Руководство по программированию для TYPO3 v4

Ключ расширения: doc_core_cgl

Язык: ru

Версия: 4.5.1

Ключевые слова: forDevelopers, forIntermediates

Официальная документация

Этот документ является частью официальной документации TYPO3. Одобрение было получено после экспертной оценки командой по документации TYPO3. Читатель вправе ожидать точной информации в этом документе, о любом несоответствии сообщайте в команду по документации (documentation@typo3.org). Официальные документы сохраняются по возможности актуальными командой по документации.

Перевод официальной документации

Этот документ является переводом официальной документации TYPO3. Перевод может быть не столь актуальным, как оригинал. В случае вопросов, обращайтесь к английской версии.

Руководство по ядру системы

Этот документ является руководством по ядру системы. Такие документы описывают встроенную функциональность TYPO3 и служат для обеспечения читателя подробной информацией. Каждый из документов, посвященных ядру системы, адресован конкретному процессу или функции, и их реализации в исходном коде ядра TYPO3. Это может быть информация по доступным API, определенным параметрам настроек и т.п.

Руководства по ядру системы написаны в виде справочников. Читатель должен сверяться с содержанием для определения конкретного, нужного ему раздела, лучшим образом подходящего под решаемую задачу.

Содержание

Введение

Об этом документе

В этом документе определены руководства по программированию проектов для TYPO3 версии 4. Следование этим советам обязательно для разработчиков ядра TYPO3 и разработчиков, помогающим развивать ядро системы.

Авторам расширений настоятельно рекомендуется следовать этим руководствам при разработке расширений для TYPO3. Эти правила призваны упрощению чтения кода, его анализу и пересмотру. Также они помогают избежать типичных ошибок при программировании для TYPO3.

Здесь описано, как должны быть структурированы и форматированы файлы, код и директории для TYPO3. Здесь нет обучения технике программирования для TYPO3 и не приводиться никакой технической информации.

Нумерация страниц соответствует печатной версии (доступной на веб сайте typo3.org).

Нововведения

Последняя версия CGL (руководства по кодированию) содержит самую полную и точную информацию по уже существующим руководствам. Здесь же отражены изменения, применительно к кодированию для TYPO3 4.5, вроде измененных деклараций XCLASS и использования кодировки UTF-8 в исходном коде. Указаны и самые последние изменения, вроде перехода к использованию Git.

Кроме того, приложение описывающее правильную настройку для различных IDE при работе с ядром TYPO3 v4 перекочевало на страницу wiki: http://wiki.typo3.org/Ru:PHP_Editors_/_IDE_for_TYPO3.

Составители

Изначально, документ руководства по программированию для TYPO3 был написан Kasper Skårhøj. Текущая версия была полностью переписана при содействии Ingo Renner и Dmitry Dulepov в 2008 году. На данный момент он курируется François Suter. Все изменения проходят процесс утверждения в команде разработчиков ядра TYPO3.

Обратная связь

По общим вопросам о документации, пишите на documentation@typo3.org.

Если найдена ошибка в этом руководстве, опишите проблему по данному руководству в системе отслеживания ошибок: http://forge.typo3.org/projects/typo3v4-doc_core_cgl/issues

Поддержка качественной документации является тяжелой работой, и команде по документации всегда нужны добровольцы. Если Вы желаете помочь, присоединяйтесь к списку рассылок (typo3.projects.documentation на lists.typo3.org).

Используемые в документе соглашения

Моноширинный шрифт используется для:

– названий файлов и директорий. Директории в конце своего названия имеют слеш (/);

– примеров кода;

– названий модулей TYPO3;

– ключей расширений;

– пространства имен TYPO3.

Внешний и внутренний интерфейсы TYPO3 сокращены по первым буквам в верхнем регистре, потому что они рассматриваются как подсистемы.

Соглашения относительно файловой системы

Имеются некие соглашения, относительно наименования файлов и директорий для ядра и расширений TYPO3. Некоторые из них сложились исторически и не зависят от других формальных правил. Они будут описаны отдельно. Новые классы ядра и расширений должны следовать следующим правилам, описанным здесь.

Структура директорий TYPO3

По умолчанию, в установленной TYPO3 имеются следующие директории:

Директория	Описание
fileadmin/	Это директория (с поддиректориями), в которой пользователи могут хранить файлы. Обычно — это изображения или файлы HTML. Зачастую, эта директория используется для загружаемых файлов. Только эта директория может быть доступна через модуль TYPO3 Файл.
t3lib/	Директория библиотек TYPO3.
typo3/	Директория внутреннего интерфейса TYPO3. В ней содержаться файлы, относящиеся ко внутреннему интерфейсу. Кроме того, здесь во вложенной папке `ext/` (не используются ядром TYPO3) и `sysext/`находятся некоторые расширения. Например, расширение "`cms`" содержит код, для формирования внешнего веб сайта.
typo3conf/	Директория настроек TYPO3.
typo3conf/ext/	Директория для расширений TYPO3.
typo3temp/	Директория для временных файлов. В ней имеются поддиректории для временных файлов расширений и компонентов TYPO3.
uploads/	По умолчанию — директория для загрузки. Например, все изображения, загружаемые для элемента "изображение с текстом", находятся в этой директории. Расширения могут использовать настройки `uploadfolder` из `ext_emconf.php` для создания собственной директории для загрузки внутри данной директории.

Это — структура, используемая по умолчанию при установке TYPO3. Другие приложение, не относящиеся к TYPO3, могут добавлять свои директории.

Файлы TYPO3 и файлы пользователей

Все файлы в директории веб сайта TYPO3 имеют свою иерархию и подразделяются на файлы TYPO3 и файлы пользователей. Файлы TYPO3 — это файлы, находящиеся в пакете исходных кодов TYPO3, выпущенных командой разработки ядра TYPO3. Они включают файлы внутри директорий `t3lib/` и `typo3/`, а также файл с названием `index.php` в корне установленной TYPO3.

Все остальные файлы относятся к файлам пользователей. Они включают расширения, файлы в директории `fileadmin/` или файлы, формируемые TYPO3 (вроде эскизов или временных файлов CSS).

Структура директории расширений

В директории расширений имеются следующие файлы и директории:

Название	Описание
ext_emconf.php	Это — обязательный для расширения файл. Он описывает расширения для TYPO3.
ext_icon.gif	Значок расширения. Название не должно меняться.
ext_localconf.php	В этом файле содержатся описания ловушек (hook) и настройка дополнений. Название не должно меняться.
ext_tables.php	Этот файл содержит таблицу деклараций. За дополнительной информацией обратитесь к документу “TYPO3 Core API”. Название не должно меняться.

ext_tables.sql	Эти файлы содержат определения для таблиц расширения. Название не должно меняться.
	В файле может быть либо полное определение таблицы, либо частичное. Полные описания таблиц объявляют таблицы расширения. Оно выглядит как обычный запрос SQL `CREATE TABLE`.
	В частичном определении таблиц содержится список полей, добавляемых к существующей таблице. Вот пример:
	`CREATE TABLE pages (`
	`tx_myext_field int(11) DEFAULT '0' NOT NULL,`
	`);`
	Обратите внимание на запятую после поля. В описании полной таблицы, это приведет к ошибке, а в частичном описании — она требуется. TYPO3 объединит описание этой таблицы с существующей при выполнении сравнения и определения фактической таблицы. В частичном определении, также могут содержатся индексы и другие указания. Они могут изменять существующие поля таблицы, это не рекомендуется делать, так как могут возникнуть проблемы с ядром TYPO3 и/или другими расширениями.
	TYPO3 анализирует файл `ext_tables.sql`. TYPO3 ожидает, что все описания таблиц в этом файле похожи на формируемые утилитой `mysqldump`. Неверные определения могут быть не распознаны анализатором SQL TYPO3.

tca.php	В этом файле содержится полное определение таблиц для расширения.
locallang*.xml	В этом файле находятся метки для локализации. Они также могут присутствовать в поддиректориях.
doc/	В этой директории находится руководство по расширению. Название не должно меняться.
doc/manual.sxw	Это — файл руководства по расширению в формате OpenOffice 1.0. Название файле и формат не должны меняться. За дополнительной информацией смотрите документ “Documentation template” на `typo3.org`.
piX/	Здесь находятся дополнения для внешнего интерфейса. Если расширение формируется через Kickstarter, *X* будет номером. Рекомендуется давать понятные названия для директорий дополнений внешнего интерфейса.
svX/	Здесь находятся директории сервисов TYPO3. Если расширение формируется через Kickstarter, *X* будет номером. Рекомендуется давать понятные названия для директорий сервисов.

Название	Описание
modX/	Эти директории в основном содержат модули внутреннего интерфейса. Если расширение формируется через Kickstarter, X будет номером. Рекомендуется давать понятные названия для директорий модулей внутреннего интерфейса.
modfuncX/	Эти директории в основном содержат подмодули внутреннего интерфейса (встроенные в другие модули). Если расширение формируется через Kickstarter, X будет номером. Рекомендуется давать понятные названия для директорий модулей внутреннего интерфейса.
lib/	Директория для файлов не TYPO3, распространяемых вместе с расширением. TYPO3 распространяется под GPL версии 2 или любой более поздней. Любой код, не предназначенный для TYPO3 должен подходить под GPL версии 2 или любой более поздней. Замечание: это не обязательное, но рекомендуемое название.

Такой структуры директорий **крайне рекомендуется** придерживаться. Расширения могут создавать необходимые директории (например, все языковые файлы могут находится в других директориях).

Названия файлов

TYPO3 требует, чтобы все названия файлов классов PHP начинались с приставки `class.`, затем приставка, определяемая пространством имен, символ подчеркивания, название класса, символ подчеркивания и расширение. За информацией по пространству имен и префиксам пространства имен обращайтесь к следующему разделу данного документа. Расширение для файлов PHP всегда должно быть `php`.

Файлы, не содержащие класса не должны начинаться с приставки `class.` Рекомендуется использовать лишь классы PHP и избегать файлов без классов.

Классы, содержащие интерфейсы PHP должны иметь приставку `interface.`

В одном файле должен быть лишь один класс или интерфейс.

Все названия файлов должны быть в нижнем регистре.

Файлы модульных тестов

Файлы модульных тестов находятся в папке "tests" в корне папки расширения TYPO3, и содержать внутреннюю структуру папок, соответствующую исходным кодам папки расширения.

Соглашения об именовании отличаются от описанного выше:

1. Названия не должны иметь приставку "class."
2. В конце названия, прямо перед расширением ".php", должно прибавляться "Test".

Пример

Файл модульного теста для `t3lib/db/class.t3lib_db_preparedstatement.php` будет следующим:

```
tests/t3lib/db/t3lib_db_preparedstatementTest.php
```

Информация по модульным тестам дана в разделе "Модульные тесты".

Пространства имен

Логически TYPO3 разбивает все файлы и директории по нескольким пространствам имен. Это преследует двоякую цель:

1. Показывает, к чему внутри TYPO3 CMS относятся файлы и директории.
2. Ограничивает выполнение PHP файлами из определенного пространства имен (например, TYPO3 полагает, что все вызываемые функции находятся в пространствах `user_` или `tx_`).

Следующий раздел описывает пространства имен применяющееся в TYPO3 на данный момент.

t3lib

Пространство `t3lib` зарезервирована для файлов TYPO3 общего назначения. Такие файлы используются и во внешнем, и во внутреннем интерфейсах. Физически это пространство относится к директории `t3lib/` в иерархии директорий TYPO3.

Все файлы классов PHP в пространстве имен `t3lib` начинаются с приставки `class.t3lib_`.

Директория `t3lib/` содержит поддиректории. Файлы классов и интерфейсов в этих поддиректориях имеют названия с добавлением названия директории к приставке `t3lib_` и отделяются от названия класса символом подчеркивания. Например, файлы в директории `t3lib/cache/` имеют названия `class.t3lib_cache_exception.php`. Файлы внутри `t3lib/cache/backend` имеют названия `class.t3lib_cache_backend_abstractbackend.php`.

Пользовательские файлы не разрешены внутри данного пространства.

typo3

Это пространство зарезервировано для файлов внутреннего интерфейса TYPO3. Пользовательские файлы здесь недопустимы.

Исторически сложилось, что файлы в этом пространстве имеют разные приставки и не следуют общим правилам именования.

tslib

`tslib` исторически служат для “библиотеки TypoScript”. Это пространство — часть расширения `cms`. Физически файлы расположены в директории `typo3/sysext/cms/tslib/` и содержат страницу внешнего интерфейса и файлы для формирования содержимого.

Исторически сложилось, что файлы в этом пространстве могут не следовать общим правилам именования..

Пользовательские файлы не разрешены внутри данного пространства.

tx_

Это пространство зарезервировано для расширений. Файлы классов PHP расширений должны начинаться с приставки `class.tx_` затем идет ключ расширения без подчеркиваний и название класса в нижнем регистре. Файлам дается расширение `php`. Например, если ключ расширения – `test_ext`, название файла должно быть

class.tx_testext_myclass.php,а название класса – tx_testext_myClass.

Пользовательские файлы из этого пространства, в основном находятся в директории typo3conf/ext/. Эти файлы можно установить в директорию typo3/ext/ общую для нескольких установок TYPO3.

user_

Это пространство используется для файлов PHP без классов PHP или для расширений, относящихся к одной установке. Не рекомендуется создавать расширения с приставкой user_.

Файлы без классов содержат функции PHP. Эти функции могут вызываться через TYPO3, только если они имеют приставку user_ (приставку может изменить администратор в Install tool). Все функции внутри таких файлов также должны иметь приставку user_. Обычно, такие файлы находятся внутри директории fileadmin/ или ее поддиректориях.

ux_

Это пространство зарезервировано для файлов XCLASS. Такие файлы обычно появляются в расширениях.

Форматирование файла PHP

Общие требования к файлам PHP

Теги PHP

Каждый файл PHP в TYPO3 должен использовать полные теги PHP. Должна быть ровно одна пара — открывающий и закрывающий теги (никаких открывающих и закрывающих тегов в середине файла). Пример:

```
<?php
      // File content goes here
?>
```

Не должно быть никаких пустых строк после закрывающего PHP тега. Пустые строки после закрывающих тегов прерывают сжатие на выходе в PHP и/или приводят к ошибкам в AJAX.

Переносы строк

TYPO3 использует переносы строк в стиле Unix (`\n`, PHP `chr(10)`). Если разработчик использует платформы Windows или Mac OS X, то необходимо настроить редактор на использование переносов строк в стиле Unix.

Длина строки

Необходимо избегать слишком длинных строк в коде в угоду удобочитаемости. Длина строки ограничена 130 символами (**включая** табуляции). Длинные строки необходимо по возможности разбивать. Каждый фрагмент строки, начиная со второго, должен быть обозначен одной или несколькими табуляциями. Пример:

```
$rows = $GLOBALS['TYPO3_DB']->exec_SELECTgetRows('uid, title', 'pages',
      'pid=' . $this->fullQuoteStr($this->pid, 'pages') . $this->cObj-
>enableFields('pages'),
      '', 'title');
```

или даже лучше для удобства чтения:

```
$rows = $GLOBALS['TYPO3_DB']->exec_SELECTgetRows('uid, title',
      'pages',
      'pid=' . $this->fullQuoteStr($this->pid, 'pages') . $this->cObj-
>enableFields('pages'),
      '',
      'title'
);
```

Строки комментариев должны быть ограничены примерно 80 символами (**исключая** табуляции), так как это упрощает их чтение.

Замечания

– Табуляция соответствует длине 4 пробелов.

– При разбиении строк, старайтесь их разбивать в наиболее малозначащей точке. В приведенном выше примере, строки разбиты между двумя аргументами, а не в середине одного из них. При длинном логическом выражении, ставьте логический оператор в начале следующей строки, например:

```
if ($GLOBALS['TYPO3_CONF_VARS']['SYS']['curlUse'] == '1'
      && preg_match('/^(?:http|ftp)s?|s(?:ftp|cp):/', $url)) {
```

Пробелы и отступы

TYPO3 использует табуляции для отступов в коде. Один уровень — одна табуляция.

В конце строки не должно быть пробелов. Их можно убрать вручную или переложить на текстовый редактор (обратитесь к разделу "Настройки редактора").

Пробелы должны быть добавлены:

– с обоих концов строки, операторов арифметических, присвоения и подобных им (например, `.`, `=`, `+`, `-`, `?`, `:`, `*`, и т. п.);

– после запятых;

– в строковом комментарии, после знака комментария (двойной слеш);

– после звездочек в многострочном комментарии.

Кодировка

Все исходные файлы TYPO3 используют кодировку UTF-8 начиная с версии TYPO3 4.5. Файлы сторонних библиотек могут иметь отличные кодировки.

Структура файлов

Файлы TYPO3 используют следующую структуру:

1. Открывающий тег PHP.
2. Замечание об авторских правах.
3. Блок информации о файле (с возможной функцией index) в формате phpDoc.
4. Включаемые файлы.
5. Блок информации о классе в формате phpDoc.
6. Класс PHP.
7. Декларацию XCLASS.
8. Возможный код выполняемого модуля (например, в классах eID).
9. Закрывающий тег PHP.

В следующем разделе разъясняется каждая из частей.

Замечание об авторских правах

TYPO3 выпускается под GNU General Public License версии 2 или более поздней. Замечание об авторских правах со ссылкой на GPL должно быть включено в начало каждого файла класса PHP TYPO3 PHP. Файлы `user_` также должны иметь данное замечание. Пример:

```
<?php
/***************************************************************
*  Авторские права notice
*
*  (c) YYYY Your name here (your@email.here)
*  All rights reserved
*
*  This script is part of the TYPO3 project. The TYPO3 project is
*  free software; you can redistribute it and/or modify
*  it under the terms of the GNU General Public License as published by
*  the Free Software Foundation; either version 2 of the License, or
*  (at your option) any later version.
```

```
*
*  The GNU General Public License can be found at
*  http://www.gnu.org/copyleft/gpl.html.
*  A copy is found in the textfile GPL.txt and important notices to the license
*  from the author is found in LICENSE.txt distributed with these scripts.
*
*
*  This script is distributed in the hope that it will be useful,
*  but WITHOUT ANY WARRANTY; without even the implied warranty of
*  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*  GNU General Public License for more details.
*
*  This copyright notice MUST APPEAR in all copies of the script!
***************************************************************/
```

Это замечание не должно изменяться, за исключение года, имени автора и e-mail автора.

Блок с информацией о файле

Блок информации о файле идет следом за авторскими правами и дает общее представление о файле. Сюда должно быть включено название файла, его описание и информацию об авторе (или авторах). Пример:

```
/**
 * class.tx_myext_pi1.php
 *
 * Provides XYZ plugin implementation.
 *
 * @author John Doe <john.doe@example.com>
 */
```

В блок информации о файле может входить функция index. Этот индекс создается и обновляется расширением `extdeveval`.

Включаемые файлы

Файлы включаются посредством функции `require_once`. Все файлы TYPO3 должны использовать абсолютные пути при вызове `require_once`. Существует два способа получения пути включаемого файла:

1. Использовать одну из предопределенных констант TYPO3: `PATH_tslib, PATH_t3lib, PATH_typo3, PATH_site`. Первые три содержат абсолютные пути к соответствующим директориям TYPO3. Последняя содержит абсолютный путь к корневой директории TYPO3. Пример:

   ```
   require_once(PATH_tslib . 'class.tslib_pibase.php');
   ```

2. Использовать функцию `t3lib_extMgm::extPath()`. Эта функция принимает два аргумента: ключ расширения и путь к включаемому файлу. Второй аргумент необязателен, но рекомендуется к использованию. Примеры:

   ```
   require_once(t3lib_extMgm::extPath('lang', 'lang.php'));
   require_once(t3lib_extMgm::extPath('lang') . 'lang.php');
   ```

Всегда пользуйтесь одним из двух предложенных способов включения файлов. Это требование распространяется даже при включении файлов из той же директории. Некоторые установки не содержат текущей директории в пути включения PHP и `require_once` без верно указанного пути приведет к фатальной ошибке PHP.

Блок информации о классе

Блок информации о классе схож с блоком информации о файле, но описывает класс в файле. Пример:

```
/**
```

```
 * This class provides XYZ plugin implementation.
 *
 * @author John Doe <john.doe@example.com>
 * @author Jane Doe <jane.doe@example.com>
 */
```

Класс PHP

PHP класс идет вслед за блоком информации о классе. Код PHP должен быть отформатирован, как описано в главе “Форматирование синтаксиса PHP” на странице 14.

Название класса должно следовать некоторым соглашениям. Пространство имен и части пути (смотрите "Пространства имен" на странице 9) должны быть в нижнем регистре и разделяться подчеркиванием (“_”). В конце идет "настоящее" название класса, которое должно быть написано в горбатом стиле.

Снова возьмем как образец файл `class.t3lib_cache_backend_abstractbackend.php`, объявление класса PHP выглядит так:

```
class t3lib_cache_backend_AbstractBackend {
      …
}
```

Декларация XCLASS

Объявление XCLASS должно идти вслед за классом PHP. Формат XCLASS очень важен. Нельзя добавлять или удалять пробелы, никакого переформатирования не должно быть при объявлении. Если формат изменится, модуль управления расширениями TYPO3 пожалуется на ошибочное объявление XCLASS.

Объявление XCLASS должно включать правильный путь к текущему файлу класса. В следующем примере подразумевается, что ключ расширения – `myext`, название файла – `class.tx_myext_pi1.php`, а файл расположен в поддиректории `pi1` расширения:

```
if (defined('TYPO3_MODE') && isset($GLOBALS['TYPO3_CONF_VARS'][TYPO3_MODE]['XCLASS']
['ext/myext/pi1/class.tx_myext_pi1.php'])) {
      include_once($GLOBALS['TYPO3_CONF_VARS'][TYPO3_MODE]['XCLASS']
['ext/myext/pi1/class.tx_myext_pi1.php']);
}
```

Возможный код выполняемого модуля

Код выполняемого модуля использует класс и его метод(ы). Обычно, такой код можно найти в сценариях eID и старых модулях внутреннего интерфейса. Вот как он может выглядеть:

```
$controller = t3lib_div::makeInstance('tx_myext_ajaxcontroller');
$controller->main();
```

Этот код должен появиться **после** объявления XCLASS. `$SOBE` — традиционное но не требуемое название.

Форматирование синтаксиса PHP

Идентификаторы

Все идентификаторы должны использовать `горбатыйСинтаксис (camelCase)` и начинаться с буквы нижнего регистра. Символы подчеркивания не разрешаются. Примеры правильных идентификаторов:

```
$goodName
$anotherGoodName
```

Примеры неправильных идентификаторов:

```
$BAD_name
$unreasonablyLongNamesAreBadToo
$noAbbrAlwd
```

Правило горбатого синтаксиса в нижнем регистре применяется и к акронимам. Так:

```
$someNiceHtmlCode
```

правильно, а

```
$someNiceHTMLCode
```

неверно.

В частности, нужно избегать аббревиатур "FE" и "BE", а вместо этого использовать полные слова "Frontend" и "Backend".

Названия идентификаторов должны быть понятны. Но возможно использовать обычные целочисленные переменные, вроде `$i`, `$j`, `$k` в циклах `for`. При использовании таких переменных, их значение должно быть понятно из контекста, в котором они используются.

Те же правила относятся к функциям и методам классов. Примеры:

```
protected function getFeedbackForm()
public function processSubmission()
```

Константы класса должны ясно показывать, что они определяют. Правильно:

```
const USERLEVEL_MEMBER = 1;
```

Неправильно:

```
const UL_MEMBER = 1;
```

Переменные в глобальной области могут использовать верхний регистр и символы подчеркивания.

Примеры:

```
$TYPO3_CONF_VARS
$TYPO3_DB
```

Комментарии

Комментарии в коде одобряются и крайне рекомендуются. Встроенные комментарии должны предшествовать комментируемой строке и отстоять на знак табуляции. Пример:

```
protected function processSubmission() {
            // Check if user is logged in
      if ($GLOBALS['TSFE']->fe_user->user['uid']) {
            …
      }
}
```

Комментарии должны начинаться с "`//`". Начинающиеся с "`#`" недопустимы.

Константы и переменные класса должны следовать стилю документа PHP и комментарии должны предшествовать им. Должен быть описан нетривиальный тип переменных и, по-возможности, тривиальные типы. Пример:

```
/** Number of images submitted by user */
protected $numberOfImages;
/**
 * Local instance of tslib_cObj class
 *
 * @var tslib_cObj
 */
```

```
        protected $localCobj;
```

Однострочный комментарий допустим, если тип переменной или константы в классе еще не был объявлен.

Если переменная может содержать значения разных типов, используйте в качестве типа `mixed`.

Вывод отладочной информации

При разработке допустимо использовать вызов функций `debug()` или `t3lib_div::debug()` для вывода отладочной информации. Но все отладочные операторы должны быть удалены (удалены, но не закомментированы!) перед передачей кода в репозиторий подверсий.

Фигурные скобки

Требуется использовать открывающие и закрывающие скобки во всех классах, где они могут использоваться, согласно синтаксису PHP (за исключением блоков `case`).

Открывающая фигурная скобка всегда должна находиться на одной строке с предшествующей конструкцией. Пред открывающие скобкой должен быть пробел (не таблуяция!). За открывающие скобкой всегда идет новая строка.

Закрывающая фигурная скобка всегда должна начинаться на новой строке, на уровне конструкции, к которой относится открывающая скобка. Пример:

```
protected function getForm() {
      if ($this->extendedForm) {
            // generate extended form here
      } else {
            // generate simple form here
      }
}
```

Следующее недопустимо:

```
protected function getForm()
{
      if ($this->extendedForm) { // generate extended form here
      } else {
            // generate simple form here
      }
}
```

Условия

Условия состоят из ключевых слов `if`, `elseif` и `else`. В коде TYPO3 не должна использоваться конструкция `else if`.

Это — корректый шаблон для условий:

```
        if ($this->processSubmission) {
              // Process submission here
        } elseif ($this->internalError) {
              // Handle internal error
        } else {
              // Something else here
        }
```

Вот пример некорректного шаблона:

```
        if ($this->processSubmission) {
              // Process submission here
        }
        elseif ($this->internalError) {
              // Handle internal error
```

```
} else // Something else here
```

Рекомендуется создавать условия, чтобы первым шел наименьший блок. Например:

```
if (!$this->processSubmission) {
      // Generate error message, 2 lines
} else {
      // Process submission, 30 lines
}
```

Если условие длинное, его можно разбить на несколько строк. Каждая строка условия, начиная со второй, должна быть отбита двумя или более отступами, в соответствие с первой строкой условия:

```
if ($this->getSomeCodition($this->getSomeVariable()) &&
            $this->getAnotherCondition()) {
      // Code follows here
}
```

Тройной оператор условия должен использоваться только если он имеет два результата. Пример:

```
$result = ($useComma ? ',' : '.');
```

Неверное использование тройного условного оператора:

```
$result = ($useComma ? ',' : $useDot ? '.' : ';');
```

Необходимо избегать использования присвоения в условиях. Но если имеет смысл делать присвоение в условии, оно должно быть заключено в дополнительную пару скобок. Пример:

```
if (($fields = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res))) {
      // Do something
}
```

Следующее допустимо, но не рекомендуется:

```
if (FALSE !== ($fields = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res))) {
      // Do something
}
```

Следующее недопустимо (пропущена дополнительная пара скобок):

```
while ($fields = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res)) {
      // Do something
}
```

Switch

Блок `case` отделяется одним отступом (табуляцией) внутри блока `switch`. Код внутри блока `case` отделяется еще одним отступом. Блок `break` стоит на уровне внутреннего кода. На один блок `case` разрешается использовать лишь один блок `break`.

Блок `default` должен быть последним в блоке `switch` и не должен иметь блока `break`.

Если один из блоков `case` имеет переход в другой блок `case` без блока `break`, то об этом должно быть указано в комментарии.

Примеры:

```
switch ($useType) {
      case 'extended':
            $content .= $this->extendedUse();
            // Fall through
      case 'basic':
            $content .= $this->basicUse();
            break;
      default:
```

```
                $content .= $this->errorUse();
        }
```

Циклы

Можно использовать следующие циклы:

- `do`
- `while`
- `for`
- `foreach`

Недопустимо использование `each` в циклах.

Циклы `for` внутри должны содержать только переменные (без вызовов функций). Следующее правильно:

```
$size = count($dataArray);
for ($element = 0; $element < $size; $element++) {
      // Process element here
}
```

А это – недопустимо:

```
for ($element = 0; $element < count($dataArray); $element++) {
      // Process element here
}
```

Циклы `do` и `while` должны использовать дополнительные скобки, если в цикле используется присвоение:

```
while (($fields = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res))) {
      // Do something
}
```

Специально оговаривается случай циклов `foreach`, когда значение не используется внутри цикла. При этом используется шаблонная переменная `$_` (подчеркивание):

```
foreach ($GLOBALS['TCA'] as $table => $_) {
      // Do something with $table
}
```

Это сделано из соображений производительности, так как это быстрее, чем вызов `array_keys()` и зацикливание на выдаваемом им результате.

Строки

Во всех строках необходимо использовать одиночные кавычки. Двойные кавычки используются лишь для символа перехода строки (“`\n`”).

Оператор связывания строк должен быть окружен пробелами. Пример:

```
$content = 'Hello ' . 'world!';
```

Но не должно присутствовать пробела после оператора связывания строк, если он является последним в строке.

Переменные не должны быть вставлены в строку. Корректно:

```
        $content = 'Hello ' . $userName;
```

Неверно:

```
        $content = “Hello $userName”;
```

Допустимо связывать несколько строк. Оператор связывания строк должен быть в конце

строки. Связываемые строки, начиная со второй, должны быть смещены относительно первой. Рекомендуется делать один отступ относительно начала строки первого уровня:

```
$content = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. ' .
                'Donec varius libero non nisi. Proin eros.';
```

Булевы выражения

Булевы выражения должны использовать конструкции языка PHP и не подобные целочисленные значения, вроде 0 или 1. Кроме того, они должны быть записаны в верхнем регистре, т. е. `TRUE` и `FALSE`.

Массивы

Объявление массивов должно использовать ключевое слово "array" в нижнем регистре, без пробелов между ним и открывающей скобкой. Так:

```
$a = array();
```

Каждый из компонентов массива декларируется на отдельной строке. Сами строки делаются с отступами в одну табуляцию от начала объявления. Закрывающая скобка должна находиться на том же уровне, что и переменные. Каждая строка с элементом массива должна оканчиваться запятой. Они может быть опущена, если следующих элементов нет, по выбору разработчика. Пример:

```
$thisIsAnArray = array(
      'foo' => 'bar',
      'baz' => array(
            0 => 1
      )
);
```

Вложенные массивы следуют тому же правилу. Правила форматирования касаются и крайне маленьких массивов с простыми объявлениями, например:

```
$a = array(
      0 => 'b',
);
```

NULL

Это специальное значение записывается в верхнем регистре, т. е. `NULL`.

Особенности PHP5

Использование особенностей PHP5 крайне рекомендуется для расширений и обязательно для ядра TYPO3 версии 4.2 или выше.

Функции класса должны иметь спецификатор типа доступа: `public, protected` или `private`. Обратите внимание, что `private` должны предотвращать XCLASSификацию класса. Поэтому, `private` может использоваться только в случае крайней необходимости.

Переменные класса должны использовать спецификатор доступа, вместо ключевого слова `var`.

Должен использоваться намек на класс, когда для функции ожидается массив или экземпляр какого-либо класса. Пример:

```
protected function executeAction(tx_myext_action& $action, array $extraParameters) {
      // Do something
}
```

Статичная функция должна иметь ключевое слово `static`. Этот ключ должен идти первым

в определении функции:

```
	static public function executeAction(tx_myext_action& $action, array $extraParameters)
{
		// Do something
	}
```

Ключ `abstract` также должен стоять первым в объявлении функции:

```
	abstract protected function render();
```

Глобальные переменные

Использование `global` не рекомендуется. Всегда используйте `$GLOBALS['variable']`.

Функции

Если функция возвращает значение, то она всегда должна его возвращать. Следующее недопустимо:

```
function extendedUse($enabled) {
	if ($enabled) {
		return 'Extended use';
	}
}
```

А вот правильное употребление:

```
function extendedUse($enabled) {
	$content = '';
	if ($enabled) {
		$content = 'Extended use';
	}
	return $content;
}
```

В общем, для функции должно быть единственно возможное возвращаемое значение (смотрите предыдущий пример). Но функция может возвращать замечания во время проверки параметров, перед тем, как начать их основную обработку. Пример:

```
function extendedUse($enabled, tx_myext_useparameters $useParameters) {
	// Validation
	if (count($useParameters->urlParts) < 5) {
		return 'Parameter validation failed';
	}

	// Main functionality
	$content = '';
	if ($enabled) {
		$content = 'Extended use';
	} else {
		$content = 'Only basic use is available to you!';
	}
	return $content;
}
```

Функции не должны быть длинными. "Длина" не относится к длине строк. Общее правило — функция должна умещаться на $^2/_3$ экрана. При этом можно вносить в функцию небольшие изменения, без нужды ее дальнейшего разбиения.

Использование phpDoc

phpDoc полезно для документирования исходного кода. Обычно, в коде TYPO3 используются следующие ключевые слова phpDoc:

- `@author`
- `@access`

- `@global`
- `@param`
- `@package`
- `@return`
- `@see`
- `@subpackage`
- `@var`
- `@deprecated`

За дополнительной информацией по phpDoc, обратитесь на веб сайт phpDoc по адресу: `http://www.phpdoc.org/`

TYPO3За дополнительной информацией по phpDoc, обратитесь на веб сайт phpDoc по адресу “Блок информации о классе” на странице 13.

Обратите внимание, что тег `@author` **не должен** использоваться в блоках комментариев phpDoc для функции или метода — только на уровне класса, так как изменения происходят слишком часто и, зачастую, авторство становится неясным. Достаточно `git blame` для отслеживания изменений.

Блок информации о функии

Функции должны иметь задокуменнтированные параметры и тип возвращаемого результата. Пример:

```
/**
 * Initializes the plugin.
 *
 * Checks the configuration and substitutes defaults for missing values.
 *
 * @param array $conf Plugin configuration from TypoScript
 * @return boolean TRUE if initialization was successful, FALSE otherwise
 * @see tx_myext_class:anotherFunc()
 */
protected function initialize(array $conf) {
      // Do something
}
```

Краткое и полное описания

Метод или класс могут иметь как краткое, так и полное описания. Краткое описание — это первая часть текста внутри блока phpDoc. Оно оканчивается пустой строкой. Любой дополнительный текст после строки и перед первым тегом является полным описанием.

Не забывайте об использовании `@return void`, когда функция не возвращает значений.

Файл ChangeLog

В ядре TYPO3 имеется файл `ChangeLog`, куда заносятся все изменения. До недавнего времени он заполнялся вручную, но с переходном на систему управления версиями Git первого марта 2011, он ведется автоматически.

Программирование: лучшие методы

В этом разделе собраны лучшие методы при разработке для TYPO3.

Доступ к базе данных

Доступ к базе данных в TYPO3 всегда должен осуществляться посредством использования `$GLOBALS['TYPO3_DB']`. Это — экземпляр класса `t3lib_db` из `t3lib/class.t3lib_db.php`.

Те же правила применимы при доступе к базе данных не TYPO3: доступ к ним осуществляется посредством разных экземпляров того же класса. Несоблюдение этого правила может повредить базу данных TYPO3 или доступ к базе данных будет отсутствовать.

Шаблоны проектирования (Singletons)

TYPO3 поддерживает шаблоны проектирования для классов. В одном запросе HTTP должен быть лишь один экземпляр шаблона, независимо от количества вызовов в `t3lib_div::makeInstance()`. Для использования шаблонов проектирования классов должен применяться интерфейс `t3lib_Singleton`:

```
require_once(PATH_t3lib . 'interfaces/interface.t3lib_singleton.php');

class tx_myext_mySingletonClass implements t3lib_Singleton {
      …
}
```

В этом интерфейсе нет методов.

Статические методы

Когда класс вызывает один из своих собственных статических методов (или методов своих родителей), в коде необходимо указывать ключевое слово `self`, вместо названия класса.

Пример

```
class tx_myext_MyClass {
    public static function methodA() {
      //...
    }
    public static function methodB() {
      self::methodA(); // instead of tx_myext_MyClass::methodA();
    }
}
```

Локализация

TYPO3 спроектирована, как полностью локализуемая система. Необходимо всячески избегать строк, жестко внесенных в код, несмотря на некоторые технические ограничения (например, некоторые ранние или низкоуровневые фрагменты кода, где не доступен объект `lang`).

Определение локализуемых строк

Некоторые правила, которые нужно соблюдать при работе с метками в файлах locallang:

– Всегда проверяйте наличие локализуемой строки в уже существующих общих файлах locallang, в частности `EXT:lang/locallang_common.xml` и

`EXT:lang/locallang_core.xml`.

- Локализуемые строки никогда не должны быть полностью в верхнем регистре. При необходимости, должен применяться соответствующий метод для перевода в верхний регистр.
- Локализуемые строки не должны разбиваться на несколько частей для включения фрагмента по середине. Вместо этого используйте одну строку с маркерами `sprintf()` (`%s`, `%d`, и т. п.).
- Когда локализуемая строка содержит несколько маркеров `sprintf()`, она **должна** использовать нумерованные аргументы (например, `%1$d`).
- Знаки препинания **должны** включаться в локализуемые строки, включая завершающие знаки, которые могут отличаться в разных языках (например “?” и “¿”). Кроме того, в некоторых языках перед знаками препинания используется пробел.
- Локализуемые строки не предназначены для тегов HTML, за исключением подготовки для контекстной справки (CSH). По возможности этого следует избегать.

Как только локализуемая строка появилась в опубликованной версии TYPO3, она не может быть изменена (несмотря на грамматические и орфографические ошибки). И не может быть удалена. Если метка локализуемой строки должна измениться, вместо нее должна быть сделана другая.

Использование локализуемых строк

Локализуемые строки выводятся посредством доступной API, в основном `lang::getLL()`, когда загружен соответствующий файл locallang, или иначе `lang::sL()`. В обоих методах, второй параметр при вызове должен быть пустым, несмотря на то, что он принудительно устанавливается в `TRUE` (что включает использование `htmlspecialchars()`).

Блочное тестирование (Unit tests)

Использование блочного тестирования

Несмотря на неполноту, уже существует довольно много блочных тестов для ядра TYPO3. При любом изменении ядра, должны быть запущены блочные тесты, чтобы удостовериться в ожидаемых результатах работы.

Добавление блочных тестов

Использование блочных тестов всячески поощряется. Всякий раз при добавлении новой функции или изменении существующей, должны добавляться блочные тесты.

Обработка устаревшего кода (deprecation)

Этот раздел описывает правила для удаления существующих функций или параметров из TYPO3. Общий принцип заключается в том, что функции или параметры, объявленные устаревшими, удаляются через **две основные версии** с момента объявления.

Для инициализации процесса устаревания (deprecation process) для параметра функции из ядра TYPO3, дайте пометку внутри части параметров phpDoc:

```
/**
 * ...
 * @param   string      DEPRECATED since TYPO3 4.X - is not used anymore because...
 * ...
 */
```

Для всей функции — внутри одного из классов ядра TYPO3, используя параметр phpDoc `@deprecated`:

```
/**
 * ...
 * @return...
 * @deprecated since TYPO3 4.X - is not used anymore, use FUNCNAME instead
 */
```

Начиная с TYPO3 версии 4.X+2, любой может предоставить исправление для удаления устаревших фрагментов.