Zend_Cache refactoring - Marc Bennewitz

<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_Cache Refactoring Component Proposal

Proposed Component Name	Zend_Cache Refactoring
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Cache Refactoring
Proposers	Marc Bennewitz
Zend Liaison	TBD
Revision	0.1 - 6 Jun 2009: Initial Draft. 0.2 - 8 Jun 2009: Finished Proposal. 0.3 - 23 Jun 2009: Only one namespace option. 0.4 - 23 Jun 2009: Handling of multiple cache ids. 0.5 - 20 Jul 2009: Added how to instantiate without factory. (UC: Own Backend) 0.6 - 30 Jul 2009: Added namespace use case 0.7 - 18 Sep 2009: Added functions replace & add 0.8 - 08 Oct 2009: Added function optimize and moved automatic optimization from sqlite backend to core fontend 0.9 - 23 Oct 2009: Added Zend_Serializer proposal on references 0.10 - 23 Oct 2009: Added increment/decrement functionality 0.11 - 21 Jan 2010: Multiple Interface & getDelayed/fetch interface 6 backwards compatibility (wiki revision: 63)

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

Refactoring Zend_Cache for ease of use, more functionality and optimized performance.

2. References

- Zend_Cache
- Zend_Cache_Frontend_Page Etag support - Kevin Schroeder
- Zend_Serializer - Marc+Bennewitz used to select a specific serializer like php, igbinary etc. on backends which needs serialization to store different datatypes.

Source Code (In Development)
svn://marc-bennewitz.de/zf-proposals

3. Component Requirements, Constraints, and Acceptance Criteria

- The Zend_Cache factory **will** use a configurable plugin loader to load frontends and backends.
- The Zend_Cache factory **will** only accept one argument for options.
- The frontend **will** implement tagging if it isn't done by backend.
- The frontend **will no longer** serialize data. This **must be** done by backend.
- The Zend_Cache **can handle** an array of cache ids to get/set/remove multiple cache ids.
- The compontent **will** accept all php datatypes (incl. NULL) to cache them.
- Matching modes **should be** joined with the binary OR (|) operator for method clean and getIds.
- Only one namespace/prefix option **will be** used.
- The sqlite backend **will** support differents php extensions and versions: php_sqlite, sqlite, sqlite3.
- The backends **will** have just one interface to implement.
- The core frontend **will be** moved to the frontend folder -> Zend_Cache_Frontend_Core.
- The Test backend **will be** removed and the tests have to use the Array backend.

4. Dependencies on Other Framework Components

- Zend_Cache
- Zend_Exception
- Zend_Loader_PluginLoader
- Zend_Serializer

5. Theory of Operation

Serialization:

- The serialization has been moved to backend.
- Backends that only supports to handle strings there is a option named "serializer".
  This can be an instance of Zend_Serializer_Adapter_AdapterInterface or a string used to instantiate the serializer with Zend_Serializer::factory.
  Additionally you can disable serialization by set this option to NULL.
- Backends supporting different datatypes doesn't implement this option but memcached.
  Memcached has own serializer selectable with Memcache::SERIALIZER_* constants. To use the build-in serializer you can set the serializer option to one of the constants or you can handle this options same as other backends.

deprecated methods:

- Zend_Cache_Core::setLifetime - Please use the option "ttl" instead.
- Zend_Cache_Core::load - Please use get instead.
  - Old: load($id, $doNotTestCacheValidity = false, $doNotUnserialize = false)
  - New: get($id = null, array $opts = array())
- Zend_Cache_Core::test - Please use exist or info instead.
  - Old: test($id = null) : return mtime or false
  - New: exist($id = null, array $opts = array()) : return boolean
  - To get mtime use info($id, array $opts=array()) : return array with "mtime" key
- Zend_Cache_Core::save - Please use set/add/replace instead.
  - Old: save($data, $id = null, $tags = array(), $specificLifetime = false, $priority = 8)
  - New: set($data, $id = null, array $opts=array())
- Zend_Cache_Core::clean - Please use clear instead.
  - Old: clean($mode = Zend_Cache::CLEANING_MODE_ALL, array $tags = array())
  - New: clear($mode = Zend_Cache::MATCHING_ALL, array $opts = array())
- Zend_Cache_Core::getIdsMatchingTags - Please use find instead.
  - Use find(Zend_Cache::MATCHING_ACTIVE | Zend_Cache::MATCHING_TAGS, array('tags' => $tags))
- Zend_Cache_Core::getIdsNotMatchingTags - Please use find instead.
  - Use find(Zend_Cache::MATCHING_ACTIVE | Zend_Cache::MATCHING_NO_TAGS, array('tags' => $tags))
- Zend_Cache_Core::getIds - Please use find instead.

    - Use find(Zend_Cache::MATCHING_ACTIVE)
- Zend_Cache_Core::getFillingPercentage - Please use status instead.
    - Use status() - return array('total' => <number>, 'free' => <number>)
- Zend_Cache_Core::getTags - Please use info instead.
- Zend_Cache_Core::getMetadatas - Please use info instead.
    - Use info($id) - return array('mtime' => <int>, 'tags' => <array>, 'ttl' => <int>[, + backend spezific information])

Bc of all deprecated methods are only implemented on core frontend which is normally the base class.

deprecated core options:

- cache_id_prefix - Please use "namespace" instead.
- automatic_serialization - The option "automatic_serialization" has been removed. There is no more usage of it.
    - Now backends who doesn't support different datatypes implement new option "serializer" (see Zend_Serializer)
- automatic_cleaning_factor - Please use "automatic_clearing_factor" instead.
    - Because the method name has changed
- lifetime - Please use "ttl" instead.
    - Zend_Cache was the only caching system who named it lifetime. On all other systems it is named ttl

deprecated file backend options:

- read_control_type - Please use "read_control_algo" instead.
- hashed_directory_level - Please use "directory_level" instead.
- hashed_directory_umask - Please use "directory_perm" or "directory_umask" instead.
    - hashed_directory_umask wasn't a really umask it was a permission value
    - The new directory_perm set permissions by the same value
    - The new directory_umask set permissions by using umask
- file_name_prefix - Please use "namespace" option of core frontend instead.
- cache_file_umask - Please use "file_perm" or "file_umask" instead.
    - Same as hashed_directory_umask
- metadatas_array_max_size - Option completely removed

deprecated sqlite backend options:

- cache_db_complete_path - Please use "dsn" instead.
- automatic_vacuum_factor - Please use "automatic_optimizing_factor" of core frontend instead.
    - Normalized backend optimization interface

Changes on memcache(d) backend:

- Renamed Zend_Cache_Backend_Memcached to Zend_Cache_Backend_Memcache (uses memcache php extension)
- added Zend_Cache_Backend_Memcached (uses memcached php extension)
    - TODO: How can bc done with this naming change ???

deprecated class frontend options:

- cached_methods - Please use "cache_callbacks" instead.
- non_cached_methods - Please use "non_cache_callbacks" instead.
    - Now the class frontend extends the new callback frontend

deprecated function frontend options:

- cache_functions - Please use "cache_callbacks" instead.
- non_cache_functions - Please use "non_cache_callbacks" instead.
    - Now the class frontend extends the new callback frontend

6. Milestones / Tasks

- Milestone 1 [DONE]: Finish proposal

- Milestone 2 [DONE]: Working prototype
- Milestone 3: Prototype checked into the incubator
- Milestone 4: Unit tests exist finished and component is working
- Milestone 5: Initial documentation exists
- Milestone 6: Changed related components
- Milestone 7: Moved to core.

7. Class Index

- Zend_Cache
- Zend_Cache_Exception

- Zend_Cache_Frontend_Core **renamed**
- Zend_Cache_Frontend_Callback **new**
- Zend_Cache_Frontend_Function (extends Zend_Cache_Frontend_Callback)
- Zend_Cache_Frontend_Class (extends Zend_Cache_Frontend_Callback)
- Zend_Cache_Frontend_File
- Zend_Cache_Frontend_Output
- Zend_Cache_Frontend_Page

- Zend_Cache_Backend_Interface
- Zend_Cache_Backend_Abstract **renamed**
- Zend_Cache_Backend_Array **new**
- Zend_Cache_Backend_Apc
- Zend_Cache_Backend_File
- Zend_Cache_Backend_Memcache **renamed**
- Zend_Cache_Backend_Memcached **new**
- Zend_Cache_Backend_Xcache
- Zend_Cache_Backend_Sqlite
- Zend_Cache_Backend_SqliteAdapter_Abstract **new**
- Zend_Cache_Backend_SqliteAdapter_Sqlite **new**
- Zend_Cache_Backend_SqliteAdapter_Sqlite3 **new**
- Zend_Cache_Backend_SqliteAdapter_PdoSqlite **new**
- Zend_Cache_Backend_ZendPlatform
- Zend_Cache_Backend_ZendServer
- Zend_Cache_Backend_ZendServer_Disk
- Zend_Cache_Backend_ZendServer_ShMem
- Zend_Cache_Backend_TwoLevels

8. Use Cases

9. Class Skeletons

Please take a look to my development repository to see all classes.
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>