ТЕЛ. +7 (965) 145-08-95
E-MAIL: studio:[s]:t3club.com

Управление кэшированием в CMS TYPO3 на основе Caching framework

Начиная с 4-тых веток в TYPO3 появился встроенных механизм кэширования данных на основе "**Caching framework (далее CF)**". Основное назначение кэширования - уменьшение времени ожидания при повторном обращении к страницам сайта, а следовательно и снижение нагрузки на серверные ресурсы (за счет уменьшить числа обращений к базе данных).

Одна из ключевых задач, которую позволяет решать CF - производить сброс кэша одтельных элементов сайта (страниц, блоков) - без сброса всего кэша проекта, что является очень актуальным для посещяемых и больших ресурсов.

При генерации уникального идентификатора записей кэша страниц ядро системы учитывает различные параметры:

Параметр	Описание
[_GET] id	Основной параметр, идентифицирующий страницу, которая существует в Базе данных. Если страница не найдена - выводится ошибка.
[_GET] type	Подтип страницы, который определяется посредством Typoscript (обычно служит для определения фреймов и служебных страниц).
[_GET] cHash	Параметр определяющий виртуальные страницы, создаваемые из плагинов посредоством добавления к Typolink-ссылкам параметра useCachHash=1.
[Login usergroups]	Группы пользователей (при авторизации и работе FE-пользователей на сайте).
[TypoScript conditions]	Дополнительные условия определенные в шаблонах TypoScript.

Управление кэшированым производится через объект:**$GLOBALS['typo3CacheManager']**. Пример инициализации объекта и выполнения сбраса кэша:

php-код

```php

    // Пример удаления всех кэшей - очистки таблиц, которые зарегистрированы в фреймворк
$GLOBALS['typo3CacheManager']->flushCaches();

    // Пример удаления всего кэша для отдельно взятой таблицы
$GLOBALS['typo3CacheManager']->getCache('cache_pagesection')->flush();

//---------------------------------------

    // Для работы с данными кэшей через менеджер кэширования подключаемся к нужным табли
$pageCache = $GLOBALS['typo3CacheManager']->getCache('cache_pages');
$pageSectionCache = $GLOBALS['typo3CacheManager']->getCache('cache_pagesection');

    // Пример получения кэша отдельно взятой страницы по ID (по тэгу)
$temp_pageCache = $pageCache->getByTag('pageId_97'); // возвращает массив
$temp_pageSectionCache = $pageSectionCache->getByTag('pageId_97'); // возвращает массив

    // Пример удаления кэша отдельно взятой страницы по ID (по тэгу)
$pageCache->flushByTag('pageId_97');
$pageSectionCache->flushByTag('pageId_97');
```

Назначение стандартных таблиц для кэша в Базе данных:

Название таблицы	Назначение
cf_cache_hash	Для записи данных кэша объектов на странице (элементы меню и другие объекты TypoScript).
cf_cache_pages	Для записи кэша страницы с набором разлинчных условий условий (id, type, MP, cHash).
cf_cache_pagesection	-
cf_extbase_object	Таблица, которую можно использовать для записи кэша в плагинах и расширения без создания новых таблиц.

Пример записи кэша в таблицу cf_extbase_object

php-код

```php

    // Введем произвльный индефикатор
$cacheIdentifier = 12345;
$myCacheInstance = $GLOBALS['typo3CacheManager']->getCache('extbase_object');
$cachedContent = $myCacheInstance->get($cacheIdentifier);

    // Проверяем, если контент, уже закэширован
if ($cachedContent) {
    print $cachedContent;
} else {

        // Если контент еще не закэширован создаем
    $content = "Привет!" . rand(123,323232);
    $myCacheInstance->set(
        $cacheIdentifier,
        $content,
        array(), // тэги кэша (оставляем пустыми)
        5 // время жизни кэша в секундах - lifetime
    );
    print $content;

}
```

Структура для создания новой таблиц выглядит следующим образом:

php-код

```
#
# TABLE STRUCTURE FOR TABLE 'tx_myext_mycache'
#
CREATE TABLE tx_myext_mycache (
    id INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
    identifier VARCHAR(250) DEFAULT '' NOT NULL,
    crdate INT(11) UNSIGNED DEFAULT '0' NOT NULL,
    content mediumblob,
    lifetime INT(11) UNSIGNED DEFAULT '0' NOT NULL,
    PRIMARY KEY (id),
    KEY cache_id (identifier)
) ENGINE=InnoDB;

#
# TABLE STRUCTURE FOR TABLE 'tx_myext_mycache_tags'
#
CREATE TABLE tx_myext_mycache_tags (
    id INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
    identifier VARCHAR(250) DEFAULT '' NOT NULL,
    tag VARCHAR(250) DEFAULT '' NOT NULL,
    PRIMARY KEY (id),
    KEY cache_id (identifier),
    KEY cache_tag (tag)
) ENGINE=InnoDB;
```

Краткий список API-функций по управлению кэшированием:

php-код

```

$cache->set($id, $content, $tags = array(), $lifetime = null);
$cache->has($id); // проверить наличие записи с данным идентификатором
$cache->get($id); // получить кэш по идентификатору
$cache->remove($id); // удалить кэш по идентификатору
$cache->flush(); // очистить все

$identifiers = $cache->findIdentifiersByTag($tag);
$identifiers = $cache->findIdentifiersByTags(array $tags);

$cache->flushByTag($tag); // очистить по тэгу
$cache->flushByTags(array $tags); // очистить по тэгам
$cache->collectGarbage(); // как понял, это запуск сборщика мусора <img src="http://ivan

$cache->getByTag($tag); // получить кэш по тэгу
$cache->getIdentifier(); // возвращает текущий идентификатор
$cache->isValidEntryIdentifier($identifier) - проверяет существование идентификатора
$cache->isValidTag($tag) - проверяет существование тэга
```

В завершении стоит отметить, что в качесвте хранилища записей кэша можо использовать альтернативу базам данных (файлы, APC, Memcached) - путем переопределения соответствующих настроек Caching framework.

Добавление тэгов для виртуальных страниц

php-код

```
$GLOBALS['TSFE']->addCacheTags(array('tx_example_model:' . $id));
    $GLOBALS['typo3CacheManager']->getCache('cache_pages')->flushByTag('tx_example_model
```