code like the following to include the Guestbook Database Model.
      - require_once APPLICATION_PATH . '/models/DbTable/GuestBook.php';
      - $this->_table = new Model_DbTable_Guestbook;
    - And the following to get the fields

        - $fields = $table->info(Zend_Db_Table_Abstract::COLS)
            - note, it doesn't even use $this, it calls a static method.
    - and to get the data:
        - $this->getTable()->fetchAll('1')->toArray();
        - $select = $table->select()->where('id = ?', $id);
            - note that is uses a question mark to substitute for $id when the query is run.

- **CakePHP Data Models**
    - http://book.cakephp.org/view/312/Models
    - Key Code Snippet: "class User extends AppModel"
        - note regardless of model purpose it extends AppModel to inherit all view, helper, loading functions etc.etc. and SO much more, This is so that the $this-> method can be used at all times and any functions can be aware of their context within the application. (This functionality is missing in Zend and would need to be written in)
    - Now for getting any data, you may use commands like these:
        - $this->findAll()
        - Returns the specified fields up to $limit records matching $conditions (if any), start listing from page $page (default is page 1). $conditions should look like they would in an SQL statement: $conditions = "race = 'wookie' AND thermal_detonators > 3", for example.
        - $this->find()
        - Returns the specified (or all if not specified) fields from the first record that matches $conditions.
    - "These magic functions can be used as a shortcut to search your tables for a row given a certain field, and a certain value. Just tack on the name of the field you wish to search, and CamelCase it (depending on your PHP version). Examples (as used in a Controller) might be:"
        - # $this->Post->findByTitle('My First Blog Post');
        - # $this->Specimen->findAllByKingdom('Animalia');
- **CodeIgniter Data Models**
    - http://codeigniter.com/user_guide/general/models.html
    - Key Code Snippet: "class Blog Extends Model"
        - Note the similarity to CakePHP
    - Getting Data is very similar to cakePHP but does not contain the "magic" that cakePHP does:
        - $query = $this->db->get('tablename');
        - $this->db->update('entries', $this, array('id' => $_POST'id'));
    - Getting Fields of a table
        - $fields = $this->db->list_fields($table);
    - CodeIgniter has various functions to get results as an array, as an object, etc.etc.

- **Summary of Model Approaches.**
    - **CakePHP and CodeIgniter** have NO reference to the database, and require no database setup for a specific Model. In both frameworks once the database is setup in the configuration it is not longer required for reference, and is assumed in the MANY database querying functions. This lack of reference to the database is setup because in both frameworks they extend a core Framework Model class (AppModel or Model). In CodeIngiters case you can create a file called MY_Model.php and simple place it in a system read file location and it will automatically extend your core Model (which will apply to ANY models that extend Model)
    - **Zend** you must create a database model which references the core Zend_Db_Table_Abstract class and then for your actual database model you must include using a raw "require_once APPLICATION_PATH . '/models/DbTable/GuestBook.php'" command, which then calls a mixture of static methods from the same ZEND class.

Contributions and Inclusions

- **Zend Contributions and Inclusion Approach**
    - http://framework.zend.com/wiki/display/ZFPROP/Home
        - Contributions are organized by their placement in a contribution lifecycle.
        - All contributions are full classes or libraries (All 200 or so)
        - The fastest growing classes are those of WebServices.
    - Since Zend's documentation is limited to a "quickstart" and class based tutorials, I can't find any documentation on the inclusion of a specific component. I had to do a google search and I found another article at snook.ca
        - To include a class (or component) you can use the commands:
        - ini_set('include_path',ini_get('include_path') . PATH_SEPARATOR .
        - vendor('Zend/Service/Delicious');
        - To use a component or class you create the object
        - $delicious = new Zend_Service_Delicious('username', 'password');
        - Note that the php.ini must be set.
- **CakePHP Contributions and Inclusion Approach**
    - CakePHP has the richest contribution home of all of the frameworks (and possibly every framework out there
        - http://bakery.cakephp.org/
    - This "Bakery" organizes its code into very clean categories which are separated into the main objects to be included into cakePHP giving a wide range of 3rd party addons:
        - http://bakery.cakephp.org/categories/view/3
    - As you can see you can download plugins, models, helpers, components, behaviors, or even browse snippets, which allow CakePHP to provide solutions at every level of your cakePHP application for almost anything.
    - For loading of any of the various imports of CakePHP a command such as the following can be used (files must only be placed in proper directories)
        - App::import('Model', 'MyModel');
        - App::import('Component', 'Auth');
        - etc.
    - Once imported a component can be used with a calls like the following:
        - $this->ModelName->method();
        - $this->ControllerName->method();
        - $this->HelperName->method();

        - Additionally, plugins can be imported and allow a function to be called anywhere within a CakePHP installation w/out $this->. With simple functionName() style calls.
- **CodeIngiter Contributions and Inclusion Approach**
    - CodeIgniter has a growing list of contributions organized by function and type of import:
        - http://codeigniter.com/wiki/Contributions
    - Given the simplicty of CodeIgniter an entire library could be one class (totally 25 lines) so, many of the contributions listed are are merely cleaned up forum posts. (and more often then not reference the post directly)
    - To use a Model,Library,Helper,Plugin, ETC, you must place the file into the appropriate folder.
    - Next you shoudl use a function such as:
        - $this->load->model('modelName');
        - $this->load->helper('HelperName');
        - $this->load->library('LibraryName');
    - You can also have any 3rd party script automatically load using the built-in autoload configuration (autoload.php) file, which contains an array for each style of import.
        - ex. $autoload'libraries' = array('database','session','template','request');
    - Once a 3rd party component has been $this->load'd, you can access it identically to cakePHP with commands like the followin:
        - $this->ModelName->method();
        - $this->Helper->method();
        - $this->Library->method();
- **Summary of Contributions and Inclusions**
    - **Zend** uses a vendor() function as a wrapper to include class files which then need to have an object created from to use. Out-of-the-box, Zend cannot autoload classes and the base classes for autoloading would need to be loaded.
    - **CakePHP** has one of the largest selection of contributions of all of the php frameworks out there. Its well organized and includes all 3rd part imports AS WELL AS code snippets. Importing code is simple and uses a static method which will include into the core. No autoloading could be found but the inheritance of all models/controllers/etc/ would make for propagation of a import fairly easily done.
    - **CodeIgniter** allows for easy integration of imports with a $this->load style command. It allows for autoloading of imports with a simple array style configuration. Easiest framework to propagate an import with.

Overall Differences / Focus

- **Zend** uses the static method call the most of the three frameworks and requires the most grassroots development to get an application (and its components) running. It has VERY powerful Components that are well developed by a strong community.
    - From a programming point of view Zend follows more traditional programming steps of "include and instantiate" for its components, and requires that basic Classes be written specific to your application so that you may intern extend them to to ensure class propagation.
- **CakePHP** uses a CORE $this-> style which roots almost all application functionality in a core Object that learns methods as you include components. Its contributions database/community is unrivaled and well organized. It provides a very strong base of data models with "magic" function naming to form complex queries very quickly.
    - From a programming point of view CakePHP runs a single Object style which inherits the various settings/aspects of the program. Little static method calls are used and has a lot of "magic" that makes assumptions and fills in gaps for your convenience.
    - CakePHP uses a multiple inheritance to provide many valuable and required methods to every object before you start writing any class, this is what brings the "RAPID" to CakePHP.
- **CodeIgniter** uses the same base structure as CakePHP except that ALL imports are done with the Core Object style. and $this-> object knows everything and can be used anywhere to access almost any setting of the application (and any imported components you wish). It has a growing list of contributions and is a "clean slate" in terms of MVC standards and how you want your application to run (There are no assumptions made by the system, and no magic functions)
    - From a programming point of view CodeIngiter is entirely object oriented and runs entirely off of a core CodeIgniter object (or "$this", in whatever context).
    - Like CakePHP it uses multiple inheritance to provide a strong base of functionality to rapidly build an application by providing a powerful stack to every model/controller/etc. This is the same experience that is acheived with CakePHP and for simple functionality developers feel more like they are gluing code rather then creating models.

Links of Interest

- **Comparison Documents**
    - http://snook.ca/archives/php/codeigniter_vs_cakephp/ - the first google result and a great comparison.
    - http://sevalapsha.wordpress.com/2007/11/13/zend-framework-vs-cakephp-symfony-seagull-wact-prado-trax-ez-and-codeigniter/ - a few interesting graph comparisons (Zend Framework vs. CakePHP)
- **CakePHP**
    - http://bakery.cakephp.org - community contributions website
    - http://cakeforge.org - a free service provided to Open Source developers offering easy access to the best in SVN, mailing lists, bug tracking, message boards/forums, task management, site hosting, permanent file archival, full backups, and total web-based administration.
    - http://en.wikipedia.org/wiki/CakePHP - Wikipedia.org
    - https://trac.cakephp.org/browser/trunk/cake/1.2.x.x/docs/COPYING.txt - MIT License that is in the latest version's svn trunk.
    - http://cakefoundation.org/pages/about - Cake Foundation site
    - irc://irc.freenode.net/CakePHP - #CakePHP on freenode.net
- **CodeIgniter**
    - http://codeigniter.com/wiki/Contributions - community contributions wiki page
    - http://codeigniter.com/user_guide/license.html - the license CodeIgniter uses
    - http://www.hopstudios.com/blog/is_codeigniter_actually_open_source/ - Line by Line License Disection
    - http://codeigniterdirectory.com/ - The first codeigniter directory for links, blogs, libraries, etc.
    - http://www.derekallard.com/ - Major CodeIgniter developer's blog (cool reading)
    - irc://irc.freenode.net/codeigniter - #codeigniter on freenode.net
- **Zend Framework**
    - http://framework.zend.com/wiki/display/ZFPROP/Home - Community Contributions Wiki
    - http://framework.zend.com/license - Zend Framework License
    - irc://irc.freenode.net/zftalk - #zftalk on freenode.net