Doctrine 1 and Zend_Tool Integration - Benjamin Eberlei

<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 Tool and Doctrine 1 Integration Component Proposal

Proposed Component Name	Zend Tool and Doctrine 1 Integration
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend Tool and Doctrine 1 Integration
Proposers	Benjamin Eberlei
Zend Liaison	TBD
Revision	1.0 - 9th November 2009 (wiki revision: 7)

Table of Contents

1. Overview

This Proposal aims to integrate Zend Framework and Doctrine 1 via Zend Tool. All Doctrine 1 CLI commands will be integrated into the Zend_Tool project context and additional context resources are provided to support Doctrine metadata, migrations, fixture etc. Both non-modular and modular applications should be supported for the task of generating models and tables. This proposal sets on top of Doctrine 1 resource proposal by Matthew Lurz.

2. References

- Zend Application Doctrine 1 Resource
- Doctrine Documentation
- Doctrine 1.2 PEAR Style Model Generation
- Zym Tool Doctrine 1 Code

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **MUST** use Doctrine 1 Resource for Zend_Application
    - This resource **SHOULD** be usable with both Modular and non-modular MVC application.
- This component **MUST** integrate all Doctrine 1 CLI Tasks
- This component **SHOULD** offer additional Zend Project Context Resources that are required for Doctrine.
- This component **SHOULD** offer further providers that help the integration of ZF and Doctrine
    - This component **SHOULD** make use of other Zend components for this goal.
    - This component **SHOULD** support prototyping of forms
    - This component **MAY** support Zend_Test and Doctrine integration.

4. Dependencies on Other Framework Components

- Zend_Application
- Zend_Tool_Framework
- Zend_Tool_Project
- Zend_Form
- Zend_CodeGenerator
- Doctrine 1.2

5. Theory of Operation

The Zend Tool integration will use the configured Zend Application resource
to configure Doctrine 1 for the use with Schema Tool or other CLI tasks.

With a configured Doctrine resource you will be able to issue the following commands:

```
zf build-all doctrine
zf build-all-load doctrine              # Generates Doctrine model, SQL, initializes
database, and load data
zf build-all-reload doctrine            # Generates Doctrine model, SQL, initializes
database, and load data
zf create doctrine                      # Generate all Database, Models, Forms
zf create doctrine.db                   # Create the db-schema
zf create doctrine.models               # Build Abstract Models
zf create doctrine.forms                # Build Abstract Forms
zf create doctrine.sql                  # Creates SQL for the current model
zf drop doctrine.db                     # Drop the database.
zf rebuild doctrine.db                  # Drop and Create database
zf migrate doctrine                     # Migrates database to current/specified
version
zf dump doctrine.data                   # Dumps data to the fixtures directory
zf load doctrine.data                   # Loads data from fixtures directory
zf run doctrine.dql "dqlstring"         # Run a DQL command and print all results,
much like mysql -e
zf insert-sql doctrine.sql             # Inserts SQL for current mode
zf generate doctrine.migration          # Generate migration class
zf generate doctrine.migrations-db      # Generate migration classes from existing
database connections
zf generate doctrine.migrations-models  # Generate migration classes from an existing
set of models
zf compile doctrine
zf convert doctrine.schema <from> <to>  # Can be YAML, Model, Database

# Convert a standard ZF Tool Project into a Doctrine Project
zf create doctrine.project <doctrineDirectory> <yamlSchemaDirectory=dcdir/yaml/>
<migrationsDirectory=dcdir/migrations> <sqlDirectory=dcdirectory/sql>

# Assign a YAML/Record Model-Name to a specific ZF Module
zf assign-model doctrine.module <moduleName> <modelName>...
# Configure a ZF Modules Doctrine Generation Tasks
zf configure doctrine.module <moduleName> zfstyle|pearstyle
<modelClassPrefix=moduleName_Models_> <formClassPrefix=moduleName_Forms_>
<baseClassPrefix=Base_>

zf clean doctrine.project                       # Clean-up Orphaned Files
```

Commands will all make heavy use of --verbose and --pretend flags.

You should be able to configure via the doctrine resource if you want to use code-generation
to generate abstract classes for forms or tables of Doctrine_Record's.

Usage would look like:

```
zf create project doctrineproject
zf create doctrine-project doctrineproject
cd doctrineproject/
vi application/configs/application.ini # configure doctrine resource
zf convert doctrine.schema database yaml # create yaml files into yamlSchemaDirectory.
zf build-all doctrine # create doctrine tables and records, aswell as forms
```

Zend Tool would keep track of the Doctrine context directories and will make sure that the generation is taking place as configured. If you want to generate different records/tables into different modules you have to configure the modules accordingly, this will take place in a new config file being called "doctrine-modules.ini" which has the Doctrine Model <-> ZF Module relationsships and can either be edited by hand or be configured with the help of "zf configure doctrine.module" and "zf assign-model doctrine.module".

```
zf create module cms
zf configure doctrine.module cms Cms_Models_ Base_
zf assign-model doctrine.module cms Category Article User
zf build-all-reload doctrine
```

It will now generate the classes in ZF-Style (Supported by Resource Loader):

```
|--application
|  |--modules
|  |  |--cms
|  |     |--models
|           |- Article.php
|           |- ArticleTable.php
|           |- Category.php
|           |- CategoryTable.php
|           |- User.php
|           |- UserTable.php
|           |- Base
|              |- Article.php
|              |- Category.php
|              |- User.php
|        |--forms
|           |- Article.php
|           |- Category.php
|           |- User.php
|           |- Base
|              |- Article.php
|              |- Category.php
|              |- User.php
```

Or in Pear Style:

```
|--library
|  |--Cms
|     |--Models
|        |- Article.php
|        |- ArticleTable.php
|        |- Category.php
|        |- CategoryTable.php
|        |- User.php
|        |- UserTable.php
|        |- Base
|           |- Article.php
|           |- Category.php
|           |- User.php
|     |--Forms
|        |- Article.php
|        |- Category.php
|        |- User.php
|        |- Base
|           |- Article.php
|           |- Category.php
|           |- User.php
```

With the help of Doctrine Model and Table context files it is possible to find orphaned files at this point and delete them.

All this is up for discussion and changes, its just an outline over what I think is possible and could be done to integrate Zend Tool and Doctrine 1. What do you think?

6. Milestones / Tasks

- Milestone 1: Community Review
- Milestone 2: Prototype
- Milestone 3: Zend Acceptance
- Milestone 4: Completion & Documentation

7. Class Index

- Zend_Doctrine_Tool_Provider_Doctrine
- Zend_Doctrine_Tool_DoctrineTasks
- Zend_Doctrine_Tool_Context_DataFixturesDirectory
- Zend_Doctrine_Tool_Context_MigrationsDirectory
- Zend_Doctrine_Tool_Context_SqlDirectory
- Zend_Doctrine_Tool_Context_ModelsDirectory
- Zend_Doctrine_Tool_Context_YamlSchemaDirectory
- Zend_Doctrine_Tool_Context_ModelFile
- Zend_Doctrine_Tool_Context_TableFile
- Zend_Doctrine_Tool_Context_ModelsAbstractDirectory
- Zend_Doctrine_Tool_Context_ModelAbstractFile
- Zend_Doctrine_Tool_Context_TableAbstractFile
- Zend_Doctrine_Tool_Context_DoctrineModuleConfigFile
- Zend_Doctrine_Tool_Context_FormDirectory
- Zend_Doctrine_Tool_Context_FormFile
- Zend_Doctrine_Tool_Context_FormAbstractFile
- Zend_Doctrine_Form

- Zend_Doctrine_CodeGenerator_Form
- Zend_Doctrine_Import
- Zend_Doctrine_Import_Builder

8. Use Cases

9. Class Skeletons

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

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